Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 81 (2019) 282–287

www.elsevier.com/locate/procedia

52nd CIRP Conference on Manufacturing Systems

A method to evaluate interface compatibility during production system design and reconfiguration

Niko Siltala[a,*], Eeva Järvenpää[a], Minna Lanz[a]

[a]*Faculty of Engineering and Natural Sciences, Tampere University, Korkeakoulunkatu 6, 33720 Tampere, Finland*

* Corresponding author. Tel.: +358-40-536-6017 . *E-mail address:* niko.siltala@tuni.fi

Abstract

Manufacturing companies operate in volatile environment, where agility and responsiveness to change are desired from the production systems. Such responsiveness could be facilitated by computerized methods for resource selection and system design. We have developed a capability based matchmaking method, which compares the product requirements against resource capabilities, and which proposes resource combinations meeting these requirements. One part of this method is interface matchmaking algorithm, which is presented in this paper. Interface matchmaking method evaluates if the proposed set of resources can be connected physically together. It utilizes the formalized interface information provided by the resource descriptions.

© 2019 The Authors. Published by Elsevier Ltd.
This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/3.0/)
Peer-review under responsibility of the scientific committee of the 52nd CIRP Conference on Manufacturing Systems.

Keywords: resource interfaces; interface matchmaking

1. Introduction

Responsiveness of manufacturing is an important strategic goal for manufacturing companies operating in a highly dynamic environment characterized 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.

Traditionally, the production 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 [3]. Meeting the requirements of fast adaptation calls for new methods and solutions that would drastically reduce the time and effort put into system design [2, 4], both in brownfield and greenfield scenarios. Plug and play interfaces, modern information and communication technologies, formal information models representing resources and products, as well as simulations and other computer-aided intelligent planning tools can all contribute to such methods and solutions [2, 4]. During the system design and re-configuration, new structural configurations are built to fulfil the functional requirements set by the product [4]. Similar to the design of modular products [5]; consideration of interfaces plays an important role in enabling the interchangeability and independence of resource elements. Thus, in order to achieve a feasible structural configuration, the combined production resources must have compatible interfaces.

Within the past decade, there have been multiple different projects and research [6-9] trying to provide computerized support for system design and reconfiguration planning process. 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 standardized interfaces to promote interchangeability, presents the possibility for developing

2212-8271 © 2019 The Authors. Published by Elsevier Ltd.
This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/3.0/)
Peer-review under responsibility of the scientific committee of the 52nd CIRP Conference on Manufacturing Systems.
10.1016/j.procir.2019.03.049

automated system design and reconfiguration methods. Important steps towards modular assembly equipment and standardized hardware and control interfaces was made, for example, in EU-funded project EUPASS [7]. In SkillPro project, an approach basing on relation between product, process, resource, and skill was proposed [8]. The recently finished project ReCaM [9], whose results this paper also reports, aimed to develop a set of integrated tools for rapid and autonomous reconfiguration of production systems. The approach relies on a formal unified functional description of resources [10], providing a foundation for rapid creation of new system configurations through capability-based matchmaking of product requirements and resource offerings [11].

The first objective of this paper is to present how the interface concept for production resources can be utilized as part of our capability matchmaking [12] procedure. Second, to present the algorithm used to filter out those production resource combinations (e.g. manipulator and gripper) generated by the capability matchmaking, which cannot be physically connected with each other.

The paper is organized as follows. The second chapter introduces our overall capability matchmaking approach and its associated concepts. Chapter three focuses on the interface matchmaking method in general and associated ontology (data model). In chapter four we describe the underlying interface matchmaking algorithm, including some implementation aspects. In the final chapter we conclude the work done and its implications to the science and technology in general.

2. Capability Matchmaking

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

2.1. Involved information models

We have defined several ontologies as connected information models [13]. A central model of them is Manufacturing Resource Capability Ontology (MaRCO) [13], which is a Web Ontology Language (OWL)-based information model that can be used to describe capabilities, i.e. functionalities, of resources and resource combinations. MaRCO imports another ontology called Process Taxonomy Model, which categorizes different manufacturing and assembly processes in a hierarchical structure. 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 (SPARQL Protocol and Resource Description Framework Query Language) queries. Detailed information about MaRCO can be found from our earlier publications [13, 14]. While the MaRCO models the capabilities of resources, Resource Interface ontology [15] models the interfaces of production resources. The latter will be opened more in details in next sections. MaRCO imports also the Resource Interface ontology.

We have developed another ontology to describe the product requirements. A Product Model ontology [12] describes the parts and their basic characteristics, sub-assemblies and their contained parts, processes related to the parts and sub-assemblies, capability requirements related to the processes, and sequence of the processes. Also, the Product Model imports the same Process Taxonomy as the Capability Model. This allows to build a link between the requirements and provided capabilities. All our ontologies can be found from [16].

2.2. Overall matchmaking process

The overall matchmaking process [11] has three stages, which all require their specific algorithms and rules: 1) Defining the combined capabilities and calculating their parameters when new resource combinations are formed [17]; 2) Checking the interface compatibility of the resources when new resource combinations are formed [15]; and 3) Matching the product requirements against the capabilities of the combined resources [18]. This paper focuses on elaborating the actions performed during the second stage.

The matchmaking process takes inputs from external design and planning systems, which control the matchmaking process. The required inputs are a Product Requirement Description (PRD) and Resource Descriptions (RDs) of the production resources as a Resource Pool. In case of reconfiguration scenario, a description of the existing production system (a System Layout) should also be provided as an input. These inputs form the search space for the matchmaking. The search space is read into a Matchmaking Ontology. The Matchmaking Ontology imports both the MaRCO and the Product Model ontologies and contains rules that are used to compare the product requirements against the provided capabilities, and to make match between those. [12]

The capability-matching process takes the capability requirements and matches them with the existing capabilities or create new resource combinations that match with the requirements. The found matches to each process step are then provided back to the external design tools, which will then show the results to the system designer for resource selection and system configuration e.g. sorted by the availability or other valued criteria. The interface matchmaking is one of the sub-processes for the overall matchmaking process, and it is presented in the next sections.

3. Interface matchmaking

The interface matchmaking refers to process of finding out physically connectable and compatible production resources from the hardware interface point of view. This can be illustrated with a few use case scenarios, which are opened more in [15]:

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 resource's interfaces;
3) To find all possible resource combinations, which can be connected together. This is generalized case of the first.
4) To analyze if the connections in the proposed system layout or a set of resources are connectable from the interface point of view. This case focuses on two selected resources, and their connection.

In the overall matchmaking process, the interface matchmaking is done in together with the resource combination generation. The capability matchmaking finds out possible resource combinations, which are then tested for the interface connection. Thus, in this paper, we will focus the use case scenario 4 (from above), even the other scenarios are also touched by the implementation of interface matchmaking algorithm. List of resources to be combined will come as an input and the output should be true or false, depending on if these resources can be connected from the interface point of view or not.

In all these four scenarios, the interface matching can be done at two different levels of detail. At first, coarse level matching analyses only the interface identifier (ID) and the gender information. The second level is fine level matching, which uses additionally the interface characteristics information and associated rules (ifCompatibilityCheckOperator). Section 3.3 focuses on the coarse level and Section 3.4 on the fine level matching, but first we explain some necessary preparations before interface matchmaking can take place.

3.1. Resource Interface Ontology

Fig. 1 represents the data model of the Resource Interface ontology, which is used to provide formalized information for interface matchmaking. [15] presents it in details, and only the main points for interface matchmaking are highlighted here. A production resource is presented as a *DeviceBlueprint* class. It has one or more *InterfaceDefinition*s, which present an implementation of *IFStandard*. *IFStandard* is a definition of international, de-facto, or company specific interface specification. The Resource Interface ontology formalizes the main characteristics of such specification through *IFStandard* and *IfStdCharacteristic* classes. The latter captures interface characteristics including the variant information provided in the specification, like mechanical size, type of electrical connection, or communication protocol used. While the *IfStdCharacteristic* presents all possible values for a variant, the corresponding *IfCharacteristic* class picks the value used for this specific *InterfaceDefinition*. This information is used to evaluate the physical fit between two resource modules, such as mechanical connections and their dimensions.

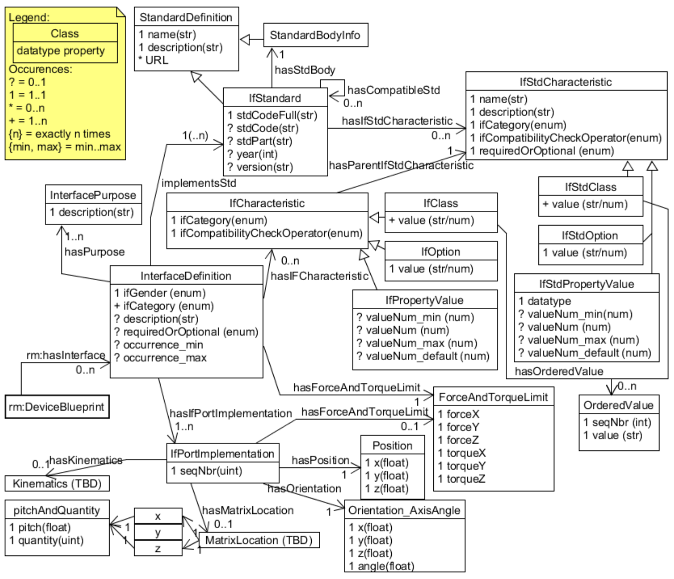

Fig. 1. Resource Interface Ontology (Modified from [15]).

3.2. Preparations for interface matchmaking

One input for the interface matchmaking is a populated Resource Model ontology containing the description of resource interfaces in format of Resource Interface ontology. Therefore, it needs to be prepared before the interface matchmaking can start. The search space for our matchmaking is a resource pool(s) and optionally a system layout. Each physical production resource present on the search space has its own formalized Resource Description (RD). A RD contains comprehensive description of characteristics and capabilities of a resource. The resource provider provides it and publishes them through a Resource Catalogue(s). The Resource Description concept is discussed in details in [10, 19], and utilization and value proposition of it in [20].

In the preparation phase, all resources belonging to the matchmaking search space are processed. The RD of each resource is read in and new or linked instances corresponding the Resource Interface ontology are created and populated on a new ontology. This ontology is then provided as one input for the interface matchmaking.

3.3. Coarse level matching

There are two property values of interface, which are used at the coarse level interface matching. The first is the *stdCodeFull* from *IfStandandard* class, which is the primary linking factor when judging, if two interface implementations, present in two resources, can be connected together. The second is the gender (*ifGender*) from *InterfaceDefinition* class, and it can have one of the three enumerated values - male, female, or neutral. This defines polarity of the interface, and which implementations of the same *IfStandard* can be connected together. The rule is simple - male and female or two neutrals can be connected together. Examples of appointing a gender are such as: a plug is male, a socket is female, and a plain flange with mounting through holes could form an interface with neutral gender.

3.4. Fine level matching and rules for interface characteristics

The fine 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 check operator* (=rule) 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) *IfCharacteristics* of an interface provide a positive match. Each of the *IfCharacteristic* has one *ifCompatibilityCheckOperator*. The operator specifies how the values from the source resource are compared with the target resource. The comparison follows a template - <Source> is/contains <Operator> (as) <Target>?

In [15] we have defined twelve different compatibility check operators with mathematical formulation, which can be used as an interface matching rule. Fig. 2 illustrates a few selected operators, and matching with sets and numerical values. For example, the operator "SAME_SET" expects at both sides of the connection (i.e. source and target sides) exactly same value or set of values. It applies also if characteristic is a numerical value or a range. Practical application of operator "SAME_SET" could be a gripper (source), which implements an interface with size variant 20. This can be connected only with manipulators (target) having the same size variant 20 defined as implementation. Another example of operator is "INSIDE_RANGE", which is applicable only for numerical values. In this case, interfaces are connectable only and only if the source side value or range remains inside range of the target. Source set $S_{2.1}$ [3,5] is inside range of the target set (T) [2,5] in Fig. 2. Thus, in such case a positive match can occur. If source set is $S_{2.3}$ [4,6], interface match is not possible with the same operator and target set (T), because $S_{2.3}$ extends above the T.

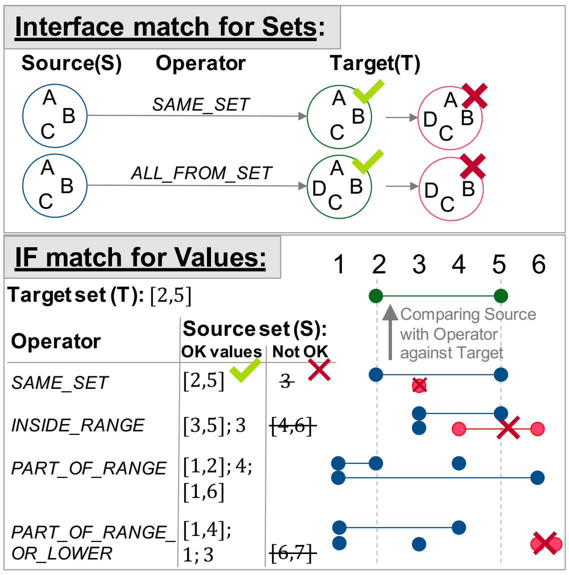

Fig. 2. A few interface compatibility operators, and illustration how source interface matches or doesn't match with target.

4. Interface matchmaking algorithm

SPARQL is used to make queries to the Resource Interface ontology. An interface matchmaking Application Program Interface (API) has been developed to run and process them in sequences, and to provide additional filtering of the results. This API hides from user the SPARQL queries, as one request requires running several different queries, calling them and analyzing the acquired results. In addition, intermediate information is cached internally for subsequent interface matchmaking calls.

Certain level of optimization is performed with the share of API side processing versus SPARQL queries. For example, analyzing possible connections between multiple resources in use case scenarios one and three from beginning of Chapter 3, one can run sequence of queries associated to the scenario 1

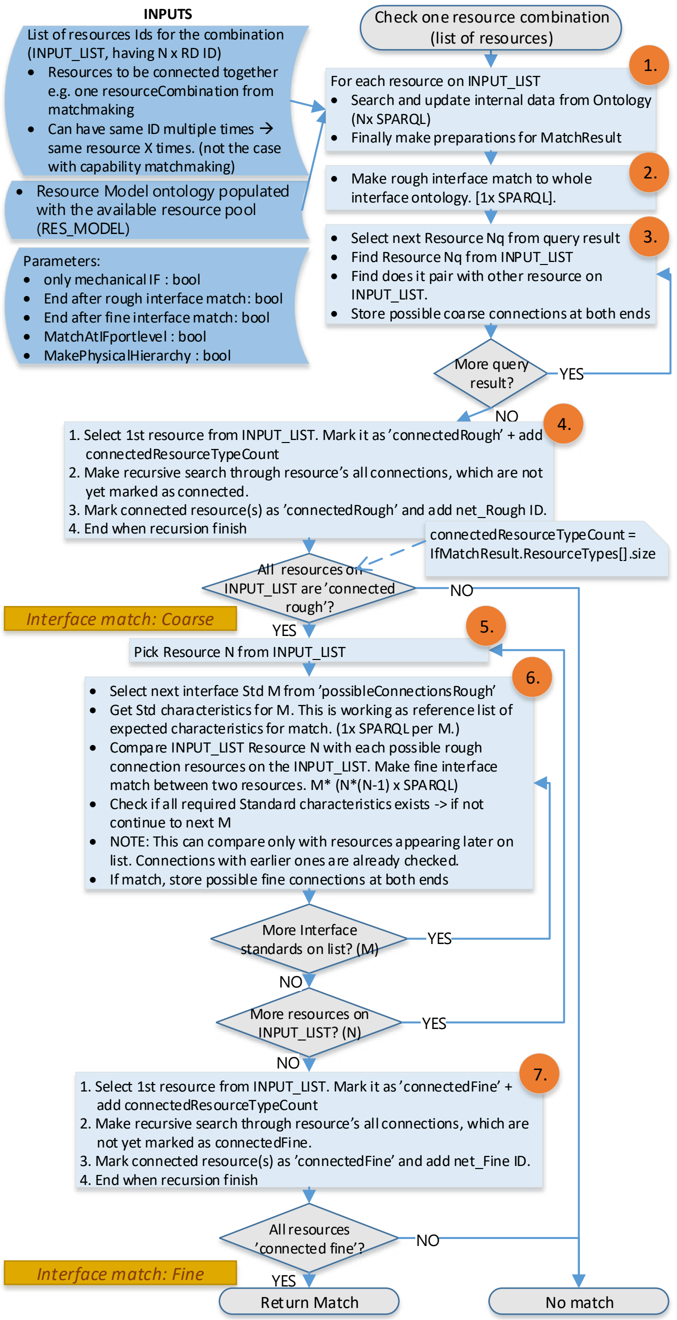
INPUTS
List of resources Ids for the combination (INPUT_LIST, having N x RD ID)
• Resources to be connected together e.g. one resourceCombination from matchmaking
• Can have same ID multiple times → same resource X times. (not the case with capability matchmaking)
• Resource Model ontology populated with the available resource pool (RES_MODEL)
Parameters:
• only mechanical IF : bool
• End after rough interface match: bool
• End after fine interface match: bool
• MatchAtIFportlevel : bool
• MakePhysicalHierarchy : bool
Check one resource combination (list of resources)
1.
For each resource on INPUT_LIST
• Search and update internal data from Ontology (Nx SPARQL)
• Finally make preparations for MatchResult
2.
• Make rough interface match to whole interface ontology. [1x SPARQL].
3.
• Select next Resource Nq from query result
• Find Resource Nq from INPUT_LIST
• Find does it pair with other resource on INPUT_LIST.
• Store possible coarse connections at both ends
More query result?
YES
NO
4.
1. Select 1st resource from INPUT_LIST. Mark it as 'connectedRough' + add connectedResourceTypeCount
2. Make recursive search through resource's all connections, which are not yet marked as connected.
3. Mark connected resource(s) as 'connectedRough' and add net_Rough ID.
4. End when recursion finish
connectedResourceTypeCount = IfMatchResult.ResourceTypes[].size
All resources on INPUT_LIST are 'connected rough'?
NO
Interface match: Coarse
YES
5.
Pick Resource N from INPUT_LIST
6.
• Select next interface Std M from 'possibleConnectionsRough'
• Get Std characteristics for M. This is working as reference list of expected characteristics for match. (1x SPARQL per M.)
• Compare INPUT_LIST Resource N with each possible rough connection resources on the INPUT_LIST. Make fine interface match between two resources. M* (N*(N-1) x SPARQL)
• Check if all required Standard characteristics exists -> if not continue to next M
• NOTE: This can compare only with resources appearing later on list. Connections with earlier ones are already checked.
• If match, store possible fine connections at both ends
More Interface standards on list? (M)
YES
NO
More resources on INPUT_LIST? (N)
YES
NO
7.
1. Select 1st resource from INPUT_LIST. Mark it as 'connectedFine' + add connectedResourceTypeCount
2. Make recursive search through resource's all connections, which are not yet marked as connectedFine.
3. Mark connected resource(s) as 'connectedFine' and add net_Fine ID.
4. End when recursion finish
All resources 'connected fine'?
NO
Interface match: Fine
YES
Return Match
No match

Fig. 3. Interface matchmaking procedure.

and conclude the result by combining these results together. Alternative is to run only one SPARQL query (scenario 3) and then pick the interested resources out from the results and collect the final result. If the processing cost of a single SPARQL query is very high, then it is beneficial to follow the latter strategy.

Fig. 3 illustrates the algorithm for the interface matchmaking both at coarse and fine levels. The algorithm is divided in two phases. The steps from 1 to 4 (orange circles at right top corner of process) forms the coarse level interface matching phase and steps from 5 to 7 correspondingly the fine level phase. Next, the key points of the algorithm are explained.

4.1. Inputs for algorithm

First, the input request from outside is processed (three document blocks in top left corner of Fig. 3). The input contains three parts. The first is a list of resource IDs (INPUT_LIST) establishing a resource set for the tested resource combination. The second is the Resource Model ontology (RES_MODEL) including the Resource Interface ontology populated with resource instance information, as presented in section 3.2. The ontology can be the same for many adjacent interface matchmaking requests, and only the list of resource IDs will change. The third input is the control and configuration information, which is used to influence the operation of the algorithm, such as terminating the algorithm after coarse interface matching phase.

4.2. Coarse level interface matching

The first step of the algorithm (in Fig. 3) is still for the preparations. A SPARQL query is executed for each resource (N) provided on the INPUT_LIST. This query is used to collect information about each resource from the ontology and to initialize the internal data structures before the execution of the algorithm. Other internal initializations are performed like the initiation of the MatchResult object.

The second step executes one SPARQL query for searching all coarse level matches within the given RES_MODEL. It uses the standard's ID code and gender information for determining the coarse interface match. Fig. 4 shows the SPARQL query used in this step, and it works as the following: Line 2 starts the SPARQL query and selects what information is shown on the resulting records. Lines 4..8 select the resource(s), which are used as source resource for the interface matching, and all its interface descriptions. Line 9 defines which gender is accepted as the counterpart. Lines 10..13 look for counter part resources implementing the appropriate interface, and selects such resources as a target resource. Finally, line 14 filters out the records connecting the source to itself.

```
1 # Interface match: coarse. Find connectable
   interfaces of all modules AND find toMOs having
   connecting interface: NEUTRAL-NEUTRAL or MALE-
   FEMALE
2 SELECT DISTINCT ?fromMO ?fromIF ?fromIFStdCode
   ?fromGender ?toGender ?toIF ?toIFStdCode ?toMO
3 WHERE {
4 ?fromMO rdf:type/rdfs:subClassOf*
   rm:DeviceBlueprint .
5 ?fromMO rm:hasInterface ?fromIF .
6 ?fromIF rim:implementsStd ?fromIFStd ;
   rim:ifGender ?fromGender .
7 OPTIONAL { ?fromIF rim:hasPurpose ?purpose . }
8 ?fromIFStd rim:stdCodeFull ?fromIFStdCode .
9 bind(xs:string(if(?fromGender="NEUTRAL","NEUTRAL",
   xs:string(if(?fromGender="MALE", "FEMALE",
   "MALE")) )) as ?toGender) .
10 { ?toIFStd rim:stdCodeFull ?fromIFStdCode . }
   UNION {?toIFStd rim:hasCompatibleStd ?fromIFStd.}
11 ?toIFStd rim:stdCodeFull ?toIFStdCode .
12 ?toIF rim:implementsStd ?toIFStd ;
   rim:ifGender ?toGender .
13 ?toMO rm:hasInterface ?toIF .
14 FILTER (?fromIF != ?toIF) . }
```

Fig. 4. SPARQL example finding all interface matches at coarse level.

In the step 3, each record of the query result from step 2 is analyzed. First, each source resource (Nq) is searched from the INPUT_LIST. If it is found on the list, next is checked that can it be connected with the other resources on the INPUT_LIST. I.e. is the target resource found also on the INPUT_LIST? The possible coarse connection between these two resources is marked and stored internally at side of both resources.

In the step 4, resources on the INPUT_LIST are iterated and analyzed recursively for finding out if they create a single network. The procedure is illustrated in Fig. 5. The analysis starts from the first resource on the list and continues spreading recursively. A box represents a resource and number in an arc the recursion round. Always, when a new resource is reached, it is marked with flag *connectedRough* and its *netID* is set. If any of the resources is left orphan (like Res X in Fig. 5) after processing through the whole list, 'no match' (false) is returned and algorithm terminates. If single network is created out of all resources on the INPUT_LIST and only coarse level matchmaking is requested, 'match' (true) is returned. In other cases, the process will continue to the fine level interface matchmaking.

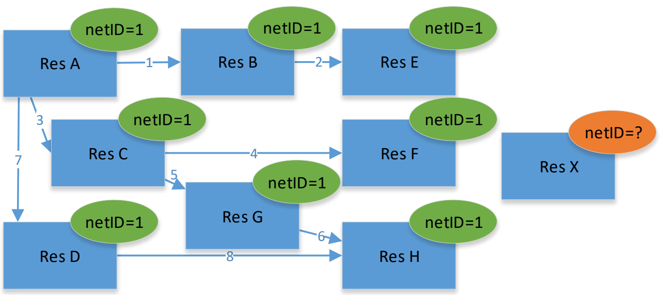

Fig. 5. Algorithm marking resources belonging to a connected network.

4.3. Fine level interface matching

The results from the coarse level interface matchmaking are utilized as input for the fine level interface matchmaking. Next, each resource N on the INPUT_LIST is processed one at the time (Step 5.). In the step 6, each standard M of the resource N, which has been marked as *connectedRough*, will be processed further. This means that through this standard interface there exist a connection with another resource at rough level, so it is a potential candidate for fine level connection.

First, in the step 6, a list of interface standard characteristics is figured out for each involved interface standard M. If the characteristic information does not yet exist in the cache, it is queried with a SPARQL query, processed, and stored to the cache. This creates an objective of the standard's characteristics for later comparisons. Then a parametric SPARQL query is executed between all resources marked as connected with resource N with interface standard M. It is enough to compare only with the resources appearing further on the INPUT_LIST, because the former are already marked as connected, if a connection between the resources does exist. In maximum N*M*(N-1) queries are executed in this phase. For each query result, it is analyzed, if all mandatory interface standard characteristics do exist. If they do, flag '*possibleFineConnection*' is marked at both resource's ends for this pair of connection. Otherwise, this resource connection is rejected. Then the iteration continues with next standard M, and after standards run out to the next resource N.

Finally, in the step 7, possible fine level connections information is resolved, with similar method as in coarse level matching (step 4), to determine if a single network (Fig. 5) is created out of the input resources. If this is the case, 'match' (true) is returned as result of the algorithm, false otherwise. This terminates the interface matchmaking algorithm.

5. Conclusions

This paper presented an interface matchmaking method for evaluating if a set of production resources can be connected physically from the interface point of view. The associated algorithm for making both coarse and fine level interface matchmaking was presented. The wider capability matchmaking method, and the presented approach as part of it, can facilitate rapid system design and reconfiguration planning, by allowing computerized methods to find feasible system configuration scenarios to different product requirements. Large resource catalogues containing thousands of resources and their variants can be automatically screened to find out the few appropriate resources. Using automatic matchmaking reduces and speeds up the manual design efforts, as the designer can focus his/her resource selection to truly connectable and fit resources, instead of searching for resources, and analyzing their interfaces. Additionally, the electrical resource catalogues can contain production resources (formal Resource Descriptions) from multiple vendors, which increases number of resources to study, but also available alternatives. Thus, matchmaking opens possibilities for new and more innovative solutions to be found. The designer is not bound to "old and known solutions", which is almost solving the requirements, but can select the optimum fit.

As a future work, we have some ideas to continue the interface matchmaking procedure to still finer level of detail including resource matching at interface port implementation level. This can extend the procedure for suggesting also physical layouts.

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] Koren Y, Shpitalni M. Design of reconfigurable manufacturing systems. J Manuf Syst 2010;29(4):130-41. doi:10.1016/j.jmsy.2011.01.001.

[2] Wiendahl H-P, ElMaraghy HA, Nyhuis P, Zäh MF, Wiendahl H-H, Duffie N, et al. Changeable Manufacturing - Classification, Design and Operation. CIRP Ann 2007;56:783–809. doi:10.1016/j.cirp.2007.10.003.

[3] Rösiö C, Säfsten K. Reconfigurable production system design – theoretical and practical challenges. J Manuf Technol Manag 2013;24:998–1018. doi:10.1108/JMTM-02-2012-0021.

[4] Westkämper E. Factory Transformability: Adapting the Structures of Manufacturing. Reconfigurable Manuf. Syst. Transform. Factories, Berlin, Heidelberg: Springer Berlin Heidelberg; 2006, p. 371–81. doi:10.1007/3-540-29397-3_19.

[5] Pakkanen J, Juuti T, Lehtonen T. Brownfield Process: A method for modular product family development aiming for product configuration. Des Stud 2016;45:210–41. doi:10.1016/j.destud.2016.04.004.

[6] Ferreira P, Lohse N, Ratchev S. Multi-agent Architecture for Reconfiguration of Precision Modular Assembly Systems. In: Ratchev S, editor. Precis. Assem. Technol. Syst., Springer; 2010, p. 247–54.

[7] EUPASS - Evolvable Ultra-Precision Assembly SystemS - project. EU FP6, GA No 507978 (2006). http://cordis.europa.eu/project/rcn/75342 en.html. [Accessed 27.12.2018]

[8] Pfrommer J, Schleipen M, Beyerer J. PPRS: Production skills and their relation to product, process, and resource. 2013 IEEE 18th Conf. Emerg. Technol. Fact. Autom., Cagliari, Italy: IEEE; 2013, p. 1–4. doi:10.1109/ETFA.2013.6648114.

[9] ReCaM consortium, ReCaM project web page, http://www.recam-project.eu. [Accessed 27.12.2018].

[10] Siltala N, Järvenpää E, Lanz M. Formal Information Model for Representing Production Resources. Adv Prod Manag Syst Initiat Sustain World APMS 2016 IFIP Adv Inf Commun Technol 2016;488:53–60. doi:10.1007/978-3-319-51133-7_7

[11] Järvenpää E, Siltala N, Hylli O, Lanz M. Capability matchmaking procedure to support rapid configuration and re-configuration of production systems. Procedia Manuf 2017;11:1053-60. doi:10.1016/j.promfg.2017.07.216.

[12] Järvenpää E, Siltala N, Hylli O, Lanz, M. Product Model Ontology and Its Use in Capability-Based Matchmaking. Procedia CIRP 2018;72:1094-9. doi:10.1016/j.procir.2018.03.211

[13] Järvenpää E, Siltala N, Hylli O, Lanz M. The development of an ontology for describing the capabilities of manufacturing resources. J Intell Manuf 2018; p.1-20. doi:10.1007/s10845-018-1427-6

[14] Järvenpää E, Lanz M, Siltala N. Formal Resource and Capability Models supporting Re-use of Manufacturing Resources. Procedia Manuf 2018;19:87-94. doi:10.1016/j.promfg.2018.01.013

[15] Siltala N, Järvenpää E, Lanz M. Creating Resource Combinations Based on Formally Described Hardware Interfaces. In: Ratchev S, editor. Precis. Assem. Technol. Syst. - 8th IFIP WG 5.5 Int. Precis. Assem. Semin. IPAS 2018, Chamonix, France: Springer Nature Switzerland AG 2019; 2019, p. 29–39. doi:10.1007/978-3-030-05931-6_3.

[16] Järvenpää E, Siltala N, Hylli O. Product, Manufacturing Resource and Capability Ontologies 2019. http://urn.fi/urn:nbn:fi:csc-kata20190225154330611362.

[17] Järvenpää E, Hylli O, Siltala N, Lanz M. Utilizing SPIN Rules to Infer the Parameters for Combined Capabilities of Aggregated Manufacturing Resources. IFAC-PapersOnLine 2018;51:84-9. doi:10.1016/j.ifacol.2018.08.239

[18] Järvenpää E, Siltala N, Lanz M. Formal resource and capability descriptions supporting rapid reconfiguration of assembly systems. 2016 IEEE Int. Symp. Assem. Manuf., IEEE; 2016, p. 120-5. doi:10.1109/ISAM.2016.7750724

[19] Siltala N. Formal Digital Description of Production Equipment Modules for supporting System Design and Deployment. Tampere University of Technology, 2016. http://urn.fi/URN:ISBN:978-952-15-3783-7

[20] Siltala N, Järvenpää E, Lanz M. Value Proposition of a Resource Description Concept in a Production Automation Domain. Procedia CIRP 2018;72:1106–11. doi:10.1016/j.procir.2018.03.154.