Journal of Intelligent Manufacturing (2019) 30:959–978
https://doi.org/10.1007/s10845-018-1427-6

The development of an ontology for describing the capabilities of manufacturing resources

Eeva Järvenpää[1] · Niko Siltala[1] · Otto Hylli[2] · Minna Lanz[1]

Received: 14 November 2017 / Accepted: 31 May 2018 / Published online: 15 June 2018

Abstract
Today's highly volatile production environments call for adaptive and rapidly responding production systems that can adjust to the required changes in processing functions, production capacity and dispatching of orders. There is a desire to support such system adaptation and reconfiguration with computer-aided decision support systems. In order to bring automation to reconfiguration decision making in a multi-vendor resource environment, a common formal resource model, representing the functionalities and constraints of the resources, is required. This paper presents the systematic development process of an OWL-based manufacturing resource capability ontology (MaRCO), which has been developed to describe the capabilities of manufacturing resources. As opposed to other existing resource description models, MaRCO supports the representation and automatic inference of combined capabilities from the representation of the simple capabilities of co-operating resources. Resource vendors may utilize MaRCO to describe the functionality of their offerings in a comparable manner, while the system integrators and end users may use these descriptions for the fast identification of candidate resources and resource combinations for a specific production need. This article presents the step-by-step development process of the ontology by following the five phases of the ontology engineering methodology: feasibility study, kickoff, refinement, evaluation, and usage and evolution. Furthermore, it provides details of the model's content and structure.

Keywords Manufacturing ontology · Resource description · Capability description · Adaptive manufacturing · Reconfigurable manufacturing

Introduction

The requirements for production systems are continuously shifting towards higher flexibility, making rapid responsiveness a new strategic goal for manufacturing enterprises along with quality and cost-effectiveness (Koren and Shpitalni 2010). This is due to increasing volatility in the global and local economies, shortening innovation and product life cycles, and a dramatically increasing number of product variants. Manufacturing companies need production systems that can rapidly adapt to the required changes in processing functions, production capacity, and dispatching of orders. Such system adaptation and reconfiguration are required on physical, logical and parametric levels (ElMaraghy 2006). The realization of these requirements calls for new solutions, such as planning methods and tools, that would drastically reduce the time and effort put into planning and implementing the alterations in a factory (Westkämper 2006), or allow autonomous adaptation during runtime, based on machine-to-machine communication and self-organization (Leitão et al. 2016).

Modular architecture paradigms for new production systems aim to promote system reconfigurability by interchangeable resources and components. These paradigms focus on the clear functional decoupling of module functionalities and the use of standardized interfaces (ElMaraghy

✉ Eeva Järvenpää
eeva.jarvenpaa@tut.fi

Niko Siltala
niko.siltala@tut.fi

Otto Hylli
otto.hylli@tut.fi

Minna Lanz
minna.lanz@tut.fi

[1] Laboratory of Mechanical Engineering and Industrial Systems, Tampere University of Technology, Korkeakoulunkatu 6, 33720 Tampere, Finland

[2] Laboratory of Pervasive Computing, Tampere University of Technology, Korkeakoulunkatu 1, 33720 Tampere, Finland

2006; Koren and Shpitalni 2010; Mehrabi et al. 2000), which opens up possibilities for developing automated adaptation methods. The fourth industrial revolution, in Europe referred to as "Industry 4.0" and in the US as "Smart Manufacturing", aims to tackle the above-mentioned issues by introducing the Internet of Things and servitization concepts into manufacturing (Thoben et al. 2017). These initiatives target so-called "cyber-physical systems" (CPS), which are physical systems, e.g. machines or workpieces, which combine with the digital world, i.e. sensors and intelligence, and are thus able to communicate, act and control themselves and each other (Baheti and Gill 2011; Yao et al. 2017). In Thoben et al.'s (2017) review of the current Industry 4.0 and Smart Manufacturing initiatives they identified the standardization of both interfaces and information models as one of the most relevant research issues for smart manufacturing.

The information needed for system design, reconfiguration planning and reactive adaptation, is based on extensive knowledge from different distributed sources in various fields of expertise. A major problem is the poor interoperability between the information and design support systems used to create, save, manage, and utilize this information. The majority of these systems use their own proprietary data structures and vaguely described semantics (Ray and Jones 2006; Lohse 2006; Matsokis and Kiritsis 2010). Also, the different actors involved in the process—humans or organizations—may have different understanding of the used terms, leading to interoperability issues (Ray and Jones 2006). Retrieving and utilizing information from multiple diverse sources puts high demands on semantic integration solutions (Ameri et al. 2012; Borgo and Leitão 2007; Jardim-Goncalves et al. 2011; Leitão et al. 2016).

Formal engineering ontologies are emerging as popular solutions for addressing the semantic interoperability issue in heterogeneous distributed environments and for bridging the gap between the legacy systems and organizational boundaries (Ameri et al. 2012; Borgo and Leitão 2007; Strzelczak 2015). One of the most quoted definitions of ontology in the literature and in the ontology community is Gruber's (1993) "Ontology is a formal, explicit specification of a shared conceptualization". According to Studer et al. (1998), "'Conceptualization' refers to an abstract model of some phenomenon in the world by having identified the relevant concepts of that phenomenon". Ontology has classes representing things in reality, properties giving details to classes, and restrictions limiting and creating rules for the ontology. Ontology itself is just a formal definition, but when combined with instances of the ontology, it forms a knowledge base (Noy and McGuinness 2001). In the context of distributed intelligent systems, such as agent-based, holonic, or SOA- (Service Oriented Architecture) based systems, ontologies play a key role as they provide a shared, machine-understandable vocabulary for information exchange among dispersed actors (Leitão et al. 2016; Ameri et al. 2012).

To allow rapid system design and reconfiguration decision-making in heterogeneous multi-vendor production system environments, a common formal resource capability model is needed. Such information model should allow resource vendors to describe the functionality of their offerings in a comparable manner, and system designers to make a match between product requirements and resource capabilities. The contribution of this paper is twofold. Firstly, it presents the Manufacturing Resource Capability Ontology (MaRCO), developed to support the rapid semi-automatic system design, reconfiguration, and auto-configuration of production systems. Secondly, it describes the systematic development process of MaRCO model following the ontology engineering methodology. The objective is to bring forward the intentions and reasoning behind the model's development, and to give a solid example of the ontology development process. Compared to other existing approaches to describe resource functionalities, the main contribution of MaRCO lies in its ability to model and infer information about combined capabilities of aggregated resources on a parametric level.

The remainder of the paper is organized as follows. In "Related work" section, the related work on manufacturing ontologies and resource descriptions is briefly reviewed, and the need for a new manufacturing resource ontology is justified. "Ontology engineering methodology" section introduces the ontology engineering methodology followed in the development process, while "Manufacturing resource capability ontology development process" section discusses the results of applying the methodology, i.e. the development process and results of MaRCO. Last of all, the discussion and conclusions are presented.

Related work

The aim of bringing automation to system design, reconfiguration, and order dispatching requires a formal, structured representation of the product requirements, as well as of the resource capabilities, properties, and constraints. Several researchers have recently addressed the issue of manufacturing resource modeling using different methods, with different purposes and viewpoints, and with different levels of detail.

For the past two decades, in the manufacturing domain, there has been an increasing interest in using emerging technologies, such as ontologies, semantics, and semantic web technologies, to support collaboration, interoperability, and adaptation needs. One of the earliest manufacturing ontologies was the Process Specification Language (PSL) that was developed with the goal of providing a neutral language

for representing process-related knowledge and to support application integration. PSL used Knowledge Interchange Format (KIF) as its modelling formalism (Gruninger and Menzel 2003). MASON (Manufacturing's Semantics Ontology) shared the same goals with PSL, but was modelled with Web Ontology Language (OWL) (Lemaignan et al. 2006). During the EU FP6 project Pabadis'Promise a manufacturing ontology (P2 ontology) and reference architecture focusing on factory floor control was developed by Chapurlat et al. (2007). Borgo and Leitão (2007) developed the ADACOR ontology for distributed holon-based manufacturing, focusing on processes and system-interaction descriptions. It consists of an ontological classification of ADACOR concepts according to DOLCE foundational ontology. During the EU FP6 EUPASS project, an ontology for modeling evolvable, modular, ultra-precision assembly systems was developed by Lohse (2006). Kitamura et al. (2006) presented an ontological definition of an assembly device's capabilities based on the function-behavior-structure (FBS) framework.

An ontology-based capability management approach for multi-agent-based manufacturing systems was developed by Timm et al. (2006). In the SIARAS project, an intelligent system, called a "skill server", was built to support automatic and semi-automatic reconfiguration of production processes (Malec et al. 2007; Bengel 2007). Barata et al. (2008) presented a multi-agent-based control architecture for a shop floor system (CoBaSa) which supports fast re-engineering and plug-and-play capabilities based on skill descriptions. Frei et al. (2010) applied CoBaSa in Self-Organizing Evolvable Assembly Systems (SO-EAS). Obitko et al. (2010) proposed an ontology for agent-based manufacturing systems. Terkaj and Urgo (2012) developed an ontological Virtual Factory Data Model, which acts as a shared metalanguage providing a common definition of the data that are shared among different software tools along a factory process lifecycle. Salonen et al. (2011) created methods and models for semantically rich machine system design processes, which relied on the domain-specific ontology dedicated to capturing the design and lifecycle information of that particular machine system type.

The Manufacturing-as-a-Service paradigm has been adopted by many researchers who have produced different approaches to formally describe service requests and offerings. Manufacturing Service Description Language (MSDL) was developed as a formal domain ontology for representing the capabilities of manufacturing services, focusing on mechanical machining services (Ameri and Dutta 2006). Later on, it was extended to other application domains, such as metal casting (Ameri et al. 2012), and used for a matchmaking methodology which aims to connect buyers and sellers of manufacturing services in distributed digital manufacturing environments (Ameri and Patil 2012). Shin et al. (2013) enriched the MSDL further to comply better with the requirements of Manufacturing Service Capability (MSC) models. Hu et al. (2009) developed an ontology-based digital description of resource services for grid manufacturing. In the ManuCloud project, an XML-based manufacturing service description was developed to enable the Manufacturing-as-a-Service operation principle in production networks (Rauschecker and Stöhr 2012). In the SkillPro project, the AutomationML-based format was used to store and communicate the resource skill descriptions to facilitate the autonomous setup and execution of production tasks (Pfrommer et al. 2014). Backhaus and Reinhart (2017) presented a similar concept aiming to simplify the task-oriented programming of assembly systems by vendor-independent skill descriptions.

Most of the existing resource description approaches are domain-specific and offer only partial solutions for very specific applications and hence lack a comprehensive view. Some of them are very detailed, focusing on a specific process domain (e.g. machining), while others are more high level representations of the whole production domain, better suited to production planning and scheduling applications, rather than to capability matchmaking. Generally, the ontologies or other data models are not publicly available, making their re-use practically impossible. The previous research efforts to describe manufacturing resource capabilities are limited in that they either do not consider the combined capabilities of multiple co-operating resources, or they do not incorporate parameter information into the capability descriptions. Furthermore, most of the approaches rely on static resource descriptions that lack the lifecycle aspect. While a resource is being used, its condition, capabilities, and other important parameters may change. Thus, instead of only incorporating information about the nominal capability, it is also important to include information about the actual capability in the resource description in order to ensure that feasible design and reconfiguration decisions can be made. These above mentioned deficiencies are what is motivating the development of the MaRCO model presented in this work.

Ontology engineering methodology

The ontology engineering methodology deals with the process and methodological aspects of ontology engineering, i.e. by providing guidelines on how to develop ontologies. The widely used ontology engineering methodology, presented by Sure et al. (2009), consists of five phases: (1) Feasibility Study, (2) Kickoff, (3) Refinement, (4) Evaluation, (5) Application and Evolution. Each of these phases will be discussed in detail below and appended by other sources.

1. Feasibility study: In the feasibility study phase, the problem area, opportunities, and possible solutions are evaluated (Sure et al. 2009). Furthermore, the most promising focus areas, goals, and desirable solutions are selected. The feasibility study helps to view problems from a wider perspective and to determine economical and technical project feasibility (Staab et al. 2001).
2. Kickoff: The requirements for an ontology need to be identified before the actual ontology development starts. Kickoff is an extensive phase, which can basically be divided into two parts: (a) Detailed requirements definition, and (b) Semi-formal ontology definition. During the kickoff phase, the ontology requirement specification document should be generated, defining the goals, domain, and scope of the ontology, i.e. describing what the ontology should support, its area of application, and the valuable knowledge sources for generating the semi-formal ontology description (Sure et al. 2009; Staab et al. 2001). To determine the scope, competency questions can be used. Competency questions are simple questions representing queries made about the ontology. These questions can be used later to verify the ontology (Noy and McGuinness 2001). Re-use of pre-existing ontologies should be considered (Sure et al. 2009).

   Furthermore, during kickoff, a semi-formal description of the ontology is generated. This forms the baseline taxonomy for the ontology (Sure et al. 2009; Staab et al. 2001) and is often conducted in the ontology refinement phase when formally modeling the target ontology (Sugumaran and Storey 2002). The first phase in building an actual ontology is ontology capturing. This means identifying key concepts and relationships, producing textual definitions, and finally defining the terms (Uschold and Gruninger 1996). Noy and McGuinness (2001) describe the ontology creation process, which starts from the definition of the domain concepts. After the concepts are identified, they are placed in a taxonomical hierarchy. Then, the concepts' properties, including their relations, limitations, and rules are defined (Noy and McGuinness 2001). Use-cases may be used to bring the terminology together. The relations between the terms can be formed in three ways: through generalization, synonyms, or associations. Generalizations define the different levels of terminology and allow the subclassing of the defined terms. Synonyms are required for the incorporation of multiple views. Associations are the most important link as they mark a direct semantic connection between classes (Sugumaran and Storey 2002).
3. Refinement: In the refinement phase, the semi-formal description of the ontology is formalized, or coded, into the target ontology by using some ontology representation language. First, the taxonomy is formed out of the semi-formal description of the ontology, and then the relations between the concepts are defined (or just coded, if they were previously defined in the kickoff phase) (Sure et al. 2009). Re-using existing ontologies may provide speed and quality for the development in this phase as they can guide the development and give ideas (Staab et al. 2001). Capturing and codifying the ontology are often merged as a single step, or they are cyclical processes that follow each other (Uschold and Gruninger 1996).
4. Evaluation: After the ontology has been developed, it needs to be evaluated and verified by comparing it to the requirements. By instantiating the ontology, its structure and expressiveness can be tested. This means creating individual elements in the class-structure and completing their properties and relations (Noy and McGuinness 2001). For an ontology-focused evaluation, ontology-evaluation rules and approaches, such as OntoClean (Guarino and Welty 2009), may be used. An easy way to test and validate class hierarchies, especially transitivity, is to run "is-a" or "is-kind-of" tests against two classes in the hierarchy. This especially helps to detect flaws in deep hierarchies (Noy and McGuinness 2001). Testing the ontology in an application environment is recommended. The evaluation phase is strongly linked with the refinement phase, and together they form a cyclical process of ontology development in which the ontology will be improved in iterations (Staab et al. 2001).
5. Application and evolution: The final phase in the ontology development is its application and evolution. Ontology requires maintenance to fulfill the changing requirements during its use. Continuous feedback from both users and applications helps to improve the ontology (Staab et al. 2001). Even if the models built are static, the real world is changing continuously. Through this natural evolution, the definitions, terms, or constrains may vary, which affects the ontology needs as well. The changes may add, subtract, or split the ontologies or their contents in order to evolve new ontologies to match the changing reality (Sugumaran and Storey 2002). It is important to clarify who is responsible for the maintenance, as well as how and in which time intervals it is to be performed (Sure et al. 2009).

Manufacturing resource capability ontology development process

The Manufacturing Resource Capability Ontology (MaRCO) was developed following the ontology engineering methodology discussed in the previous section. In the following sub-sections, the development process and associated results are described.

Feasibility study

During the feasibility study, the scope and requirements of the ontology were defined. The need for the resource capability model was previously justified in the Introduction and Related Work section, and will not be repeated here. The scope of MaRCO is to formally model capabilities of manufacturing resources, initially concentrating solely on machine and tool, but not human, resources. The resource vendors should be able to utilize the capability model to represent the capabilities of their resource offerings in a formal, vendor-independent manner, and to publish these descriptions in local or global resource catalogues and digital marketplaces. System integrators and manufacturing companies can then browse these machine- and human-interpretable representations when designing new production systems from scratch or when reconfiguring the existing systems. Furthermore, the resource end users (i.e. manufacturing companies) should be able to maintain local knowledge bases of the existing in-house resources and append the nominal resource descriptions with updated capability information throughout the lifecycle of the individual resources. The aim is to allow automatic matchmaking methods to suggest suitable resources and resource combinations for certain product requirements from large search spaces (resource pools), facilitating semi-automatic system design and reconfiguration decision-making, and consequently faster adaptation to changing demands. The resource capability model should allow the definition of functional configurations of resources, but the detailed behavior of resources (e.g. movement trajectories) is not in the model's scope. Also in this phase, the decision was made to concentrate first on resource capabilities and to leave other aspects, such as resource interface descriptions, as a future work.

Kickoff

Kickoff was divided into two phases: the detailed requirement definition and the definition of the important concepts. The results of these phases, including the conceptual model of MaRCO, will be presented in the following sub-sections.

Definition of the detailed requirements

In this phase, the following issues were addressed: (a) the aim and intended use of the ontology, (b) the information that must be provided by the ontology, (c) users of the ontology, and (d) the questions which the ontology needs to be able to answer, i.e. the competency questions.

(A) Aim and use of the ontology

In order to define the aim and intended use of the ontology, a use-case approach was utilized. Two main use cases were identified: the reconfiguration and greenfield design scenarios. In the reconfiguration scenario, the model should support the comparison of the product requirements with the capabilities of the resource combinations in the current system layout. If no match is found for all product requirements, reconfiguration actions should be suggested. This implies that new resource combinations within the existing in-house resources and catalogue resources, that match with the product requirement, should be generated and suggested for the reconfiguration planner. In the greenfield design scenario, new resource combinations from the given catalogue resources should be generated that match with the product requirements.

Furthermore, some general requirements for the digital knowledge representation of the resources and their capabilities were defined:

1. The model must represent a selected domain, which is manufacturing and assembly resource capability modeling;
2. The model must be both human- and computer-interpretable, it must be formalized and have a meaning that can be understood and used by different users, applications, and systems;
3. Semantics need to be defined in such a way that the meaning of each structure in the knowledge representation is clear and there is no ambiguity in terminology;
4. The resource capability descriptions should be established easily and rapidly, and the descriptions should be easily re-used and extended;
5. The proposed knowledge representation must be suitable for reasoning, and it should support activities such as matching the resource capabilities against product requirements, searching for and selecting suitable resources, and proposing feasible system configurations.

(B) Information that must be provided by the ontology

Specific requirements for the MaRCO's content were defined as:

1. The model should allow the representation of the functions (capabilities) that a resource is able to perform and the parameter values within which the resource is able to perform these functions;
2. The capability model should define the generic capabilities that can be assigned to resources, including their name and related capability parameters;
3. The model should provide a process-oriented definition of capabilities which allows the matching of a product's

processing requirements with the resource capabilities (i.e. requirements and offerings);
4. The model must support the aggregation of the capabilities of multiple co-operating resources, i.e. combined capabilities;
5. The model should allow the description of capabilities with the lowest level of granularity at which the reconfiguration takes place. This enables the separation of resources and the treatment of system components as exchangeable entities which can be organized in a wide variety of different configurations. It also allows the reconfiguration decisions to be taken based on the capability model (e.g. separating the description of a drill from that of a drill bit);
6. The model should be flexible, in the sense that it allows also the capabilities to be described with different abstraction (granularity) levels, e.g. describing the drill + drill bit combination as a single resource with a single combined capability description;
7. The model should classify the process capabilities in a hierarchy, allowing subsumption-based reasoning of different levels of processes (e.g. riveting is classified as a joining process);
8. The model must take the lifecycle of the resources into account. Therefore, there needs to be a separate representation for each individual resource instance into which the lifecycle information can be collected and updated.

(C) Users

The users of the ontology were analyzed by dividing them into three categories:

1. *Contributing the information* The ontology is developed based on manufacturing engineering text books, on system and component provider datasheets and web pages, and on discussions with the manufacturing companies, both with the end users and with the resource providers. The main contributors of the instance information to the knowledge base will be the resource providers describing the capabilities of their offerings, i.e. the supplied machines, tools, and systems. Also, the end users of the manufacturing resources may update the capability information of specific individual resources during their lifecycle;
2. *Using the information* The main users of the information will be the system designers, system integrators, and end users who want to search for resources matching their specific processing requirements;
3. *Maintaining and modifying the model* Capability model evolution needs to be in the hands of some sort of harmonization or standardization organization. The descriptions need to follow common syntax and semantics to allow automatic capability matchmaking from global multi-vendor resource catalogues. This goal cannot be achieved without centralized maintenance of the ontology. For local usage, such centralized management is not required.

(D) Competency questions

Based on the requirement analysis and anticipated use cases, competence questions for MaRCO were formulated:

1. What capabilities does a specific resource have?
2. What simple capabilities are needed to form combined capability X?
3. What capabilities does a combination of specific resources have?
4. What are the parameter values of combined capability X when the required simple capabilities are combined?
5. What resources are involved in the current system configuration?
6. What resources form functional combinations in the current system configuration?
7. What capabilities exist in the current layout?
8. What capabilities satisfy a certain upper-level process requirement at the concept-name level, e.g. what capabilities can be used to perform a "material removing" process?
9. What resources in the current layout have capabilities that match with a product requirement on a capability-concept-name level?
10. Which of these resources match with the product requirements on the capability-parameter level? (Continuation from Q9)
11. Which resources in a certain search space have capabilities that contribute to combined capability X?
12. With which resources could resource A be combined in order to achieve combined capability X?

Determining the essential concepts and their relationships

This step was the key phase in the development of MaRCO. It was responsible for defining the important concepts and attributes of MaRCO and analyzing their relationships. The end result of this activity was the MaRCO conceptual model. However, the detailed taxonomy, and the properties and relationships definitions were drawn up in the refinement phase. Thus, the concepts of the developed conceptual model do not directly translate into the classes of the coded ontology, but the detailed ontology capturing was carried out in parallel with ontology codification.

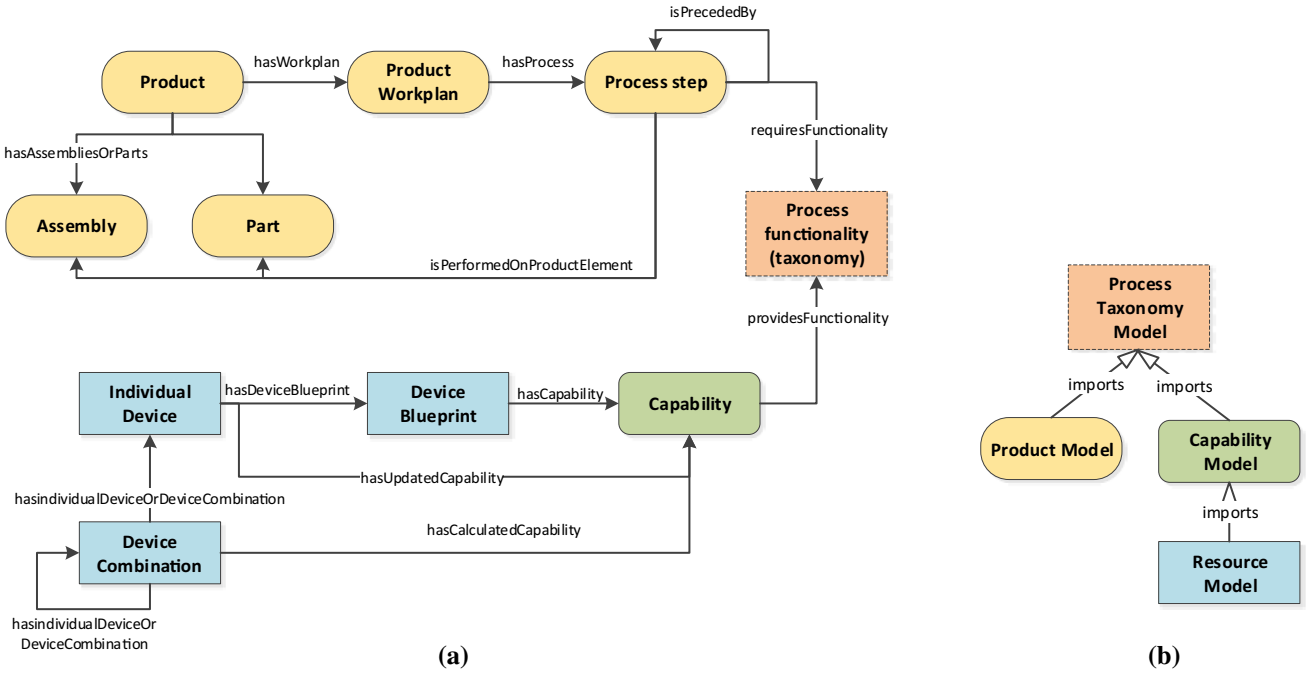

Fig. 1 **a** Simplified conceptual model of the relations between product requirements, resources, and capabilities; **b** distributed capability-related ontologies

The purpose of the ontology is to support automatic matchmaking between product requirements and resource capabilities. Thus, the essential concepts and their mutual relations in the areas of Product, Process, Capability, Resource, and System were first identified. Figure 1a gives a conceptual description of how the product, resource, and capability concepts are linked together in matchmaking. For greater simplicity and readability, only the most relevant identified relations are shown here. The term "Device" is used to indicate that the work concentrates only on machine and tool resources, and not on other resource types, such as human operators or raw materials.

During the analysis, it was identified that four distributed ontologies are needed to facilitate the capability matchmaking. Figure 1b illustrates these distributed ontologies. The shape and color of the elements correspond with the concepts in the conceptual model presented in Fig. 1a. These ontologies have different users and different usage phases during the product and production system lifecycles. Instead of having one large ontology, the distribution reduces the complexity of the model from the user's point of view. The Process Taxonomy Model defines the hierarchical categorization of different manufacturing processes, e.g. "milling" is classified as "machining" and further as a "material removing" process. The Product Model is used to model the product characteristics and manufacturing requirements. The Capability Model defines the capability names, parameters, and relations between simple and combined capabilities. The Resource Model defines the resources and the systems composed of the resources. This paper focuses on the Resource Model which imports the Capability Model, thus referred to here as the Manufacturing Resource Capability Ontology (MaRCO).

The conceptual model of capabilities was initially presented in (Järvenpää et al. 2011, 2016; Järvenpää 2012). The main concepts of the capability model are presented in Fig. 2. The capabilities are described by name and parameters. The capability concept name indicates the natural name of the capability, such as "Moving", "Drilling", "Screwing", and "Grasping". Capability parameters describe the characteristics of a capability, e.g. the "Moving" capability is characterized by "speed" and "acceleration" parameters, among others. The capability parameters help to distinguish between different resources with similar capabilities. In other words, the concept name of the capability indicates the operational functionality of the resource, whereas the capability parameters determine the range and constraints of that functionality.

The Capability Model divides the capabilities into simple and combined capabilities. Combined capabilities are upper-level capabilities which can be divided by functional decomposition into simple, lower-level capabilities ("part-of" hierarchy). Combined capabilities are combinations of two or more (simple or combined) capabilities. In the Capability Model, the simple and combined capabilities are linked by *hasInputCapability* relations (Fig. 3). All input capabili-

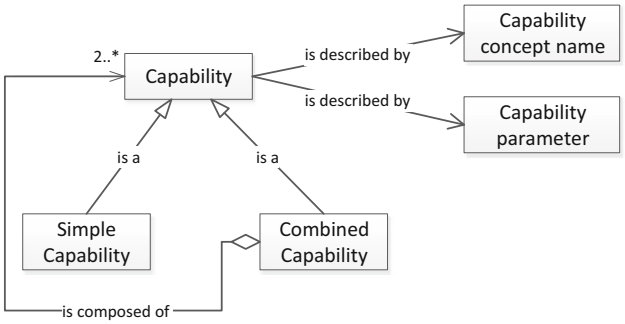

Fig. 2 Concepts of the capability model

ties must be available before a combined capability can exist in a system (logical AND). For example, in order to transport an item the system needs to be able to move within some workspace and to grasp the item or to hold it by gravity. Several different capabilities may provide alternative input for certain combined capability. In the previous case, "Grasping" and "HoldingByGravity" are alternative inputs for "Transporting" (logical OR). As a result, both robot with an attached gripper, and conveyor belt alone provide the "Transporting" capability. Similarly the same input capabilities may be required by multiple different combined capabilities. As an example "SpinningTool" capability is part of both "Screwing" and "Drilling" combined capabilities, as illustrated in Fig. 3.

Capability parameters describe the characteristics of a capability. In the Capability Model, the capability parameters have been defined based on the most common parameters given in the tool and machine providers' catalogues. The Capability Model aims to model the parameters solely from the resource perspective, which means that the resource provider should be able to define those parameters without having any knowledge of the product for which it may be used. Figure 4 shows examples of the capability parameters in the cases of "Moving" and "FingerGrasping" capabilities.

The capabilities form the capability catalogue, which consists of the pool of generic capabilities that may exist in a production system. These capabilities can be assigned to resources in the Resource Model, whereupon the capability parameters are filled with the resource-specific parameter values. Figure 5 illustrates the important concepts in the Resource Model. The Device Blueprint describes the capabilities, interfaces, and properties of one type and model of device, as given in the vendor's catalogue. This represents the nominal capability of the device. The individual devices are presented as a separate concept which, while referring to the Device Blueprint, presents the actual capabilities of the particular, individual resource on the factory floor. The individual devices have actual capabilities, which are affected by the lifecycle of each individual device and updated according to the measured or calibrated values on the factory floor. For example, if the measured accuracy of the machine differs from the value defined in the nominal capability (e.g. because of wear), this updated value can be given in the actual capability definition. The device combinations are combinations of multiple individual devices. These combinations can then have combined capabilities.

Based on the defined relations between capabilities in the Capability Model, the resource combinations contributing to a certain combined capability on a concept name level can be identified. In order to calculate the parameter values for the combined capabilities of aggregated devices, combined capability rules are needed. Such rules are used, for instance, to automatically calculate the "payload" parameter of the "Transporting" capability. "Transporting" is a combined capability belonging to resources or resource combinations which can transport items from one place to another. For instance, a conveyor alone or a robot combined with a gripper have this capability. In order to automatically define the "payload" parameter of the "Transporting" capability of a robot and finger gripper combination, the following rule can be applied (Järvenpää et al. 2017): Transporting payload = "payload" property of "Moving" capability minus

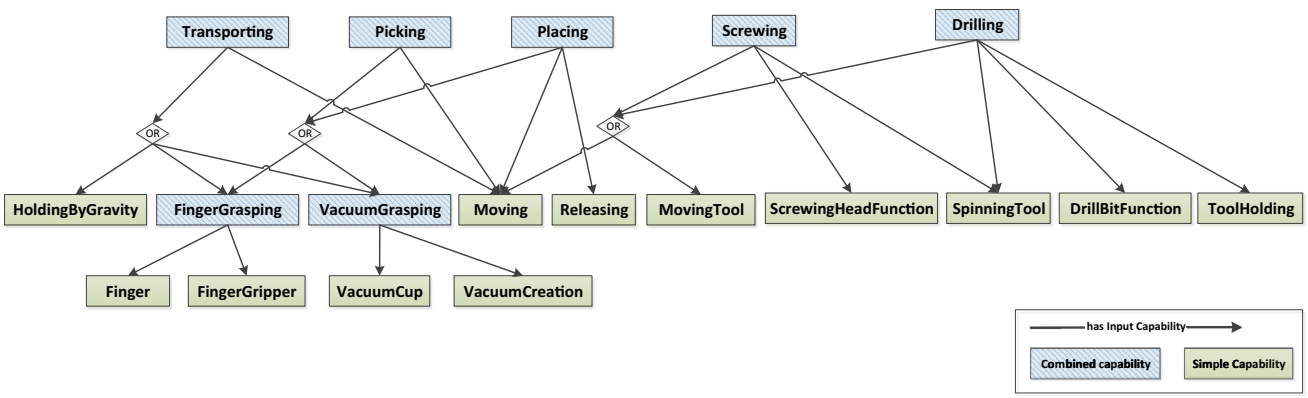

Fig. 3 Example capabilities and relations between simple and combined capabilities

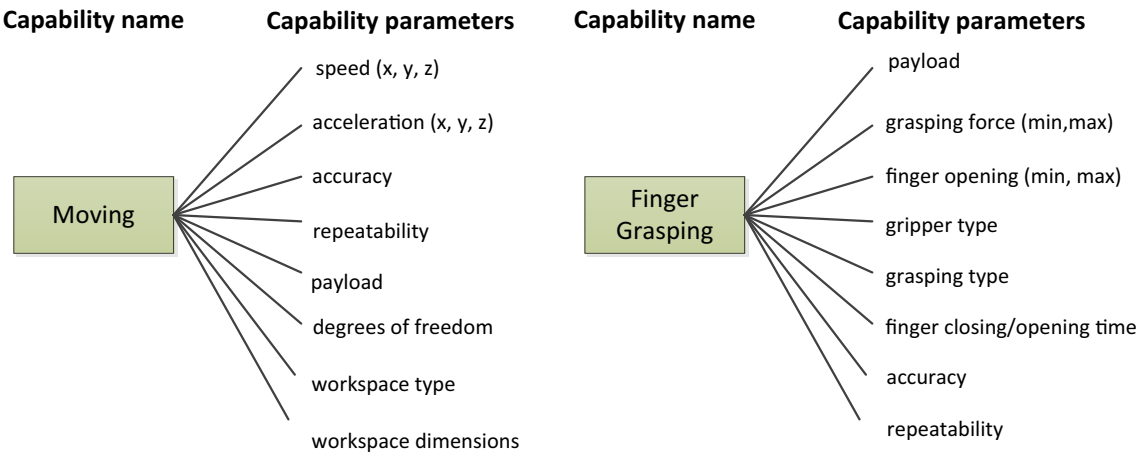

Fig. 4 Examples of capability parameters

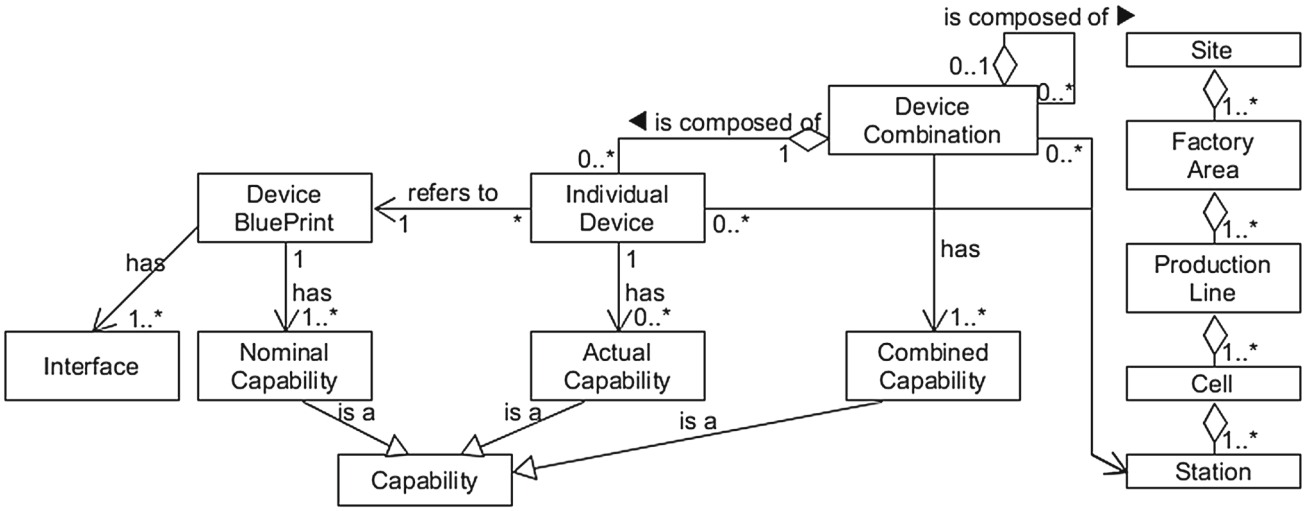

Fig. 5 Conceptual relations between capabilities and resources

"mass" of the gripper, OR "payload" property of "Finger-Grasping" capability. The smaller value is dominating.

Refinement: representation of the ontology

In the refinement phase, the ontology is formalized and represented. Ontology representation language plays a key role in the establishment of an ontology. The following requirements were identified for language representing MaRCO: (1) be able to describe the manufacturing capability of a resource in a uniform way; (2) provide a powerful representation and reasoning ability for automatic capability discovery, intelligent search, and resource-matching capabilities; (3) have a powerful support of interior logic system; (4) be compatible with the existing and established semantic web standards; and (5) be extensible. The Web Ontology Language (OWL) was selected as it is the most commonly used ontology language and satisfies the previously mentioned requirements. OWL is a semantic mark-up language for publishing and sharing ontologies on the World Wide Web and is developed as a vocabulary extension of RDF (Resource Description Framework) (W3C 2004). Protégé ontology editor (Protégé 2015) was used to construct the ontology.

Figure 6 and Table 1 introduce the main classes of the developed Capability Model ontology and their most important properties. Due to space limitations and for greater readability of the figures, some classes and relations are omitted from the presentation. For instance, Fig. 6 only illustrates two capability classes, namely Moving and Transporting, as examples. Those classes, which have more siblings than illustrated by the figure, are marked by boxes with dashed lines. The same applies to Fig. 8.

The Resource Model ontology imports the Capability Model. Together, they form the Manufacturing Resource Capability Ontology (MaRCO). The formalization of the Resource Model quite closely follows the semi-formal

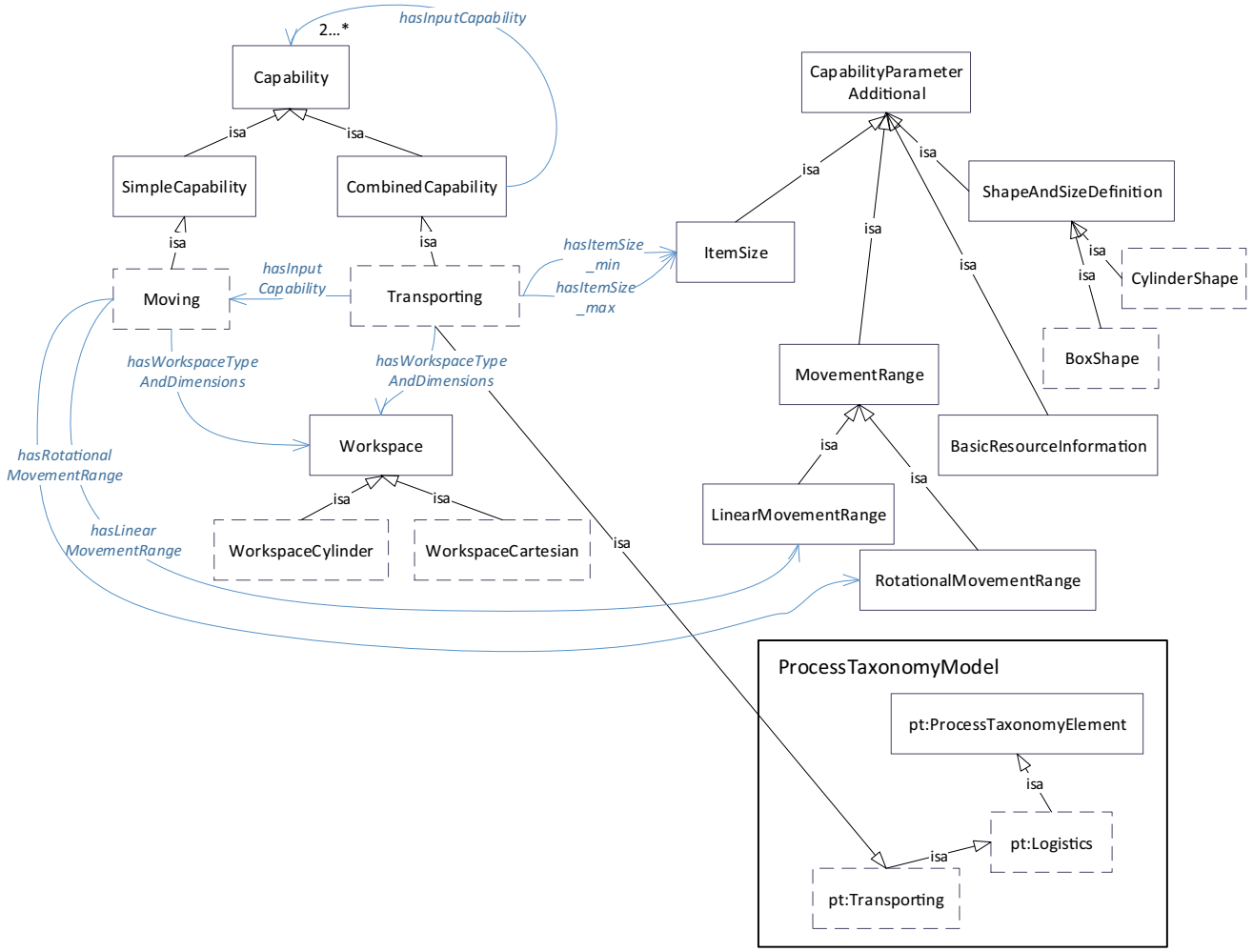

Fig. 6 Main classes and relations of capability model ontology

ontology definition presented in "Determining the essential concepts and their relationships" section. Table 2 gives a brief description of the most relevant classes in the Resource Model. In addition, Fig. 8 presents the main classes and their main relations. The focus is especially on the class descriptions for DeviceBlueprint, IndividualDevice and DeviceCombination. These classes are at the very core of the developed model as they link the capabilities to the devices, both the catalogue devices and the actual devices on the factory floor. Moreover, they allow the modeling of the device combinations from which the combined capabilities can emerge.

Evaluation

Before being used, the ontology should be evaluated and tested. The class hierarchy, i.e. the taxonomic structure of MaRCO, was validated by utilizing "is-a" and "is-kind-of" tests, as suggested by Noy and McGuinness (2001). Currently MaRCO is not intended to be used for automatic classification of instances. Thus, a more systematic analysis of the ontology structure, e.g. by OntoClean (Guarino and Welty 2009) or a similar approach, was not deemed necessary. If classification is required in the future, it will be crucial to utilize more systematic evaluation methods to eliminate incorrect classification results. Several example resources were modeled with MaRCO to test its expressiveness and ability to answer the defined competency questions. In the following sub-sections, a small case example will be outlined and the observations made during the testing will be analyzed.

Test cases

Figure 9 represents the demonstration setup in an industrial laboratory environment, which was used to perform the initial tests with the MaRCO model. It consists of a 2-axis manipulator, a finger gripper, a feeder unit with exchangeable

Table 1 Main classes of the Capability Model

Class	Description
Capability	This is a parent class for the specific capability classes, which define the functionalities of resources. It has two sub-classes, namely **SimpleCapability** and **CombinedCapability**, which are further divided into sub-classes formalizing the description of specific capabilities. These sub-classes include all simple and combined capabilities currently modeled into the Capability Catalogue. The capabilities are related to each other through the *hasInputCapability* object property. Capability parameters are implemented as datatype and object properties and are linked to the capabilities by property restrictions. Property restrictions are used to restrict the properties of instances belonging to a certain class. An example of capability definition can be seen in Fig. 7, which represents the class description of the "Transporting" capability. Most of the capability parameters are implemented as datatype properties, which can be used to save literal data values. Some parameters are implemented as object properties, which are used to model relationships between instances. These instances belong to the CapabilityParameterAdditional class, discussed below.
CapabilityParameterAdditional	This is a parent class for storing capability parameter groups, which cannot feasibly be stored as individual datatype properties, but rather as instances linked through object properties. This approach is used in two cases. Firstly, it is used for capability parameters which form a natural group of conceptually linked parameters and such group (parameter set) can be re-used to describe multiple different capabilities (example class: ItemSize). Secondly, it is used with capability parameters, which relate directly to certain capability, but whose properties depend on the nature of the capability (example class: Workspace). Examples of the sub-classes and their purposes are given below.
Workspace	The workspace type of the movement capability determines the parameters of the workspace and thus cannot directly be modeled as datatype properties of the "Moving" capability. This class is used to save the details of the workspace of the "Moving" capability through the *hasWorkspace* object property. The sub-classes: WorkspaceArticulated, WorkspaceCartesian, WorkspacePolar define the dimensional datatype properties of the specific workspace type.
MovementRange	This class is used to store the dimensional information of movement ranges. Linear and rotational movement ranges are saved in separate sub-classes. Instances of "Moving" capability are related to the instances of this class by the *hasLinearMovementRange* and *hasRotationalMovementRange* object properties.
ItemSize	This class is used to store the dimensional and mass information of items. It can be used to define the maximum and minimum item size that can be handled with a certain capability on a coarse level. Each capability, of which the item size is critical, is related to instance(s) of this class by *hasItemSize_max* and *hasItemSize_min* object properties.
ShapeAndSizeDefinition	This class includes the different shape and size definitions that can be used to describe e.g. the shape of a mold or adapter on a coarse level, or the shape and size of an item that can be handled. The sub-classes: ConeShape, BoxShape, CylinderShape, PyramidShape, and SphereShape define the dimensional properties of different shapes. This can be used to describe capabilities for which the shape is traditionally important, e.g. the tray of a tray feeder, or the tube in a tube feeder, and is related through the *hasAllowedShapeAndSize_min*, *hasAllowedShapeAndSize_max* and *hasAllowedShapeAndSize_exact* object properties.
BasicResourceInformation	This class is used for storing a collection of basic information about a resource, such as its physical dimensions and mass. This information is not logically related to any specific capability, but is needed for combined capability calculation and capability matchmaking. Each resource modeled by the Resource Model is related to one instance of this class by the *hasBasicResourceInformation* object property (see Figure 8).
ProcessTaxonomy (imported)	The process taxonomy defines the hierarchical classification of different process functionalities. It is a pure taxonomy, without any properties. The taxonomy can be used to link the capabilities to the upper levels in the process hierarchy, e.g. "Drilling" capability would be classified as a "Material removing" process in the process taxonomy. The Capability Model imports the Process Taxonomy, and the capability classes are defined as the sub-classes of the relevant classes in the process taxonomy hierarchy. The reasoning ability of OWL allows direct inferences to be made that each instance saved to a certain capability class is also an instance of the process taxonomy class, which was defined as the parent class of the specific capability.

magazine, a conveyor belt, and exchangeable positioning units at both ends of the conveyor belt. Figure 10 presents the capabilities of the resource combination formed by the manipulator and the finger gripper. The simple capabilities, shown in the middle of the figure, are manually asserted to the Resource Model instances ontology. The right side of the figure shows the expected combined capabilities that should be automatically calculated and added to the model, on the basis of the relations between the capabilities and the combined capability rules.

Figure 11 shows an extraction of the information modeled about the demonstration setting with MaRCO. It is a graph

SubClass Of

- acceleration_x_max max 1 rdfs:Literal
- acceleration_y_max max 1 rdfs:Literal
- acceleration_z_max max 1 rdfs:Literal
- accuracy exactly 1 rdfs:Literal
- CombinedCapability
- dof max 6 rdfs:Literal
- dof min 1 rdfs:Literal
- hasAllowedItemShapeAndSize_max min 0 owl:Thing
- hasAllowedItemShapeAndSize_min min 0 owl:Thing
- hasInputCapability some (FingerGrasping or HoldingByGravity or VacuumGrasping)
- hasInputCapability some Moving
- hasItemSize_max exactly 1 owl:Thing
- hasItemSize_min exactly 1 owl:Thing
- hasWorkspaceTypeAndDimensions exactly 1 owl:Thing
- hasWorkspaceTypeAndDimensions only Workspace
- payload exactly 1 rdfs:Literal
- repeatability exactly 1 rdfs:Literal
- speed_x_max max 1 rdfs:Literal
- speed_y_max max 1 rdfs:Literal
- speed_z_max max 1 rdfs:Literal
- Transporting

Fig. 7 Property restrictions defined for the "Transporting" capability. A screenshot from Protégé

generated by OntoViz (Sintec 2007) directly from the Protégé ontology editor, and it visualizes the modeled instances and their relations. Due to the readability, only the robot and gripper combination from the demonstration station is included into the visualization. The figure shows the devices associated to that combination and their capability definition. All the other resources and their capability instances were modeled in a similar fashion.

Pure OWL (W3C 2004) does not provide solutions for making the required inference about the combined capability parameters, and assertions of new instances and their property values. Therefore, the OWL-based ontology needed to be enriched with semantic rules. SPARQL (SPARQL Protocol and RDF Query Language) (W3C 2008) queries were written and run to test the ontology against the competency questions. As an example, Fig. 12 shows a SPARQL query which searches for potential device combinations forming "Transporting" capability with "FingerGrasping" and "Moving" capabilities and calculates the "payload" parameter value for the combined capability. The query contains the combined capability rule for calculating the value of the "payload". The results it finds are shown at the bottom. Each row represents a possible resource combination with the involved resources and the calculated value for the "payload" parameter.

For the competency question: "What are the parameters of combined capability X when the required simple capabilities are combined?", it is possible to use a sequence of SPARQL queries to calculate each parameter value separately, as exemplified in Fig. 12. Another option is to use semantic rule

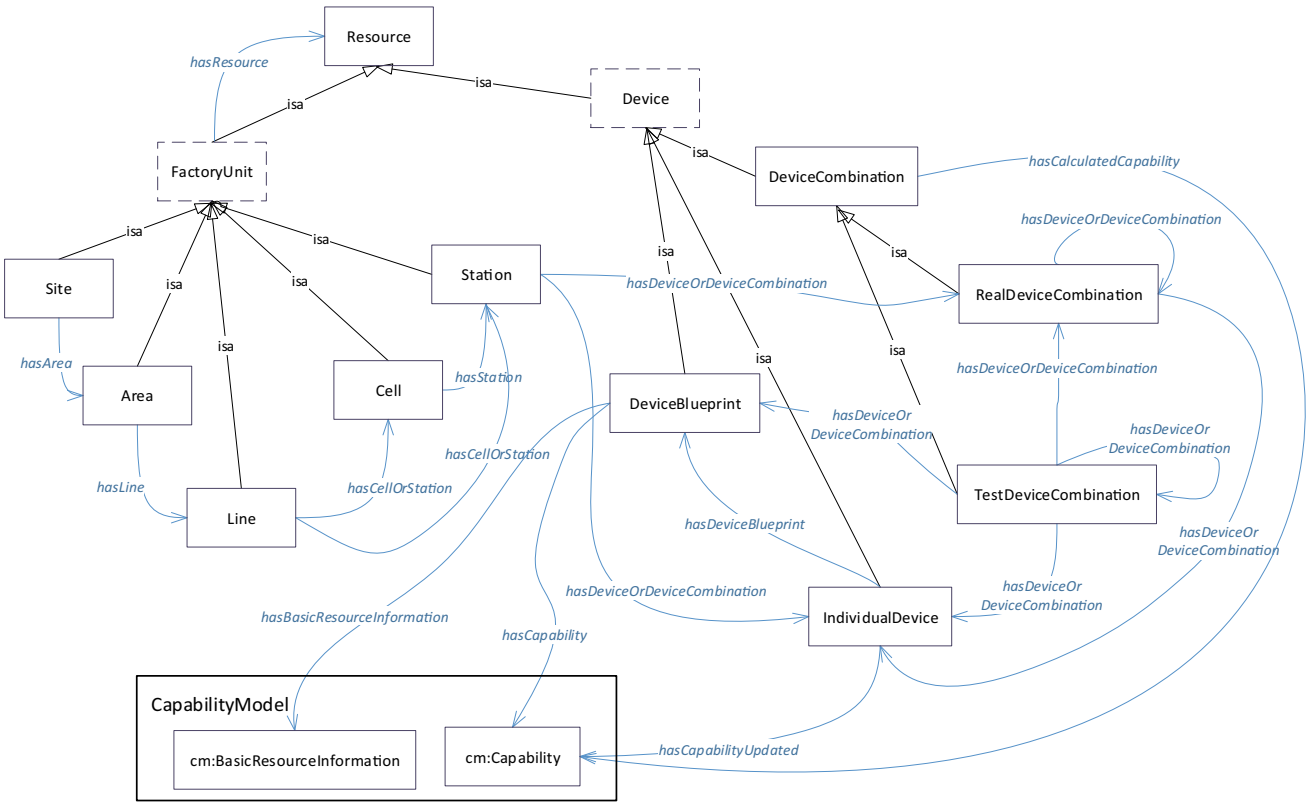

Fig. 8 Main classes and relations of the resource model ontology

Table 2 Main classes of the Resource Model

Class	Description
Resource	This is a parent class for all resource-related classes. Direct sub-classes are Device, FactoryUnit, Human, Raw Material and Software. In this article, only the Device and FactoryUnit classes are discussed.
Device	This is a parent class for different device-related classes. It includes any machines, equipment, and tools used to manufacture or assemble a product. Devices can be either catalogue devices (blueprints), actual individual devices, or device combinations.
DeviceBlueprint	This class contains the catalogue information about the devices. This class has several subclasses for different categories of devices. The instances of DeviceBlueprint relate to specific capability instances by the *hasCapability* object property. These capability instances are filled with resource-specific capability parameter values, making the capability description unique for each instance of the DeviceBlueprint class.
IndividualDevice	The instances of this class represent the actual individual devices present on the factory floor or in the warehouse. This class stores the life cycle information of the devices, and refines and updates the capability properties. The individual devices relate to the DeviceBlueprint through the *hasDeviceBlueprint* object property. Through this association, the nominal parameter information for this individual device gets defined. The updated capabilities are modeled through the *hasCapabilityUpdated* object property.
DeviceCombination	This class includes instances which represent combinations of multiple devices. It has two sub-classes: **TestDeviceCombination** and **RealDeviceCombination**. The former may consist of or include catalogue devices, which are not physically present (i.e. DeviceBlueprints). This class is used during capability matchmaking, when searching for alternative resource combinations in the catalogues, i.e. when generating temporal device combinations for tryouts and further investigations. RealDeviceCombination includes only real physical devices (i.e. IndividualDevices). Relations between the DeviceCombination instances and the instances of IndividualDevice or DeviceBlueprint classes are modeled through the *hasDevices Or DeviceCombinations* object property. The DeviceCombination instance may also relate to other DeviceCombination instance through the same object property. Furthermore, for the DeviceCombination instance, combined capability information can be saved through the *hasCalculatedCapability* object property.
FactoryUnit	The FactoryUnit class refers to the physical place in which the operations take place. It is a parent class for other factory-area-related classes, such as Site, Line, Cell, and Station. FactoryUnit relates to resources through the *hasResource* object property. Station relates to the instances of IndividualDevice or RealDeviceCombination classes through the *hasDevice Or DeviceCombination* object property. Furthermore, the relations between the different stations on the factory floor, and thus the coarse layout, can be described by the *adjacentTo*, *connectedTo* and *precededBy* object properties.

languages, which allow rules to be attached directly to the ontology, thereby increasing the reasoning abilities of pure OWL ontologies.

First, Semantic Web Rule Language (SWRL) (Horrocks et al. 2004) was tested for rule implementation. SWRL allows a reasoner, such as Pellet (Sirin et al. 2007), to infer the results automatically within the ontology. However, based on the tests and evaluations, SWRL has several limitations for this application. It cannot create and save new instances, and its performance in terms of speed and memory consumption was poor in the tests. Thus, the current implementation utilizes SPIN (SPARQL Inferencing Notation) for the rules. It is W3C Member Submission that has become the de-facto industry standard to represent SPARQL rules and constraints on Semantic Web models (SPIN working group 2017). It allows rules written with SPARQL to be attached to the classes of the ontology. A suitable reasoner tool, such as SPIN API (Knublauch 2016), can then infer the extra information created by the rules and use it during SPARQL query execution. The main advantage of SPIN is its meta-modeling capabilities that allow users to define their own reusable SPARQL query templates and functions (SPIN working group 2017). Based on the performed tests, SPIN performs better for this purpose and does not have the shortcomings of SWRL.

In addition to the query interface provided by Protégé, an external Java application and library, called Capability Query Library (CQL), was developed for querying the ontology and showing the query results in a command line based user interface. This API library can be used by other software tools to send queries to the MaRCO knowledge base. This library also handles the execution of the SPIN rules for calculating the combined capability parameters and for asserting this new inferred information to a new ontology file.

Apart from the initial test case described above, the MaRCO model has been used to model a wide variety of different resource instances. These resources have been collected mainly from two assembly scenarios from automotive and aeronautics industry. The modelled resources include: feeding devices, conveyors, fixtures, grippers, robots, manipulators, screw drivers and wrenches, presses, milling machines, as well as associated tooling. Over 30 different resource descriptions have been created. These resource instances have been used to test and validate that feasible

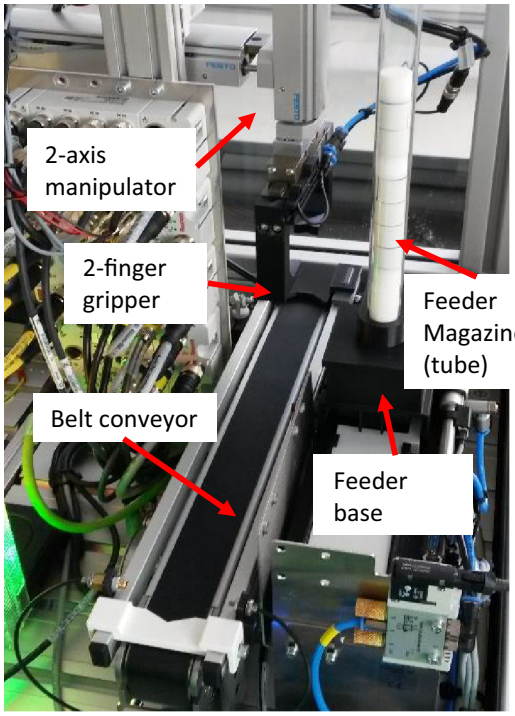

Fig. 9 Laboratory demonstration environment

resource combinations for a certain capability requirement can be generated and combined capabilities can be calculated. Also several production lines and stations have been modelled by assigning devices and device combinations to the stations and stations to the lines. Furthermore, several SPIN rules have been implement to infer the combined capability parameters. Their implementation is discussed in Järvenpää et al. (2018a). MaRCO has also been tested in the capability matchmaking context, i.e. matching the capability requirements of a product with the capabilities of the resources. This matchmaking relies also on the SPIN rules, which are used to compare the product information against the resource descriptions to find feasible resource combinations. This matchmaking process and some example rules, in the context of screwing process, are discussed in Järvenpää et al. (2018b).

Based on the performed tests, all the competency questions can be answered. Some of them require multiple SPARQL queries to be run in sequence, and this is programmatically handled by the developed CQL API. Table 3 illustrates how the information needed to answer the competency questions can be retrieved from the model.

Analysis

MaRCO was developed following tightly the requirements defined in the Kickoff phase and listed in "Definition of the detailed requirements" section. It satisfies all the information requirements set for it. From the general requirements' perspective, more final end user tests are needed before evaluating the ease of creating the resource capability descriptions. However, throughout the development, specific emphasis was put to the unambiguity, human-readability and clarity of the class and property names. All the capability classes and properties were systematically described by the annota-

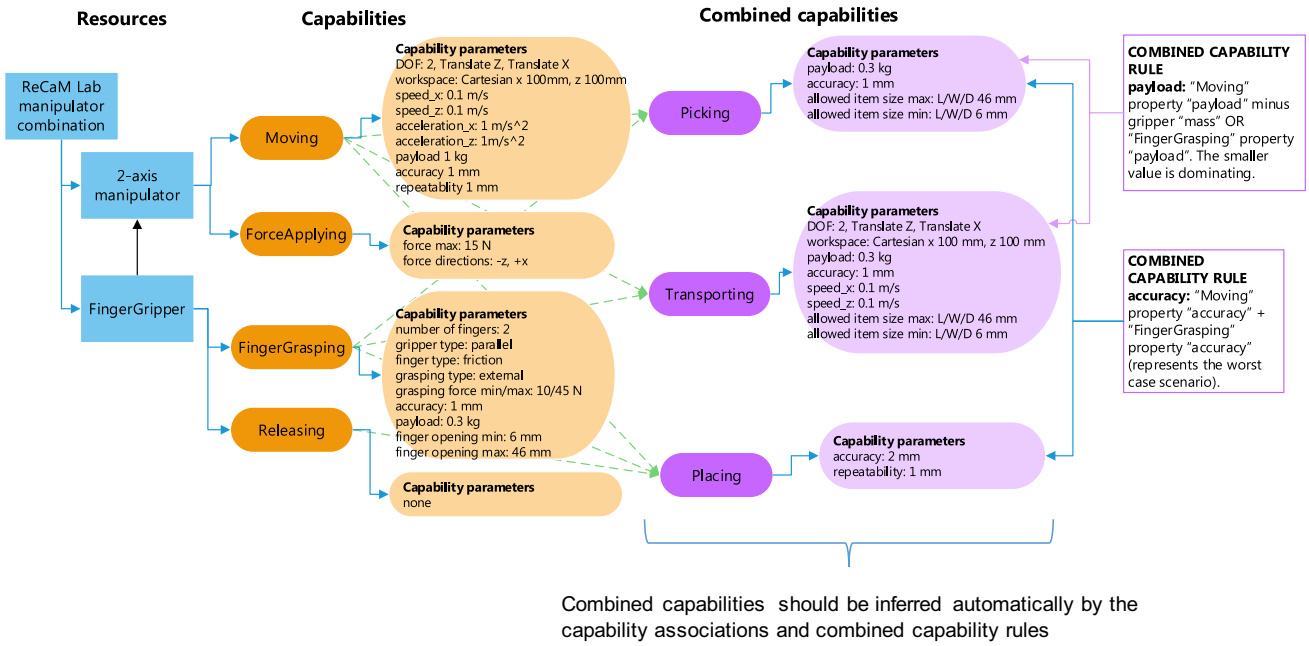

Fig. 10 Defined capabilities for the 2-axis manipulator and the finger gripper, and inferred combined capabilities

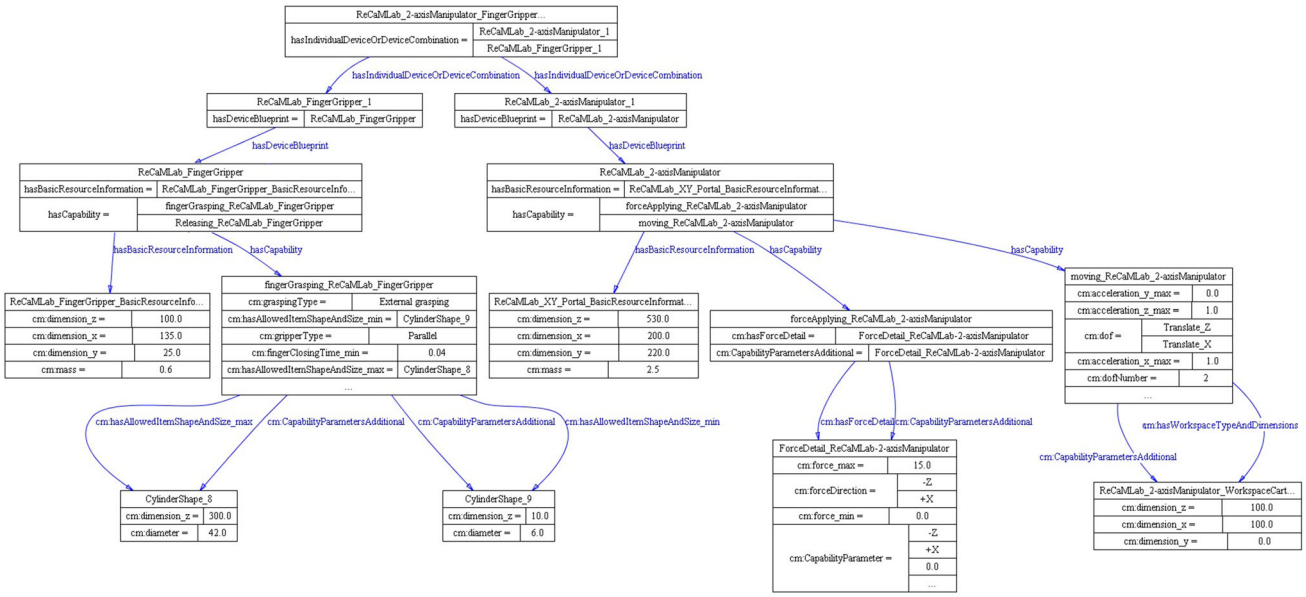

Fig. 11 Extracted instance data of MaRCO, representing manipulator and finger gripper resources on the demonstration environment

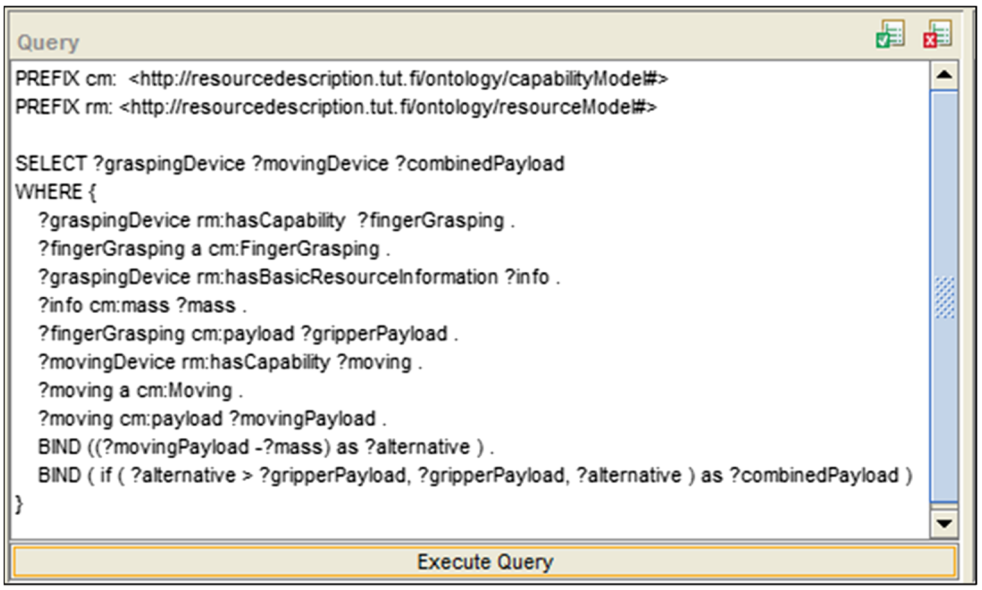

Fig. 12 SPARQL query and its results. A screenshot from Protégé

tion property rdfs:comment. This is expected to support the usability of the model.

Based on the tests performed with the demonstration data, MaRCO can provide answers to the competency questions defined in "Definition of the detailed requirements" section. Thus, in that scope, it satisfies the set requirements. However, in order to answer the question of whether two resources can actually be physically combined, a comprehensive interface

Table 3 How to retrieve the information required by the competency questions?

Competency question	How the information is retrieved?	Single/multiple SPARQL	CQL needed	SPIN used
Q1. What capabilities does a specific resource have?	Query directly from the resource through *hasCapability* property	Single	No	No
Q2. What simple capabilities are needed to form combined capability X?	Query the super classes of the combined capability X that define restrictions on the *hasInputCapability* property. For each restriction one of the capabilities defined through *someValuesFrom* property is required for the combined capability X	Single	No	No
Q3. What capabilities does a combination of specific resources have?	Query the required capabilities for each combined capability (Q2), query the capabilities of the combination's devices (Q1) and determine the combination's capabilities programmatically based on the retrieved information	Multiple	Yes	No
Q4. What are the parameter values of combined capability X when the required simple capabilities are combined?	Parameters are calculated with SPIN rules attached to the Capability class X, and values retrieved with a SPARQL query	Multiple	Yes	Yes
Q5. What resources are involved in the current system configuration?	Query resources, which link to FactoryElement through *hasResource* and *hasIndividualResourcesOrDeviceCombinations* object properties	Single	No	No
Q6. What resources form functional combinations in the current system configuration?	Query resources, which link to DeviceCombination through *hasIndividualResourcesOrDeviceCombinations* object property	Single	No	No
Q7. What capabilities exist in the current layout?	Use process for Q3 for each device combination in the layout, starting from the lower level ones	Multiple	Yes	No
Q8. What capabilities satisfy a certain upper-level process requirement at the concept-name level, e.g. what capabilities can be used to perform a "material removing" process?	Query the sub-classes of the given ProcessTaxonomyElement class (e.g. MaterialRemoving) that are also sub-classes of the Capability class, and retrieve the instances of these Capability classes	Single	No	No
Q9. What resources in the current layout have capabilities that match with a product requirement on a capability-concept-name level?	Use a SPIN rule to connect capability instances that match to the requirement on the name level (Q8), and query the resources, which connect to this capability through *hasCapability* object property	Single	Yes	Yes
Q10. Which of these resources match with the product requirements on the capability-parameter level? (Continuation from Q9)	Use process for Q4 to calculate the parameter values. Then use ProcessTaxonomyElement specific SPIN rules to connect those capabilities that match on parameter level. Query the results with SPARQL	Multiple	Yes	Yes
Q11. Which resources in a certain search space have capabilities that contribute to combined capability X?	Query capabilities required for combined capability X (Q2), query their instances, query resources associated with these instances	Single	No	No
Q12. With which resources could resource A be combined in order to achieve combined capability X?	Same as for Q11, except there is no need for search the capabilities already offered by resource A. (This is purely capability-centric view, not considering interfaces)	Single	No	No

description incorporating aspects of mechanics, control, and energy has to be integrated into the Resource Model. Until now, it has been beyond the scope of MaRCO's development. However, it will be an important part of future work so as to allow the generation of feasible resource combinations with compatible interfaces. Previously, the authors have proposed a comprehensive XML-based Resource Description Concept (Siltala et al. 2016), which includes a detailed formalism to describe the resource information including the interfaces. This work is currently being reused and the relevant parts of the XML-based interface definition model are codified into OWL-format and integrated into MaRCO (Siltala et al. 2018).

Currently, the tradeoff between generality and resource specificity in the capability classes is under consideration. One of the initial aims was to minimize the amount of different capability classes in the Capability Model. Thus, developers targeted general atomic capabilities which can be assigned to different kinds of resources (e.g. describing the moving capability of a milling machine and robot with the same generic capability). However, such generality in the capability definitions leads to a massive amount of practically impossible device combinations when searching for possible combinations with certain combined capabilities. Therefore, these need to be filtered out of the search results somehow, e.g. based on their interface definitions. Another extreme would be to define the specific capabilities for each different technological solution. This would, on the other hand, vastly increase the size of the capability catalogue and make it rather difficult for resource vendors to know the proper capabilities that they should assign to a certain resource. Thus, a more general approach would be preferable, even though it requires a detailed interface description. However, MaRCO supports both approaches, which means that when new capabilities (which are not connected through *hasInputCapability* property to pre-existing capabilities) are included in the catalogue, the level of generality can be evaluated and decided independently.

Application and evolution

As exemplified in the previous section, the researchers have successfully applied the MaRCO model to describe various manufacturing resources using the Protégé ontology editor and the Capability Query Library. The information has been collected from the resource providers or owners in spreadsheet templates and then codified into OWL by the researchers. In order to support a wider application of the model in real industrial scenarios, an "easy-to-use" user interface must be provided for resource vendors to create resource descriptions of their offerings. Such a user interface is currently in the development phase. It is expected that user feedback will result in extensions to the current model.

The MaRCO model is openly published and maintained in TUT server,[1] under the Creative Commons Attribution-ShareAlike 4.0 International License (CC BY-SA 4.0), to allow free usage and further development for anyone interested (here also permanent link to the present version[2]). However, when considering the intended usage of the model, i.e. the automatic matchmaking of product requirements against the resource capabilities in multi-vendor resource catalogue platforms, the centralized maintenance of the model becomes an essential aspect. In order to remain usable and compatible in a multi-vendor environment, there needs to be one central player taking care of the model evolution. In the initial phase, it will be the original developers, but at some point this activity should be taken over by some standardization organization.

The resource vendors may suggest new capabilities for the ontology, but the standardization organization should be in a position to decide how the capability is integrated into the overall model, how it affects to other capabilities, and what needs to be modified. This is regarded as necessary for keeping the model consistent and preventing similar concepts from appearing multiple times in the ontology. Furthermore, as the intended usage of the ontology relies heavily on rule-based reasoning, the usability of the general rules will be violated if there is no ontology manager to integrate new capabilities and modifications into the ontology and to develop and update the related rules. However, a certain actor, e.g. a single company or company network, may adopt the capability model for its own internal use, and extend and modify it as desired. This will lead to incompatibility with the global model and thus cause issues when comparing resources described with different models. Furthermore, the pre-defined capability matching and combined capability rules may no longer be usable after internal modifications to the model. Thus, if the resource providers want to make their offerings available through global resource catalogues, for the sake of interoperability and compatibility, they should use the global MaRCO model, which is extended in a controlled and coherent manner.

Discussion

This article presented the development process of a common knowledge representation for manufacturing resource capability information. The developed information model, MaRCO, provides resource vendors with a common, vendor-independent way to describe the capabilities of their resource offerings. Since MaRCO is a formal OWL-based specifica-

[1] http://resourcedescription.tut.fi/ontology/resourceModel.owl.

[2] http://urn.fi/urn:nbn:fi:csc-kata20180322182123116691.

tion, it also enables information interoperability across a wide variety of different design and planning tools.

MaRCO can ease and speed up the design and decision making related to different phases of production system design, reconfiguration, and even self-organization. In particular, together with the combined capability rules, implemented with SPIN, as well as the capability matchmaking rules and the associated SW tool currently under development (Järvenpää et al. 2017), it can support at least the following activities related to production system design and reconfiguration: Finding resource combinations that fulfill a given product requirement; Checking whether the current system is able to fulfill the given requirement; Making sure that all the required system components are present in the system; Indicating any missing capabilities in the current system and suggesting alternative resources with the needed capability; Showing the parameter range of the available capabilities to facilitate decision-making about parametric adaptation; Showing the available capabilities on the factory floor and their coarse location to provide the necessary information for the logical adaptation (e.g. re-routing); and Automatic generation of configuration scenarios based on the given product requirements.

Often in real production environments the properties of the combined capabilities emerge as a behavior 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 exact physical relation between the combined resources. This information is not handled with the current model, and can not thus be taken into consideration. Even though the combined capability rules can sometimes produce only crude estimations of the combined capabilities, it is expected that this approach can reduce the workload of a system designer and reconfiguration planner. In particular, the developed knowledge representation and associated software tools facilitate the automatic search of suitable candidate solutions from vast amounts of input information, i.e. from multiple resource catalogues. With this approach, it is possible to explore a large solution space and rapidly filter out unsuitable resources, leaving only the possible resources and resource combinations for the given requirement. Therefore, it could ease up adaptation planning and provide remarkable savings in the time used for system design, reconfiguration, production planning, and order dispatching. Moreover, it can provide human designers with new, previously unthinkable solutions to manufacturing problems.

The developed MaRCO model contributes to the existing resource and capability modeling research by providing an approach for modeling the combined capabilities of multiple co-operating resources. This eliminates the need to describe the resource combinations manually. Instead, they can be dynamically created for certain requirements based on resource descriptions of single resources. In addition, the existing resource combinations can be decomposed into single resources for allowing automatic reasoning methods to provide suggestions for reconfiguration actions. Such suggestions may relate, for example, to changing the magazine of a tube feeder to allow differently sized parts to be fed by the feeder, or to changing the gripper of a robot in order to manipulate parts of different weights. Furthermore, the presented approach also describes the parameters relating to the capabilities and allows rules to be created to infer the parameters of combined capabilities from the parameters of the simple capabilities and to insert them to the model. It is a unique approach and implementation which has not been presented by other researchers. Similar conceptual ideas for capability matching have been presented, e.g. in Ameri et al. (2012). The main difference, however, is in the ability to automatically manage combined capabilities, which first of all, allows the resources to be described on a lower level of granularity, and secondly, eliminates the need to describe the combined capabilities manually for each possible resource combination.

Conclusions

The goal of this article was to develop a formal ontology for representing capabilities of manufacturing resources in a vendor-neutral format, in order to support rapid semi-automatic system design and reconfiguration of production systems. Second goal was to present the systematic development process of the model, following ontology engineering methodology. Web Ontology Language (OWL) was used to model the capability and resource related concepts, properties and relations between these concepts. As OWL is not able to infer and assert new instances to the ontology, nor to perform complex arithmetic operations, SPARQL Inferencing Notation (SPIN) was used to extend the reasoning abilities of pure OWL-ontology and to perform the needed combined capability calculations as well as instance and property assertions.

The contribution of the paper is twofold. From scientific perspective, it presented a novel information model, Manufacturing Resource Capability Ontology (MaRCO), which overcomes many of the limitations identified in other existing resource description approaches. Compared to other approaches, the main contribution of MaRCO lies in its ability to model and infer information about combined capabilities of aggregated resources on a parametric level. From practical perspective, the article gave an illustrative example of exploiting the ontology engineering methodology in the development of a domain ontology. The included description of the development process reveals the reasoning behind the design decisions, which is rarely seen in other research

papers on practical ontology development in the manufacturing domain.

The presented model is currently being tested with industrial companies in near-real industrial scenarios in automotive and aerospace assembly. The model will be gradually evolved and extended in response to the results and feedback got from the tests. Furthermore, the resource interface description is presently being integrated into the Resource Model. In addition, the rules and associated software for performing the capability matchmaking between product requirements and resource capabilities are in implementation. In the future, new industrial projects will be established to test the model and associated capability matchmaking in wider industrial settings covering larger amount of different process capabilities. Consequently, new capability classes and their associated properties will be implemented to increase the capability catalogue when needed.

Acknowledgements This research has received funding from the European Union's Horizon 2020 research and innovation program 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).

References

Ameri, F., & Dutta, D. (2006). An upper ontology for manufacturing service description. In *ASME conference proceedings* (pp. 651–661).

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

Ameri, F., Urbanovsky, C., & McArthur, C. (2012). A systematic approach to developing ontologies for manufacturing service modeling. In *Proceedings of the workshop on ontology and semantic web for manufacturing, 2012*.

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

Baheti, R., & Gill, H. (2011). Cyber-physical systems. In *The impact of control technology* (pp. 161–166).

Barata, J., Camarinhamatos, L., & Candido, G. (2008). A multiagent-based control system applied to an educational shop floor. *Robotics and Computer-Integrated Manufacturing*, *24*(5), 597–605.

Bengel, M. (2007). Modelling objects for skill-based reconfigurable machines. In *Innovative production machines and systems, 3rd I*PROMS virtual international conference 2007* (p. 13).

Borgo, S., & Leitão, P. (2007). Foundations for a core ontology of manufacturing. In *Integrated series in information systems* (Vol. 14).

Chapurlat, V., Diep, D., Kalogeras, A., & Gialelis, J. (2007). Building and validating a manufacturing ontology to achieve interoperability. In R. J. Gonçalves, J. P. Müller, K. Mertins, & M. Zelm (Eds.), *Enterprise interoperability II*. London: Springer. https://doi.org/10.1007/978-1-84628-858-6_28.

ElMaraghy, H. A. (2006). Flexible and reconfigurable manufacturing systems paradigms. *International Journal of Flexible Manufacturing Systems*, *17*(4), 261–276.

Frei, R., Di Marzo Serugendo, G., Pereira, N., Belo, J., & Barata, J. (2010). Implementing self-organisation and self-management in evolvable assembly systems. In *IEEE international symposium on industrial electronics* (pp. 3527–3532).

Gruber, T. A. (1993). Translation approach to portable ontology specification. *Knowledge Acquisition*, *5*(2), 199–220.

Gruninger, M., & Menzel, C. (2003). The process specification language (PSL) theory and applications. *AI Magazine*, *24*(3), 63–74.

Guarino, N., & Welty, C. A. (2009). An overview of OntoClean. In S. Staab & R. Studer (Eds.), *Handbook on ontologies* (2nd ed., pp. 201–220). New York: Springer. ISBN 978-3-540-70999-2.

Horrocks, I., et al. (2004) SWRL: A semantic web rule language—Combining OWL and RuleML. W3C member submission. http://www.w3.org/Submission/SWRL/. Accessed March 10, 2016.

Hu, Y., Tao, F., Zhao, D., & Zhou, S. (2009). Manufacturing grid resource and resource service digital description. *The International Journal of Advanced Manufacturing Technology*, *44*(9–10), 1024–1035.

Jardim-Goncalves, R., Sarraipa, J., Agostinho, C., & Panetto, H. (2011). Knowledge framework for intelligent manufacturing systems. *Journal of Intelligent Manufacturing*, *22*(5), 725–735.

Järvenpää, E., Luostarinen, P., Lanz, M., & Tuokko, R. (2011). Presenting capabilities of resources and resource combinations to support production system adaptation. *IEEE International Symposium on Assembly and Manufacturing (ISAM)*, *2011*, 6.

Järvenpää, E. (2012) Capability-based adaptation of production systems in a changing environment. Ph.D. thesis, Tampere University of Technology.

Järvenpää, E., Siltala, N., & Lanz, M. (2016). Formal resource and capability descriptions supporting rapid reconfiguration of assembly systems. In *Proceedings of the 12th conference on automation science and engineering, and international symposium on assembly and manufacturing* (pp. 120–125). IEEE.

Järvenpää, E., Siltala, N., Hylli, O., & Lanz, M. (2017). Capability matchmaking procedure to support rapid configuration and reconfiguration of production systems. *Procedia Manufacturing*, *19*, 87–94. https://doi.org/10.1016/j.promfg.2018.01.013.

Järvenpää, E., Hylli, O., Siltala, N., & Lanz, M. (2018a). Utilizing SPIN rules to infer the parameters for combined capabilities of aggregated manufacturing resources. In *16th IFAC symposium on information control problems in manufacturing, 11–13 June 2018, Bergamo, Italy*.

Järvenpää, E. Siltala, N., Hylli, O., & Lanz, M. (2018b). Product model ontology and its use in capability-based matchmaking. In *51st CIRP conference on manufacturing systems, 16–18 May 2018, Stockholm, Sweden*.

Kitamura, Y., Koji, Y., & Mizoguchi, R. (2006). An ontological model of device function: Industrial deployment and lessons learned. *Applied Ontology*, *1*(3–3), 237–262.

Knublauch, H. (2016). The TopBraid SPIN API. http://topbraid.org/spin/api/. Accessed April 1, 2017.

Koren, Y., & Shpitalni, M. (2010). Design of reconfigurable manufacturing systems. *Journal of Manufacturing Systems*, *29*(4), 130–141.

Leitão, P., Colombo, A. W., & Karnouskos, S. (2016). Industrial automation based on cyber-physical systems technologies: Prototype implementations and challenges. *Computers in Industry*, *81*, 11–25.

Lemaignan, S., Siadat, A., Dantan, J.-Y., & Semenenko, A. (2006). *MASON: A proposal for an ontology of manufacturing domain*. Prague: Institute of Electrical and Electronics Engineers Computer Society.

Lohse, N. (2006). Towards an ontology framework for the integrated design of modular assembly systems. Ph.D. thesis, University of Nottingham (p. 245).

Malec, J., Nilsson, A., Nilsson, K., & Nowaczyk, S. (2007). Knowledge-based reconfiguration of automation systems. In *3rd annual IEEE conference on automation science and engineering, 2007* (pp. 170–175).

Matsokis, A., & Kiritsis, D. (2010). An ontology-based approach for Product Lifecycle Management. *Computers in Industry*, *61*(8), 787–797.

Mehrabi, M. G., Ulsoy, A. G., & Koren, Y. (2000). Reconfigurable manufacturing systems: Key to future manufacturing. *Journal of Intelligent Manufacturing*, *11*(4), 403–419.

Noy, N. F., & McGuinness, D. L. (2001). Ontology development 101: A guide to creating your first ontology. Technical report SMI-2001-0880, Stanford Medical Informatics.

Obitko, M., Vrba, P., Marik, V., Radakovic, M., & Kadera, P. (2010). Applications of semantics in agent-based manufacturing systems. *Informatica*, *34*, 315–330.

Pfrommer, J., Stogl, D., Aleksandrov, K., Schubert, V., & Hein, B. (2014). Modelling and orchestration of service-based manufacturing systems via skills. In *Proceedings of the IEEE emerging technology and factory automation, 2014*.

Protégé. (2015). Protégé Ontology Editor web page. http://protege.stanford.edu/. Accessed November 1, 2015.

Rauschecker, U., & Stöhr, M. (2012). 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*.

Ray, S., & Jones, A. T. (2006). Manufacturing interoperability. *Journal of Intelligent Manufacturing*, *17*, 681–688.

Salonen, J., Nykanen O, Ranta P. A., et al. (2011). An implementation of a semantic, web-based virtual machine laboratory prototyping environment. In Aroyo, L., et al. (Eds.), *The semantic web ISWC 2011 10th international semantic web conference, proceedings, Part II. LNCS* (Vol. 7032, pp. 221–236).

Shin, J., Kulvatunyou, B., Lee, Y., et al. (2013). Concept analysis to enrich manufacturing service capability models. *Procedia Computer Science*, *16*, 648–657.

Siltala, N., Järvenpää, E., & Lanz, M. (2016). Formal information model for representing production resources. In I. Nääs, et al. (Eds.), *Advances in production management systems. Initiatives for a sustainable world. APMS 2016. IFIP advances in information and communication technology* (Vol. 488). Cham: Springer.

Siltala, N., Järvenpää, E., & Lanz, M. (2018). Creating resource combinations based on formally described hardware interfaces. In *Eight international precision assembly seminar*, IPAS (unpublished).

Sintec, M. (2007). OntoViz homepage. https://protegewiki.stanford.edu/wiki/OntoViz. Accessed August 9, 2017.

Sirin, E., Parsia, P. Cuenca, Grau, B., Kalyanpur, A., & Katz, Y. (2007). Pellet: A practical OWL-DL reasoner. *Journal of Web Semantics*, *5*(2), 51–53.

SPIN Working Group. (2017). SPIN—SPARQL inferencing notation. http://spinrdf.org/. Accessed October 15, 2017.

Staab, S., Studer, R., Schnurr, H.-P., & Sure, Y. (2001). Knowledge processes and ontologies. *IEEE Intelligent Systems*, *16*(1), 26–34.

Strzelczak, S. (2015). Towards ontology-aided manufacturing and supply chain management: A literature review. In *Advances in production management systems: Innovative production management towards sustainable growth* (pp. 467–475).

Studer, R., Benjamin, V., & Fensel, D. (1998). Knowledge engineering: Principles and methods. *Data and Knowledge Engineering*, *25*, 161–197.

Sugumaran, V., & Storey, V. C. (2002). Ontologies for conceptual modeling: Their creation, use and management. *Data and Knowledge Engineering*, *42*, 251–271.

Sure, Y., Staab, S., & Studer, R. (2009). Ontology engineering methodology. In S. Staab & R. Studer (Eds.), *Handbook on ontologies* (2nd ed., pp. 135–152). New York: Springer. ISBN 978-3-540-70999-2.

Terkaj, W., & Urgo, M. (2012). Virtual factory data model to support performance evaluation of production systems. In *CEUR workshop proceedings* (Vol. 886, pp. 44–58).

Thoben, K.-D., Wiesner, S., & Wuest, T. (2017). "Industrie 4.0" and smart manufacturing: A review of research issues and application examples. *International Journal of Automation Technology*, *11*(1), 4–16. https://doi.org/10.20965/ijat.2017.p0004.

Timm, I. J., Scholz, T., & Herzog, O. (2006). Capability-based emerging organization of autonomous agents for flexible production control. *Advanced Engineering Informatics*, *20*(3), 247–259.

Uschold, M., & Gruninger, M. (1996). Ontologies: Principles, methods and applications. *Knowledge Engineering Review*, *11*(2), 1–63.

Westkämper, E. (2006). Factory transformability: Adapting the structures of manufacturing. In A. Daschenko (Ed.), *Reconfigurable manufacturing systems and transformable factories* (pp. 371–381). Berlin: Springer.

W3C. (2004). OWL web ontology language—Reference. http://www.w3.org/TR/owl-ref/. Accessed November 1, 2015.

W3C. (2008). SPARQL query language for RDF. http://www.w3.org/TR/rdf-sparql-query/. Accessed March 10, 2016.

Yao, X., Zhou, J., Lin, Y., Li, Y., Yu, H., & Liu, Y. (2017). Smart manufacturing based on cyber-physical systems and beyond. *Journal of Intelligent Manufacturing*.