Mechanisms for Service Composition in Collaborative Cyber-Physical Systems

Artem A. Nazarenko(✉) and Luis M. Camarinha-Matos

School of Science and Technology and UNINOVA-CTS, Nova University of Lisbon, 2829-516 Monte Caparica, Portugal
{aan,cam}@uninova.pt

Abstract. Recent advances in the IoT and Cyber-Physical Systems (CPS) enabled the possibility to combine capillary services or even services from different domains, such as smart home, smart city, smart infrastructure, etc. Thus, it is possible now to talk about the collaborative IoT or collaborative CPS, where services are not isolated from each other, but collaboratively offer added value and should provide advanced mechanisms for conflict resolution. However, still more efforts are needed to explore the ways of how capillary services might be selected and combined in an automated or semi-automated way forming composed or collaborative services. This process includes various stages, such as identification of selection criteria, discovery, negotiation, etc., that need to be considered from the very beginning of the system's design. For this reason, we propose an ontology that addresses issues related to various collaborative aspects of service composition. Moreover, we discuss the composition of services or coalition formation principles with advanced mechanisms for negotiation and conflict resolution considering, for instance, the access rights and ownership.

Keywords: Collaborative Cyber-Physical Systems · Collaborative services · System design

1 Introduction

With the progress of the Industry 4.0 development, the Cyber-Physical Systems (CPS) have been adopted in many areas of human activities, such as home and industrial process automation, healthcare, smart agriculture, etc. According to [2] the current industrial stage presumes deep integration between the operational systems and information and communication technologies (ICT). It is important to understand the processes and main drivers behind this trend. The drivers can be split into social and economic, as well as technological. Examples of social and economic ones are [1]: (i) short development periods, (ii) individualization of demand, (iii) flexibility, (iv) decentralization, and (v) resource efficiency. The technological drivers include: (i) further automation of various manufacturing processes, (ii) digitalization and networking, (iii) miniaturization, and (iv) increased availability of electronic components. In other words, the new CPS are

Published by Springer Nature Switzerland AG 2022
L. M. Camarinha-Matos (Ed.): DoCEIS 2022, IFIP AICT 649, pp. 62–73, 2022.
https://doi.org/10.1007/978-3-031-07520-9_6

not isolated and single problem focused, but rather encompassing multiple aspects and addressing complex systems [2].

Thus, a Complex CPS can be represented as an ecosystem that is composed of tens or hundreds of heterogeneous entities collaborating with each other in order to provide composed services. The modeling of IoT/CPS elements is well known and can be advantageous in multi-owners' landscape [14]. These elements or entities can be both of technical nature, such as sensors and actuators, as well as social nature as represented by human users. The human users can be at the same time the service consumers and the service owners, or the owners of several devices involved in the service provision. Effective support to these complex relations raises the need to introduce collaborative mechanisms for the orchestration of the involved CPS entities. In this regard, we can speak now of Collaborative CPSs that are characterized by "jointly acting and sharing information, resources and responsibilities in order to achieve a common goal" [3]. Thus, the CCPSs, contrary to ordinary CPS, lay larger emphasis on increased connectivity, collaboration, and distributiveness, allowing them to enrich their functionality. Moreover, we can differentiate between the internal collaborative entities – coming from within the considered CPS, and external collaborative entities – coming from the outside. Both the internal and external entities are part of a CCPS environment.

In this context, to deliver a composed service, a CCPS has to allocate the needed competencies and thus discover the components that will be able to provide the necessary functionality. The process of service composition can be compared to the process of coalition formation [4]. The collaborative mechanisms are tightly interrelated with service composition aspects. The main purpose of a service is to satisfy the users' needs through allocation of required capabilities. In general, and in addition to interoperability issues, the process of services composition can face significant obstacles that can arise due to possible conflicts. One example could be, if different users with equal access rights attempt at acquiring the same service at the same time and in the same location/using the same resources. The mechanisms for conflict resolution should be introduced right from the design phase of the Collaborative CPS. Another challenge is to define how different constituents of the Collaborative CPS are interrelated.

In this work, we address these two challenges. Thus, the main research question guiding the work is:

What could be a suitable set of models and organizational structures to support the design of increasingly complex and evolving CCPSs?

However, the current work addresses only some aspects of the general question formulated above. Thus, we can formulate the guiding research sub-questions for this particular paper as:

What could be a suitable upper ontology to represent the service-centric model of a CCPS?

What can be a set of reasoning mechanisms to support collaborative service composition in CCPS?

In the proposed solution, conflict resolution mechanisms are introduced as Prolog rules, and to map different Collaborative CPS concepts we provide an ontology covering the identified critical aspects. A prototype development is then discussed.

2 Contribution to Digitalization and Virtualization

Digitalization can be defined as the process of applying digital technologies [7] and is often interlinked with other terms such as digitization and digital transformation. However, the digitization term is more specifically used to describe the process of moving from analogue to digital data representation, while digital transformation is often seen as process of "restructuring economies, institutions and society on a system level" [8]. The notion of digital transformation can be also applied at the technological level addressing the evolution of legacy systems [9], i.e., the effects of digitalization on characteristics and performance of existing systems. In the context of our work, digitalization is considered as a corner stone due to the need to address the integration of the designed system with other important drivers of digitalization, such as Machine Learning and Automated Reasoning in general.

In this regard virtualization can be seen as an important part of the digitalization process. The notion of virtualization refers to a set of technologies enabling abstraction of underlying resources, while providing a logical view of resources [5]. Implementation of virtualization technologies can significantly improve the performance, facilitate system evolution, and improve system management through representing physical resources in more suitable form for processing. This paper addresses some aspects of virtualization, namely logical representation of physical resources. In this case, physical resources are, for instance, sensors and actuators and services are their logical representations. Virtualization of a CPS can be expressed at various complexity levels, from simple approaches where only a small part of the physical world is depicted, to more complex approaches or digital twins, providing a high level of computational and graphical identity of a CPS or a process [6]. This work contributes to this complex perspective.

3 Design Framework for the Proposed Solution

The proposed solution is based on the Collaborative CPS design framework previously described in [13]. The framework includes three pillars: (i) Application Domain, (ii) CCPS Design, (iii) Knowledge Base, as inspired by the design science research paradigm. The application domain delivers all relevant information about the domain, such as spaces, users, roles, physical environment peculiarities, etc., where the designed system is going to be deployed. This information is derived from the domain specific scenario and is used as an input for the Collaborative CPS design phase.

During the design phase, the input information from the Application Domain is formalized to create a virtual abstraction of the physical environment. During this stage a designer can form the future system from the available functional building blocks. The process of formalization is supported through ontologies and taxonomies stored within the Knowledge Base. For instance, taxonomies can represent the details of a

specific application domain, so that the designer can select the appropriate one, allowing the design tool to stay domain agnostic. The rules, available from the Knowledge Base, specify many aspects of the system behavior, for instance determining the conflict resolution or service composition principles. Moreover, the formalized models of key concepts to be used in the Collaborative CPS are available in the Knowledge Base. In terms of practical implementation, we adopted the following tools: Python environment for the Collaborative CPS design phase, Neo4j graphical database and Prolog engine for the Knowledge Base implementation. The Collaborative CPS design framework is illustrated below (Fig. 1):

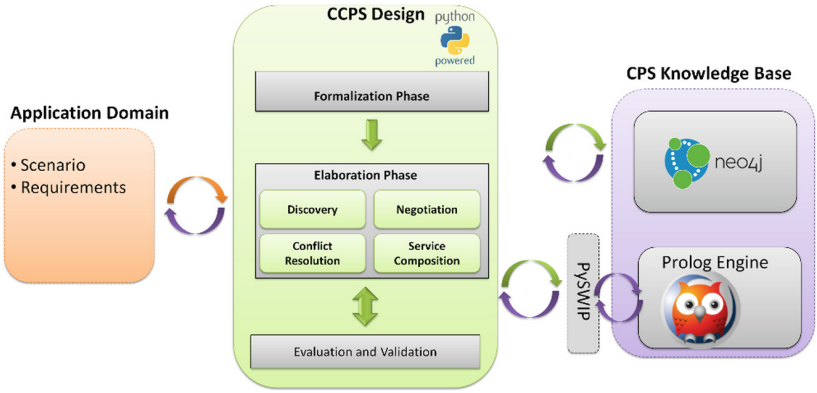

Fig. 1. Collaborative CPS design framework

The aforementioned tools were selected due to a set of reasons. First the Prolog engine provides powerful querying and reasoning capabilities and the PySWIP library allows for querying from within a Python program. Python is a suitable environment for the Collaborative CPS design pillar, enabling simple integration of various functional modules. And finally, the Neo4j, that is a graph-oriented database, is suitable to store different taxonomies, as well as to provide functionalities to track and define the ties between different components of the designed system. In other words, the graph-based approach allows identifying the strength of connections between the different nodes. For instance, during the operation phase, different devices can support a composed service for a limited time, so that they are connected and every time they build a coalition with the same devices the value of connection will change/increase.

4 Ontology

Ontologies are helpful for a wide variety of tasks in the process of CPS design. One example is the task of IoT/CPS system's configuration at the system level [10]. In the mentioned publication, the ontology proposed by the authors represents the structure or main concepts of sensor configuration mechanisms implemented in the CASCoM solution. This ontology, however, is purely focused on the sensor configuration tasks, without tackling the actuators and only slightly touching the high-level abstraction aspects.

Independently of various forms and types, ontologies include a vocabulary of concepts with definitions and the structure on how these concepts are interrelated, limiting the possible interpretations of the concepts in the modeled area [12]. In other words, the general characteristics specific to every ontology are: the vocabulary of concepts that are explicitly defined and the way they are formed into a connected graph. For the specific area of Collaborative CPS design, an ontology addressing the core concepts and relationships among them can be used by a designer to implement the system's layout. This allows for the designer to be focused on the system's functionality and service design, rather than on implementation of basic building blocks supporting the above mentioned process.

In our ontology we intentionally ignore some data processing aspects, such as, for instance, measurement accuracy or quality in order to not overload the ontology, but still be able to demonstrate how the high-level concepts wrap up lower-level details including sensors, actuators, state, etc. Another significant issue with many Ontologies is the problem that they are too domain specific [11]. In the proposed Ontology we tried to stay domain agnostic to some extent (Fig. 2).

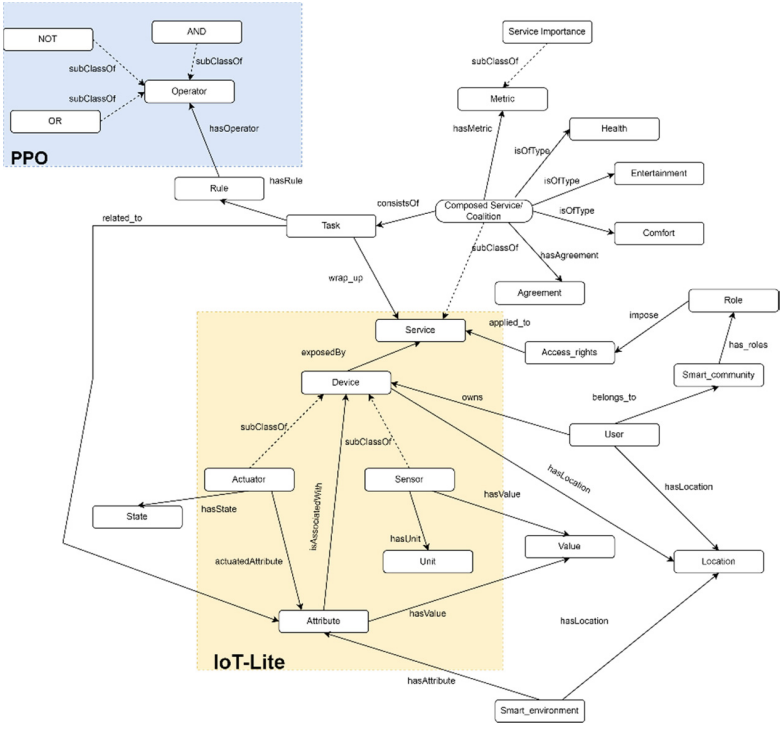

Fig. 2. Excerpt of a collaborative CPS service ontology

The ontology building process is partially based on the approach described in [19]. The first step implies the definition of terminology characterizing the observed domain. During the next step the analysis of available ontologies that, for instance, are developed

and provided by W3C, is accomplished. The following step involves definition of the textual description for the terms selected during the first step or the import of the textual descriptions for the terms used from the W3C ontologies. On the next stage the domain terms that have parent-child (subClassOf) relations are organized in a Taxonomy, similarly to the ISA (is-a) hierarchy. Followed by parthood or excretion of meronymy to establish connections of complex terms to their sub-components. Finally, the last step includes ontology composition.

The proposed Ontology – Collaborative CPS Service Ontology **(CCSO)** – is an ontology explicitly identifying the aspects intrinsic to the process of collaborative service establishment in a CPS ecosystem. The high-level concepts of the ontology are the Smart Environment and the Smart Community. The Smart Environment is the abstraction addressing the technical aspects of the designed system that is used to describe the surrounding enriched with sensing, actuating, and computing capabilities with increased contextual awareness and extended interaction capabilities [16]. The Smart Community, on the other hand, is the abstraction defining the group of human users sharing common interests, preferences, and resources that are having a common social bond. The proposed ontology also includes elements of other well-known ontologies, such as Privacy Preference Ontology (PPO) [17] and IoT-lite [15]. The PPO contains the notion of "Operator" [17] that can be used for the rules design of the planned services. Another ontology used is the IoT-lite which offers, for instance, the notions of the sensor, actuator and device [15] – the underlying definitions for the Smart Environment. Thus, the IoT-Lite is the IoT specific ontology for managing data that covers some important low-level aspects of the design process.

The CCSO ontology considers the following aspects:

- Assuming that every "***Device***" provides a "***Service***", thus device and service are related.
- "***Device***" can be of several types (mostly sensors and actuators, however, not only)
- "***Sensor***" measures certain "***Value***" within the environment.
- The "***Value***" is associated to "***Attribute***". According to [15], "***Attribute***" is some property of an object (e.g. temperature, humidity, etc.)
- On the other hand, "***Actuator***" has a "***State***" (e.g., on/off, open/closed, etc.).
- Both "***Value***" and the "***State***" are related to "***Attribute***".
- A particular case of object that has an "***Attribute***" is the "***Smart Environment***".
- "***Smart Environment***" has an associated physical "Location".
- Moreover, every "***User***" has certain "***Location***" within the system (that might be changed).
- Every "***Device***" is owned by a "***User***".
- Every "***User***" belongs to a "Smart ***Community***", within which he/she has a "Role".
- Every "***Role***" affects the "***Access right***" in certain way.
- "***Access right***" affects the access to a "***Service***".
- A set of "Services" can be combined into a "***Composed Service/Coalition***"
- "***Composed Service/Coalition***" is of a certain type (e.g., Health, Entertainment, Comfort).
- Moreover, it possesses a set of "***Metrics***" to assess the service.

– It also has a set of "***Tasks***". The notion of a "Task" has the meaning similar to the one of the business process [18], since the business process defines the way the "Service" is performed. "Task" has also a set of associated "Rules".

For the next stage, we are planning to elaborate in more details the willingness of devices to collaborate. The idea is that devices/assets have two operating modes – the so called "slave" mode, when the user/admin tells them what to do and the "free reasoning" mode when devices with the help of reasoning rules can discover the new assets and negotiate possible coalition formation. For the second case there is a need of ontologies that will define how the device/asset has to proceed to form a coalition. In the design mode, these ontologies can be updated or extended on demand.

5 Rule-Based Reasoning

The reasoning component is the crucial part of the proposed solution. Reasoning mechanisms are responsible for many tasks, such as service discovery, service composition, and conflict resolution. To implement the reasoning part, a decision was made to use the Prolog engine as a powerful reasoning tool. Prolog is used through the PySWIP interface, enabling Prolog querying from within Python programs. Moreover, Python is used as a bridging point between the designer interface, Neo4j graph-oriented database and directly Prolog.

The rules can be used for addressing the reasoning challenges related to various CPS/IoT aspects. One example is the implementation of attribute-based access control (ABAC) model to manage the access in publish-subscribe networks based on MQTT [20]. These authors also consider the conflict resolution policy relying on XACML standard that provides several possibilities for conflict resolution (deny overrides, permit overrides, etc.). The limitation of their approach is that the authors do not consider different user groups having different access rights. In [21] the authors use basic Prolog rules to retrieve the sensor values or acquire data or conditions of the physical devices. The authors mention the conflict resolution module as a part of the future work. Different from the mentioned approaches, we specifically cover the issue of service composition trying to provide a high level technology-agnostic ecosystem for services management with conflict resolution mechanisms.

Concerning service composition, the rule-based approach is addressed in [22] where the authors provide a framework for service composition in assembly systems. The core idea of the framework relies on the notion of microservice implying that a service is represented as a collection of sub-components called microservices. The authors consider that the IoT-compliant assembly workers exposing their properties as microservices. However, the proposed approach is not adapted to the automatic or semi-automatic service discovery and composition. In [23] the authors allocate several stages of the service composition process, namely: (i) encapsulation of all resources as services, (ii) formulation or capture of the user demand, and (iii) bridging the customer demand with existing service or provide the optimal service composition.

The goal of the proposed design framework is to allow the designer not only to define the physical layout or structure of the system, but also to import and embed the

corresponding reasoning mechanisms from the Knowledge Base to the design advisory or supervision component of the designed system. These reasoning mechanisms are delivered as functional ready-to-use blocks that will support the operational phase. One of these reasoning mechanisms is the service discovery, which serves the goal of bridging the user's demand and the proposition or suggesting a configuration for the service composition. The main idea here is to find the devices, e.g., sensors and/or actuators, that might satisfy the user needs that are formulated as user preferences. Below is a code snippet illustrating the discovery mechanisms:

```
from pyswip import Prolog
prolog = Prolog()
prolog.assertz("sensor(sens_1, temperature, 20, room_1, john_smith)")
prolog.assertz("sensor(sens_1, temperature, 20, room_1, john_smith)")
prolog.assertz("sensor(sens_2, humidity, 50, room_1, jane_smith)")
prolog.assertz("actuator(act_1,  temperature,  [increase,  decrease,
maintain], room_1, john_smith)")
prolog.assertz("actuator(act_3,   humidity,   [increase,   decrease,
maintain], room_1, john_smith)")
prolog.assertz("actuator(act_2,  temperature,  [increase,  decrease,
maintain], room_2, john_smith)")
prolog.assertz("preference(john_smith, temperature, 20)")
prolog.assertz("preference(john_smith, humidity, 50)")
prolog.assertz("preference(jane_smith, temperature, 25)")
prolog.assertz("user(john_smith, room_1)")
prolog.assertz("user(jane_smith, room_1)")
prolog.assertz("find_service(User, [Sensor, Actuator]) :- user(User,
Smart_Environment),sensor(Sensor,Feature,_,Smart_Environment,_),actua
tor(Actuator,Feature,_,Smart_Environment,_),preference(User,Feature,_
)")
prolog.query("find_service(%s, [X,Y])")
user = 'john_smith'
seen = set()
result = []
for soln in prolog.query("find_service(%s, [X,Y])" % (user)):
    if soln not in result:
        result.append(soln)
```

By combining different reasoning mechanisms the designer can create a kind of a sequence that will be the part of the supervision component of the designed system. For instance, such sequence can have the following structure: firstly, discover devices with required capabilities secondly check the discovered devices for possible conflicts and then accomplish the service composition. In this way, during the system's development stage, the designer defines the structure and the order of capillary reasoning blocks execution through sequences establishment. For instance, the above-mentioned code allows the system to retrieve the sensors and actuators that are coping with the same attribute or parameter, are in the same location, and can satisfy the user need formulated within the Prolog fact "preference". It is worth mentioning, that a set of user's preferences can be activated automatically. For instance, considering a Smart Home scenario, if a user enters a room, the temperature parameter can be automatically adjusted based on the "preference" field. This example serves the goal of demonstration that the system designer gets the discovery mechanism from the knowledge base without the need to

additionally implement it. The discovery reasoning mechanism is part of another tool used to launch the negotiation process, if, for instance, the devices have different owners or are used by another user. After devices with required capabilities are discovered, conflict check mechanisms that need to find contradictions in users' needs and preferences that are in the same Smart Environment are launched. For this purpose, we consider the following code as a part of a complex conflict check reasoning mechanism:

```
check_task(Task,                          Bag)                          :-
task(Task,Sensor,_,_,Actuator,_,_,_,Location),          findall(Name,
(task(Name,Sensor,_,_,Actuator,_,_,_,Location), Name \= Task), Bag).

no_conflict(Task,H)   :-   check_task(Task,   Bag),   member(H,   Bag),
task(H,_,_,Tar_Value,_,_,Tar_Action,_,_),task(Task,_,_,Ini_Value,_,_,
Ini_Action,_,_), Ini_Value == Tar_Value, Ini_Action == Tar_Action.
```

The "Task" specifies the flow of actions required to provide a service addressing specific user need. In the case of composed service, the task has also the orchestration functions of different sub-services that are comprised in the composed service. Thus, the Task identifies the parameter/attribute that needs to be changed, as well as the required actuator action. As an example of actuator action – a thermostat example, which can increase the temperature, if it is lower than the one identified in the user preference, decrease if it is higher or maintain it, if it is around the preferred one. Thus, the first part of the code checks if there are tasks manipulating the same parameter and the second one defines the tasks that have no conflict. If the users have the same preferences there is no conflict, even if the users have different access rights and roles.

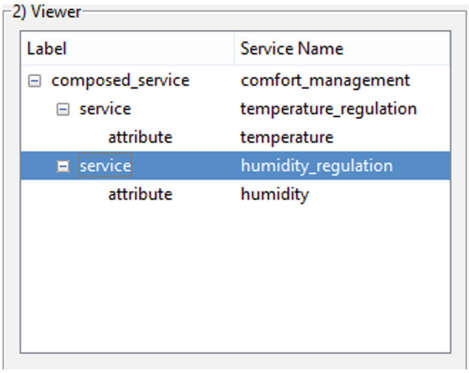

Fig. 3. Composed service viewer

The Fig. 3 demonstrates the part of the prototype, namely the viewer displaying the structure of the composed services. The designer can establish a composed service through combination of different sub-services. The idea is that every composed service consists of a set of capillary services managing various attributes. In this example, the composed service "comfort_management" consists of the "temperature_regulation"

and "humidity_regulation" sub-services. Every sub-services will subsequently have an attribute (humidity, temperature, etc.), which later, during the operation phase, will be connected with the real world devices delivering capabilities related to attribute. The proposed framework should assist the CCPS designer in building the system's structure and the logical core of the system. The logical core, in this context, contains a set of reasoning modules composed of reasoning rules. The framework enables the virtualization, i.e., abstraction from the low level, when the physical assets with corresponding functionality are managed as virtual entities (like digital twins). Contrary to some mentioned approaches, the proposed framework is intended to be technology agnostic, possessing some general reasoning modules that can be applied to various application domains. The next steps planned include the enriching of the reasoning core with further reasoning mechanisms and to consider the semi-automatic service composition during the operational phase. However, the focus is on the design phase and the tools that can be helpful for the CCPS designer to provide customized solutions.

6 Conclusion

This work is a logical continuation of previous works addressing some practical aspects of the Collaborative CPS design framework. The rising need to improve the digitalization of the complex systems affecting the digital transformation, forces to take the digitalization related challenges into consideration. In this regards, complex CPS composed of different heterogeneous constituents has to be designed in a certain way, when different mechanisms for services composition, conflict resolution, service discovery, etc., are stipulated. These reasoning mechanisms are the part of the supervision component of the designed system that will be executed during the operation phase. The designer is able to combine these mechanisms to develop a supervision component supporting the designed system. We provide some examples of the reasoning mechanisms devoted to service discovery and conflict check implemented in Prolog. These will be available for the designer from the Knowledge Base of the framework. Moreover, a collaborative service ontology is provided, indicating core concepts and their interrelations. Finally, the part of the CCPS design framework to construct the composed services was provided.

As a part of further work, we are developing rules for service composition that will include coalition formation and dissolution mechanisms, as well as introducing some domain-specific taxonomies that will be stored and available from Neo4j.

Acknowledgments. This work was supported in part by the Portuguese FCT foundation through the program UIDB/00066/2020.

References

1. Lasi, H., Fettke, P., Kemper, H.-G., Feld, T., Hoffmann, M.: Industry 4.0. Bus. Inf. Syst. Eng. **6**(4), 239–242 (2014). https://doi.org/10.1007/s12599-014-0334-4
2. Frank, A.G., Dalenogare, L.S., Ayala, N.F.: Industry 4.0 technologies: implementation patterns in manufacturing companies. Int. J. Prod. Econ. **210**, 15–26 (2019). https://doi.org/10.1016/j.ijpe.2019.01.004

3. Nazarenko, A.A., Camarinha-Matos, L.M.: Towards collaborative cyber-physical systems. In: 2017 International Young Engineers Forum (YEF-ECE), Almada, pp. 12–17 (2017). https://doi.org/10.1109/YEF-ECE.2017.7935633
4. Barata, J., Camarinha-Matos, L.M.: Coalitions of manufacturing components for shop floor agility - the CoBaSA architecture. Int. J. Network. Virtual Org. **2**, 50–77 (2003). https://doi.org/10.1504/IJNVO.2003.003518
5. Yu, F.R., Liu, J., He, Y., Si, P., Zhang, Y.: Virtualization for distributed ledger technology (vDLT). IEEE Access **6**, 25019–25028 (2018). https://doi.org/10.1109/ACCESS.2018.2829141
6. Martins, H.C., Neves, C.: Shop floor virtualization and industry 4.0. In: 2019 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC), pp. 1–6 (2019). https://doi.org/10.1109/ICARSC.2019.8733657
7. Ritter, T., Pedersen, C.L.: Digitization capability and the digitalization of business models in business-to-business firms: past, present, and future. Ind. Mark. Manag. **86**, 180–190 (2020). https://doi.org/10.1016/j.indmarman.2019.11.019
8. Rachinger, M., Rauter, R., Müller, C., Vorraber, W., Schirgi, E.: Digitalization and its influence on business model innovation. J. Manuf. Technol. Manag. **30**(8), 1143–1160 (2018). https://doi.org/10.1108/jmtm-01-2018-0020
9. Osório, A.L., Camarinha-Matos, L.M., Dias, T., Gonçalves, C., Tavares, J.: Open and collaborative micro services in digital transformation. In: Camarinha-Matos, L.M., Boucher, X., Afsarmanesh, H. (eds.) PRO-VE 2021. IAICT, vol. 629, pp. 393–402. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-85969-5_36
10. Perera, C., Zaslavsky, A., Compton, M., Christen P., Georgakopoulos, D.: Semantic-driven configuration of internet of things middleware. In: 2013 Ninth International Conference on Semantics, Knowledge and Grids, pp. 66–73 (2013). https://doi.org/10.1109/SKG.2013.9
11. Agarwal, R., et al.: Unified IoT ontology to enable interoperability and federation of testbeds. In: 2016 IEEE 3rd World Forum on Internet of Things (WF-IoT), pp. 70–75 (2016). https://doi.org/10.1109/WF-IoT.2016.7845470
12. Hildebrandt, C., et al.: Ontology building for cyber-physical systems: application in the manufacturing domain. IEEE Trans. Autom. Sci. Eng. **17**(3), 1266–1282 (2020). https://doi.org/10.1109/TASE.2020.2991777
13. Nazarenko, A.A., Camarinha-Matos, L.M.: The role of digital twins in collaborative cyber-physical systems. In: Camarinha-Matos, L.M., Farhadi, N., Lopes, F., Pereira, H. (eds.) DoCEIS 2020. IAICT, vol. 577, pp. 191–205. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45124-0_18
14. Gonçalves, C., Osório, A.L., Camarinha-Matos, L.M., Dias, T., Tavares, J.: A collaborative cyber-physical microservices platform – the SITL-IoT case. In: Camarinha-Matos, L.M., Boucher, X., Afsarmanesh, H. (eds.) PRO-VE 2021. IAICT, vol. 629, pp. 411–420. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-85969-5_38
15. IoT-Lite Ontology. https://www.w3.org/Submission/iot-lite/#term_Attribute
16. Nazarenko, A.A., Camarinha-Matos, L.M.: Basis for an approach to design collaborative cyber-physical systems. In: Camarinha-Matos, L.M., Almeida, R., Oliveira, J. (eds.) DoCEIS 2019. IAICT, vol. 553, pp. 193–205. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17771-3_16
17. Sanchez, O.R., Torre, I., Knijnenburg, B.P.: Semantic-based privacy settings negotiation and management. Futur. Gener. Comput. Syst. **111**, 879–898 (2020). https://doi.org/10.1016/j.future.2019.10.024
18. Camarinha-Matos, L.M., Afsarmanesh, H., Oliveira, A.I., Ferrada, F.: Cloud-based collaborative business services provision. In: Hammoudi, S., Cordeiro, J., Maciaszek, L.A., Filipe, J. (eds.) ICEIS 2013. LNBIP, vol. 190, pp. 366–384. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-09492-2_22

19. De Nicola, A., Missikoff, M.: A lightweight methodology for rapid ontology engineering. Commun. ACM **59**(3), 79–86 (2016). https://doi.org/10.1145/2818359
20. Gabillon, A., Gallier, R., Bruno, E.: Access Controls for IoT Networks. SN Computer Science **1**(1), 1–13 (2019). https://doi.org/10.1007/s42979-019-0022-z
21. Bohé, I., Willocx, M., Lapon, J., Naessens, V.: Towards low-effort development of advanced IoT applications. In: Proceedings of the 8th International Workshop on Middleware and Applications for the Internet of Things (M4IoT 2021), pp. 1–7. Association for Computing Machinery, New York (2021). https://doi.org/10.1145/3493369.3493600
22. Thramboulidis, K., Vachtsevanou, D.C., Kontou, I.: CPuS-IoT: a cyber-physical microservice and IoT-based framework for manufacturing assembly systems. Annu. Rev. Control. (2019). https://doi.org/10.1016/j.arcontrol.2019.03.0
23. Xue, X., Wang, S., Lu, B.: Manufacturing service composition method based on networked collaboration mode. J. Netw. Comput. Appl. **59**, 28–38 (2016). https://doi.org/10.1016/j.jnca.2015.05.003