Nazarenko, A.A., Camarinha-Matos, L.M. (2024). A Human-AI Framework to Design Collaborative Cyber Physical Systems. In: *Technological Innovation for Human-Centric Systems. DoCEIS 2024. IFIP Advances in Information and Communication Technology*, vol 716. Springer, Cham. https://doi.org/10.1007/978-3-031-63851-0_2

A Human-AI Framework to Design Collaborative Cyber Physical Systems

Artem A. Nazarenko[1,2], Luis M. Camarinha-Matos[1]

[1] School of Sciences and Technology, CTS-UNINOVA and LASI, Nova University Lisbon, 2829-516 Monte Caparica, Portugal
[2] Chair of IT Security, Brandenburg University of Technology Cottbus-Senftenberg, Konrad-Wachsmann-Allee 5, 03046 Cottbus, Germany
a.nazarenko@campus.fct.unl.pt, lcm@fct.unl.pt

Abstract. The ongoing digitalization processes across various fields present challenges that demand new approaches to address them. A significant challenge involves the collaborative aspect, where smart components of cyber-physical systems (CPS), as well as CPS themselves must collaborate to perform more complex tasks or to adjust to the changing/evolving environment. This has led to emergence of the concept of Collaborative CPS (CCPS). On its turn, the complexity of CCPS requires novel approaches to support the CCPS design process. Therefore, the CCPS designer requires a clear vision of the design process phases, as well as AI-based tools that can support the design process. This work introduces a framework built in line with the Design Science Research Methodology, unveiling the key stages of the design process. Moreover, a set of tools to support the different design process stages are introduced and discussed.

Keywords: Collaborative Cyber-Physical Systems, Design Science, Collaboration, Design Framework, Human-AI collaboration.

1 Introduction

Over a decade ago, the onset of the 4th industrial revolution initiated a transformation process, not only in the industrial sector, but also in various other domains, including home automation. Cyber-physical Systems (CPS) emerged as a pivotal component of Industry 4.0, enabling intelligent decision-making and leading to higher personalization of products and services [1]. Intelligent decision-making was made possible through collaboration among different system's entities that share information contributing to a data-rich environment. As businesses areas across diverse sectors widely embraced Industry 4.0 principles, a novel concept, Industry 5.0 (also related to Society 5.0), has emerged. In contrast to preceding trends, Industry 5.0 places a significant emphasis on symbiotic systems that combine "human subjectivity and intelligence with the efficiency, artificial intelligence, and precision of machines" [2]. Some authors [1] consider Industry 5.0 as a value-driven initiative rather than a

mere technological revolution, highlighting its focus on human-machine / human-AI interaction technologies and sustainability [3], [18].

Therefore, Industry 5.0 makes use of core technological concepts from Industry 4.0 such as CPS, while also emphasizing human, social, and value creation aspects. In this regard, CPS cannot be anymore merely focused on aspects of networking of machines and their joint actions towards achieving common goals but should also consider the social aspects turning them into Collaborative CPS (CCPS) [4]. In other words, human users become members of a CCPS that can participate in joint decision making and whose contribution can be used in a feedback loop to improve and customize products and services. Thus, in contrary to conventional CPSs, CCPSs should enable the mechanisms enhancing the interconnectivity and collaboration among intelligent entities providing high-level services that are able to adapt to the changing context and evolving user's needs. These trends create the need for a new approach to design CCPS, as most of the traditional approaches are mainly focused on low-level technological aspects, such as communication technologies, Machine Learning-based intelligence enhancement tools, etc. However, a system-level design approach reflecting CCPS intrinsic aspects and human-AI collaboration is missing.

In this work, we address such issues, particularly illustrated with the case of CCPS design for home automation. The smart home or home automation domain drifts from the pure control functions towards socio-technical systems [19], which makes it a relevant application area for CCPS. The application area of the proposed solution is not limited by home automation domain, but requires further research efforts to demonstrate its validity in other application scenarios. The proposed approach is inspired in the Design Research Framework method [5] combining behaviour- and design-science paradigms. The work is guided by the following general research question:

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

Moreover, the design process is split into stages that are aligned with the proposed design-approach. The paper also addresses tools that are utilized to support and/or automate some key design stages. As such, the following aspects are considered: Relevance of presented topic for the Human-Centric systems, in Section 2; Introduction of the proposed approach, in Section3; Requirements Elicitation stage of the design process, in Section 4; and Initial CCPS draft stage that includes service definition, Service Catalogue Formation and Discovery Mechanisms and Service Composition and Conflict Resolution Mechanisms, in Section 5. Finally, Section 6 presents some conclusions and directions for further research.

2 Relevance to Human-Centric Systems

The digitalization process happening in various areas of human activities, including manufacturing, daily routines, etc., allows shifting the focus towards personalization of products and services. This also enables data-rich environments, where data

collected are used to adjust products and services to the users' needs and demands [6]. The process of systems design is also affected by this trend. The designer that develops a blueprint of a system is now able to use a wide range of intelligent automation tools, accelerating the development process. Vivid examples include the mapping of user's requirements specified in the natural language to the functional blocks of the future service, or benchmark establishment to facilitate collaboration among independently acting services. Moreover, the new modelling capabilities allow the designer to assess the efficiency of the future system before deployment.

In this regard, there is a need of novel approaches and methods for design, modelling, development, and seamless integration of advanced cyber-physical systems [7]. This work is focused on an approach to design the Collaborative CPS taking into consideration a tight collaboration of a human designer with a range of design process enhancing/supporting AI-based tools covering the full design cycle from the user's requirements elicitation to the services development to satisfy these requirements. The proposed approach is applied in the smart home domain, which poses a synergy of social and technical aspects [20] requiring human-centric approaches during both design- and run-time stages. The whole design process is guided by the CCPS Design Methodology covering the full design cycle and the supporting tools are aligned with specific design stages. As such, this work contributes to bring practical insights on human-AI collaboration and a new human-centric perspective for CPS design.

3 Overview of the Approach

The current focus of this ongoing research work is mainly on the design phase or design-time stage of a CCPS. During this stage, the designer is primarily engaged in defining the main elements of the system, establishing the basic rules governing the coexistence and communication among these elements, as the main leitmotiv of the CCPS lies in the ability of distributed elements to communicate for collaboration purposes. The goal is to establish a general framework that clearly delineates the main steps that comprise the design process and to provide the necessary tools that support various stages of the design process, ensuring a systematic and effective approach to the creation of Collaborative Cyber-Physical Systems.

Our CCPS design framework [5] assumes a design method inspired by Design Science Research. Previous works [4, 12] have highlighted how the framework is adjusted to the CCPS design. The Design Science Research method is here used as an inspiration for the proposed CCPS design method, and not the research method adopted in developing our work. Accordingly, the designer is guided through a series of detailed steps to develop an artifact. Here, the term "artifact" encompasses both the outcome of the design process and the process of artifact creation itself, which can be considered an artifact as well [5]. The emphasis in this paper lies primarily on the process of artifact (a high-level service that is part of the CCPS) development, including the necessary steps to create it, while evaluation is left for future work.

The general view of the stages comprising CCPS design process is depicted in Fig.1. The initial stage involves elicitation of user requirements, typically captured in

natural language text. This text contains essential details that need to be extracted for consideration during the subsequent formalization process, specifically within the digital services establishment phase. Once the user's requirements are formalized, digital services aimed at fulfilling these requirements are allocated and defined as blueprints. This process involves precisely defining the attributes of each service and how it can fulfill the user's requirements. In the following stage, the devices responsible for enabling the functionality of established services are allocated. The formal descriptions of these devices are added to the Knowledge Base and associations/mapping between these devices and services are established. If some devices are unable to fully satisfy the functional needs of the service, the tool assisting the design process generates a request for the missing functionalities. To avoid potential problems and conflicts that might arise due to various reasons, such as shared resources/devices usage by different services or different access rights of different user groups, some conflict resolution mechanisms should be defined and introduced during the design-time, before the operation starts.

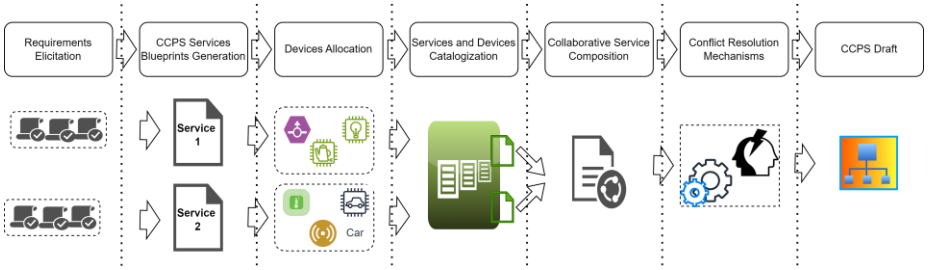

Fig. 1 General view of CCPS design stages

4 Requirements Elicitation

As mentioned above, the initial stage in the CCPS design process is Requirements Elicitation. In this phase, the designer gathers the customer's vision for the future system and delineates the tasks that the system should fulfill or support. One option involves asking the customer to describe the daily and weekend routines in natural language, aiming to understand and extract potential challenges the customer faces in these tasks. For a Smart Home CCPS it is also important to consider not only the tasks accomplished within the Smart Home ecosystem, but also external tasks to find the intersection points where the Smart Home CCPS can contribute. The design requirements are often described in a "design brief" that contains information that is used to frame the system, i.e. what the stakeholders expect to achieve [8]. It is important to note that this document is not a description of the final product or system, but rather a structured set of problems, opportunities, and key objectives for the product/system. This document serves as a foundation for guiding the design process, ensuring alignment with stakeholders' expectations and objectives.

A significant challenge lies in processing the customer's needs to precisely formulate the problems addressable by the system being designed. The designer can

collect crucial information about the customer's needs from a plain description of activities or employing a target questionnaire [8]. In the next stage the identified needs are used as the basis for requirements elicitation [9]. To assist the designer and partially automate the process of requirement elicitation from plain text, a tool that extracts the language patterns from the annotated plain text is proposed. The basic idea involves the designer marking text deemed important, from which a language pattern that considers the linguistic features of the marked text, including the connections among the text parts is extracted. This pattern can be used to further explore the needs represented in the plain text or for the future design, as the patterns are stored within the Knowledge Base. The process of requirements elicitation is illustrated in Fig. 2.

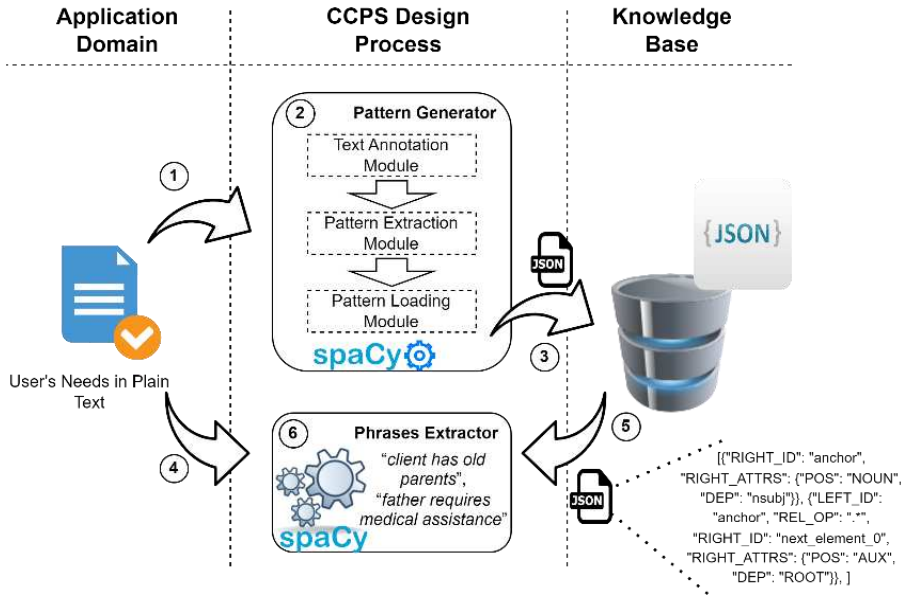

Fig. 2. Requirements Elicitation process as part of the CCPS Design Methodology

Fig. 1 illustrates the whole process of meaningful text pieces / phrases mining / extraction aligned with the proposed CCPS Design Methodology. It includes the following steps:

- During the first step (1) the user's needs, in the form of plain text, are provided to the Pattern Generator tool.
- In the beginning of the second phase (2), the designer uses the Pattern Generator tool that relies on spaCy library for Natural Language Processing (NLP) for the text annotation, i.e., selecting some of the phrases that might be useful for the requirements formulation and formalization of the aimed CCPS. It is important to state that the designer is supposed to possess the domain knowledge in the area to allocate useful information, which is not always possible to do otherwise. Afterwards, the tool checks how the single words are interrelated within the phrases and generates the patterns.

- The generated patterns are converted into JSON (Java Script Object Notation) document and are stored in the Knowledge Base.
- During the next iteration, when there is a new input updating the requirements for the system being designed, or when a completely new system is to be deigned, the stored patterns are used to extract potentially useful information from the plain text. The plain text containing potentially important information for the design process is fed to the Phrases Extractor module (4).
- The Phrases Extractor uploads the stored patterns (5) and checks if there are some matches, which later are either rejected or accepted by the designer (6). The Phrases Extractor is also based on the spaCy functionality.

5 Initial CCPS Design Draft

Following the Requirements elicitation, where user requirements are formulated and formalized, the design process enters its core stage. This phase involves the building blocks creation of the future system, aimed at satisfying the user's requirements. These building blocks define the functionality of the designed system. Various aspects that shape the future system need to be considered and clarified, such as: structure of the building blocks, mechanisms enabling their collaboration including the conflict resolution. After completing and populating the system's layout using the building blocks, the designer must address the mechanisms that enable "comfortable" co-existence and collaboration of the functional elements during run-time. These mechanisms are expressed as a set of rules, which are also available in the Knowledge Base, and are directly related to the templates. The overall approach in the developed prototype follows the Service-oriented paradigm, as it is widely recognized as one of the most prominent ways in design and development of CPS, being also aligned with Industry 4.0 principles that include modularity and service orientation [10].

The templates of both services and devices are stored within the service / device catalogue. Using this catalogue, the designer can automatically allocate the devices that might satisfy the functional requirements defined within the service. In this way, the designer is able to launch the search process for suitable/appropriate devices after finalizing the service template. The tools proposed to support the design process deliver the necessary information about available devices and the extent to which they can satisfy the user requirements that are formulated as the services. Later, during the run-time phase, which is not the focus of this work, the devices can be orchestrated accordingly.

5.1 Service Definition

The first sub-step in preparing the system's draft involves defining the service structure. This encompasses establishing the mechanism through which different devices are incorporated in a service. The service is conceptualized as an intelligent entity that can reason over the different collaboration aspects and can select the appropriate cyber-physical devices based on a set of metrics. Essentially, the service

is viewed as a quasi-autonomous actor that can reason over various aspects of device selection and inter-service collaboration.

The focus in this section is made on introducing the high-level structure of the services that will populate the service catalogue. The formal language chosen to describe the high-level structures is the JavaScript Object Notation (JSON). The choice is driven by several advantages of JSON including the format being human-friendly and it being widely adopted for web services representation [11]. The intention is to provide a JSON structure for a "typical" service aligned with Collaborative CPS Service Ontology outlined in [12]. An additional assumption is that a service can be complex or, in other words, composed of other capillary services, which should be also reflected in the structure. One example is the composed service responsible for comfort regulation inside the smart home that contains one capillary service for temperature regulation and another service that is responsible for light regulation based on the user's presence / activity. The idea is to represent a high-level concept or a feature through the collection of its sub-features "in a hierarchically recursive manner", where the sub-features / concepts are also built-up of their respective features [13].

The structure of the composed service has the following fields:

- "Service name";
- "Anchor", to enable service identification inside the service catalogue.
- "Description", short description of the service purpose.
- "Category", specifying the privileges of the service, for instance, if it is a service related to health, then it will have the highest priority, in the case of entertainment services they can be assigned the lowest priority. This field is important to regulate the access to shared resources, the services with higher importance need to get access first.
- "Tags", needed to simplify the search process.
- "Task", has the same meaning as defined above, where the ontology is introduced. Each service can have multiple tasks.
  Every task field has the following fields: "task_id", identifying the specific task inside the service; "goal", specifying what this task is about; "timer", specifying if the task needs to be automatically triggered in regard to set time intervals; "devices", specifying all devices that are involved in task execution; and finally "rules", intended to define how exactly the task is executed.

5.2 Service Catalogue Formation and Discovery Mechanisms

Once the services are established, they can be uploaded to the Knowledge Base as templates, serving as building blocks for the future CCPS. These templates need to be stored in a reliable and agile manner, so the designer can easily find and retrieve the necessary service templates and automatically associate them with appropriate devices that can provide the needed functionality. The service catalogue for a specific CCPS being designed is formed based on a set of services, whose structure was defined during the previous stage. These services reflect the formalized user needs.

The first step of the catalogue formation process is the storage of service templates/blueprints defined during the previous stage. Service templates are

represented in a JavaScript Object Notation Format (JSON) being a key format for web applications information exchange, which can be both understood by human-developers and processed by machines [21]. Based on the selected format for services representation, an appropriate storage technology, namely MongoDB, has been selected. MonogoDB provides a simple NoSQL style Application Programming Interface (API) that enables the lightweight and agile storage capabilities through create, read, update and delete operations "over collections of schema flexible document entities" [22]. The API is used from the Python application to store and retrieve the service specifications as JSON documents (see Fig. 2).

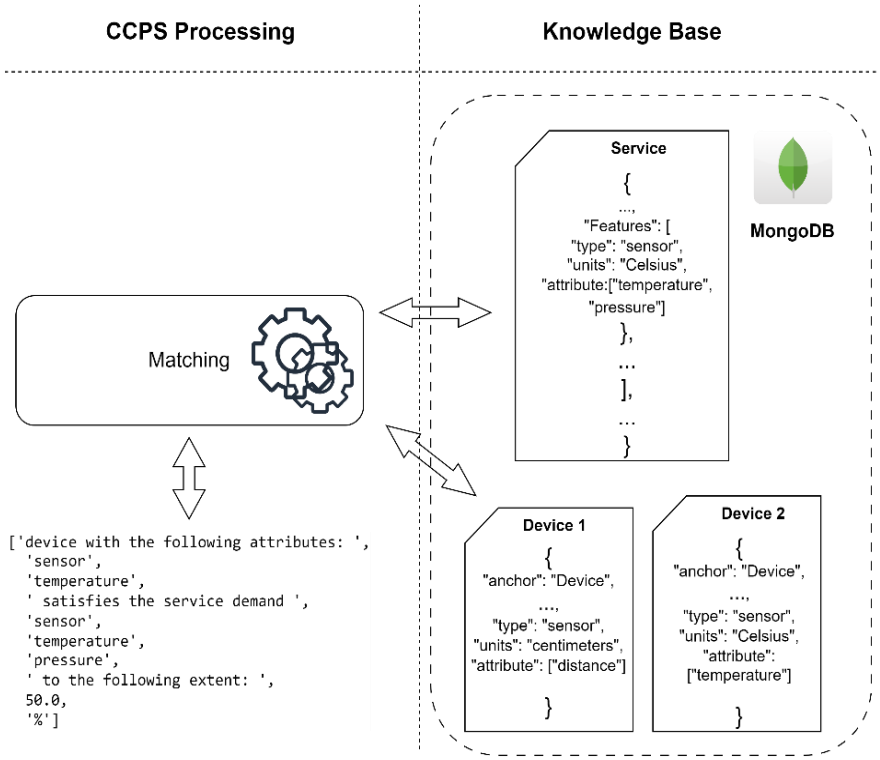

Fig. 3. Matching Process of services and devices acquired from the Knowledge Base

After the catalogue is filled with customized services, a matching process is initiated to find the necessary devices or combinations of devices. The goal in this case is to be able to find the devices that fully or partially satisfy the formalized requirements embodied in the service. This selection process can be influenced by the set of weights that the designer sets for the system layout. Various aspects can affect the final choice, such as: device ownership, if the devices are already booked by another service, etc. Based on the weights and the service templates stored within the Knowledge Base, an automated ranking system provides suggestions for the devices that can be adopted or rejected by the designer (see Fig. 3). Fig.3 illustrates the part of the matching process between the service template and candidate devices that can

potentially deliver the required functionality in the smart home scenario. If the suggestion is adopted, the mapping between the device and the service is made and uploaded to the Knowledge Base.

Direct discovery or matching process presumes that the service and device are formalized in the compliant way, so it is possible to directly select a particular device type (e.g., sensor, actuator, complex device) and the attributes (e.g. temperature, humidity, light intensity, etc.) that can be directly mapped to the needed functionalities. For instance, a temperature regulation service needs both the temperature sensor and the thermostat regulator. On the other hand, indirect matching is a more high-level search/discovery procedure that through the utilization of NLP mechanisms allows discovery of devices that can potentially contribute to the service implementation. This is performed by using the less formalized descriptions of the services and devices. The main goal is to provide the designer with a tool that can suggest suitable devices for the service formation, as well as to provide an initial ranking of the devices' suitability, simplifying the selection process.

The tool that enables search and discovery relies on natural language processing. It performs high-level matching of the service functionality with the high-level description of the device. This stage relies on the similarity metrics calculation between two word-vectors or word embeddings. There are algorithms that enable the word vectors generation, for instance the word2vec, that relies on two approaches or architectures. The first one considers the surrounding words to predict the word and the other one is used to predict multiple surrounding words from the single input word [14]. To implement the tool, an existing language model that ships the word vectors, namely "en_core_web_lg" of Spacy NLP library [15], was used. Every word in the model is represented as a float-number vector of 300 dimensions. These vectors are emerged from observing different words in their context of appearance.

The basic concept behind the process of similarity assessment is that similarity of two words is measured by taking the vectors of these words that are available in the language model and compare them. Typically, in NLP, where the Vector Space Models are widely utilized, to measure the similarity of two objects represented as vectors, the cosine similarity is used [16], which is also the case in spaCy. When the text contains a set of words, the similarity is measured by averaging the vectors for each of the words. It is important to mention that the similarity assessment is inherently subjective, as words can be related to each other in many ways depending on the context and similarity score is always a mix of different signals that consider different contexts [15]. The proposed solution for discovery and matching includes the following steps (see Algorithm 1):

- During the previous stage of the design process, the device catalogue is populated with the records. The designer wants to know if there are any devices that can be used by the service to enable its functionality. For this purpose, designer only needs the description and a set of keywords for both service and devices. This high-level information is loaded from the catalogue.
- Before launching the matching process between the keywords, the keywords are "lemmatized", i.e., assigning base forms of the words (e.g. said – say). Moreover, the synonyms of the keyword are retrieved from the lexical database to improve the coverage of matching/discovery process. However, some of the

synonyms might have less importance depending on the context. For this purpose, for every synonym, the similarity with the keyword is measured to filter less "appropriate" synonyms.

- Then the keywords of a specific functional block of the service are matched against the attributes of the devices in the catalogue, which results in the matching coefficient.
- The last step includes calculating the cumulative coefficient that considers both the description matching and keywords / attributes matching. The combination of both allows reducing the possible biases. Based on the matching coefficient the best match is returned.

Algorithm 1. Calculation of the cumulative similarity coefficient

```
INPUT: spaCy Library, NLTK library, Device Catalogue
OUTPUT: cumulative similarity coefficient
Foreach Device in Device Catalogue:
   Get Device Description and Attributes
   Get Service Description and Keywords
   Set "matches" to empty list
   Foreach Keyword in Keywords:
          Set "synonyms" to empty list
          Lemmatize Keyword
          Assign synonym to "synonyms"
          Foreach Attribute in Attributes:
                Lemmatize Attribute
                      If Keyword = Attribute OR Keyword is
                      in "synonyms":
                      Assign Attribute to "matches"
   Keywords_matches = Length("matches") / Length(Service
   Keyword)
   Description_similarity = spaCy similarity between Device
   Description and Service Description
   Cumul_value = (Keywords_matches + Description_similarity)
   / 2
   RETURN Cumul_value
```

5.3 Service Composition and Conflict Resolution Mechanisms

During the design process itself, additional user needs and requirements might appear, which should be considered in the final draft of the CCPS. The same can happen after the CCPS design is finalized and it is deployed at a run-time. In certain cases, it is possible to satisfy these needs using already existing services, without creating new service templates. If this is the case, the mechanisms supporting the service composition and conflict resolution need to be introduced. It is important to state that various aspects of service composition need to be considered during the design phase before system deployment.

Let's say the new service can be potentially established through the combination of several services already added to the service catalogue. First, there is a need to identify and discover the capabilities that can be acquired from the other services. Afterwards, it should be decided if the available capabilities are enough to satisfy the

newly received requirements/needs. If there are two or more services that could deliver the needed functionalities to establish a new collaborative service or business process, the next stage is launched, when the orchestration mechanisms are defined. The process of service composition is accomplished during the design-phase and includes both the task (by analogy with the business process) and the associated rules that are assigned to every task. If the service has multiple tasks, the designer should also specify the execution order [17]. Moreover, the service design process includes conflict resolution mechanisms, if different services need to utilize the same devices simultaneously, which can potentially cause collisions.

The first phase includes the identification and discovery of available capabilities and if they can satisfy the requirement for the new collaborative service. This process includes the search across the available services stored in the Knowledge Base, supplemented by the mapping of the formalized requirements of the new collaborative service to the capabilities of existing services. The goal is to extract the similarity metric of the capabilities of existing services and the requirements of a new collaborative service. During the second phase, after the "suitable/appropriate" services are allocated, there is a need to define the tasks of the future service and the corresponding rules that should be assigned to it. These tasks will be executed during the run-time (see Fig. 4) by the services selected and used to establish the new collaborative service, but the tasks themselves and the order are to be defined during the design phase.

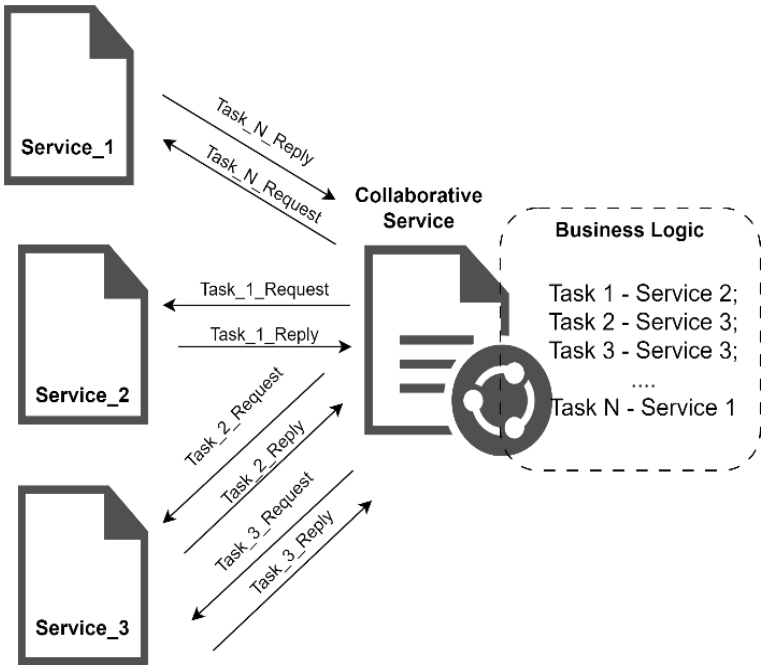

Fig. 4. Generic Schema of Collaborative Service Execution

In the beginning of the service creation/composition process, it is important to provide the description of the service and corresponding tags that should contribute to the service discoverability. After this step, the designer formulates the tasks that the service will execute. One service can encapsulate one or more tasks. An example is a "comfort service" that can have a task responsible for temperature regulation and another task that is responsible for indoor light regulation. In fact, the tasks define the scope of the service. Based on the formulated tasks, the designer can select the devices that will enable the required functionality and subsequently formulate the rules specifying how the task is executed (see Fig. 5).

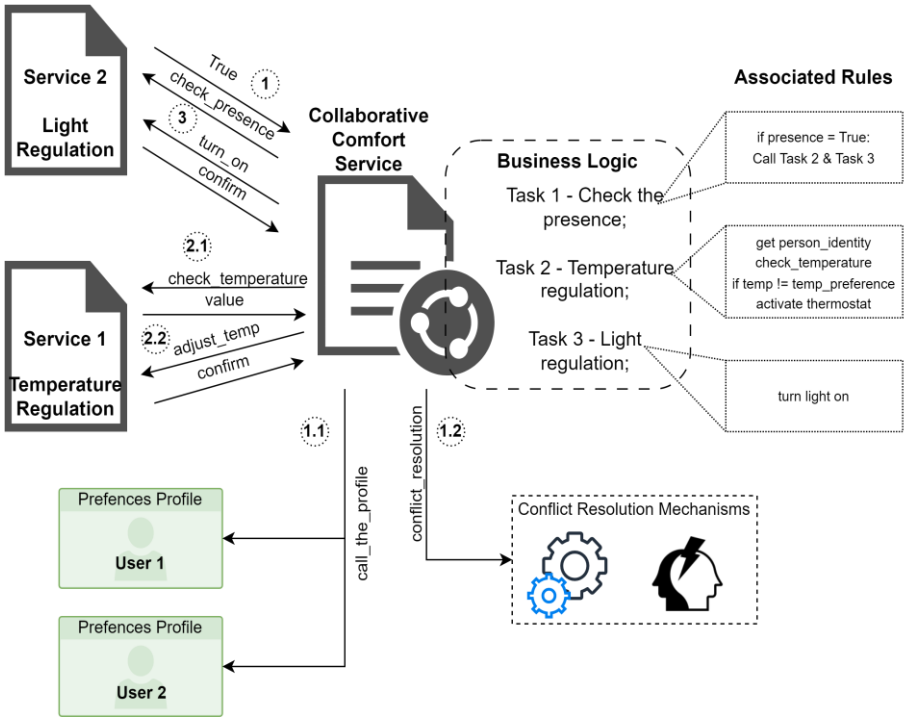

Fig. 5. Example of Collaborative Service Execution

Fig. 5 illustrates an example of a collaborative service execution. It is a simple comfort service, which goal is to check the presence of a user, turn on the light and adjust the temperature based on individual preference profile. Business logic contains 3 tasks with associated rules that specify the manner the service is executed. The first task serves the goal of checking the presence of the person and call task 2 and 3. If the presence is confirmed, the corresponding communication is marked with 1 on the left side of the picture, the profiles of the corresponding users are called (1.1) and if any possible conflicts between the profiles are detected, conflict resolution mechanisms are invoked. The preference profiles can be hardcoded or derived from the users' behavior. After potential conflicts, if any, are resolved the task 2 and 3 are activated. The logic of task 2 is to extract the personal comfort preferences and, if the

temperature deviates from the specified preferences, to adjust the temperature in the room accordingly. The task 3 simply turns the light on, if the presence is confirmed. More sophisticated solution for the task 3 could potentially adjust the light intensity based on the user's activities.

Moreover, service composition should be supported by conflict resolution mechanisms. These mechanisms are needed to avoid potential collisions and bottle necks, if, for instance, a specific service is used simultaneously by several users. In some cases, if the service provides sensor measurements, these can be done without any significant difficulties for several requesting entities. On the other hand, if an actuator or a robot is accessed, there should be a clear policy to resolve potential conflicts. This is done using an appropriate access policy that considers different aspects such as ownership, preference profiles service execution priorities, or position in the queue. Service execution priorities imply that different services can have different priorities. For instance, some services that are related to wellbeing assurance have a greater priority than services related to comfort. For services with equal or comparable priorities the competitive principle is applied, i.e., the service that is the first requesting the resource will have the priority and the other one will be placed in a queue. The combination of these mechanisms should be specified at the design time to be then applied at the run-time.

At the end of the design process, a high-level vision of the CCPS containing a set of blueprints or templates of the services uploaded to the Knowledge Base, as well as the shell mechanisms that enable smooth coexistence of these services in a real-world environment are produced. The shell mechanisms include, inter alia, the conflict resolution and service discoverability, where discoverability is used to address challenges related to CCPS evolution, as new user needs might emerge requiring system re-configuration. The blueprints will be used to deploy the services at the run-time defining the structure, functional blocks, and the manner the services are delivered. Moreover, services are mapped to the specific devices providing the required functionalities. It is important to state that the services blueprints and shell mechanisms are specified for a high-level vision of the CCPS and might require additional middleware for mapping between high- and low-level specifications. The design of the proposed framework assumes the existence of a middleware platform to support the "connection" between the devices and the high-level components of the CCPS. In the case of smart homes, there are a variety of such platforms such as openHAB, Node-RED, domticz, iobroker, among others.

6 Conclusions

The transition to Industry 5.0 significantly increases the importance of CCPS and Human-AI collaboration. In this line, we present a system-level approach that makes use of Design Research method to develop a CCPS design framework. Contrary to traditional approaches, the proposed solution focuses on high-level aspects of the design process, whereas low-level technological aspects are out of the scope of the current work. The main goal is to support a designer with an appropriate approach that gives clear guidelines for CCPS implementation, but also provides the necessary

AI-based tools to support various design stages. The design process itself includes two main stages: Requirements Elicitation and CCPS Draft generation. During the Requirements Elicitation the designer has to convert the user's needs and requirements expressed in the natural language into formalized expressions that can be later turned into services. The CCPS Draft generation includes the following sub-stages: (i) *service definition,* providing the formalization guidelines, (ii) *service catalogue formation and discovery mechanisms,* to store the services generated based on the formalized requirements and enable discoverability for collaboration purposes, and (iii) *service composition and resolution mechanisms,* for complex services formation and to avoid potential conflicts after deployment.

As a part of the future work, we plan to add additional tools to support the design process. One example is the service composition tool that can generate suggestions for potential service combinations from service catalogue in order to satisfy emerging user's needs. Another important step includes validation of the proposed approach. The validation is an ongoing work assuming utilization of the Node-red platform, where the designed services will be deployed. Some of the design evaluation methods that are planned to be used for validation of the CCPS design framework previously mentioned in [5] and adjusted to the current case are listed below:

- Analytical – architectural analysis of designed artifacts is planned to check the artefact compliance with the technical architecture.
- Experimental – the resulting CCPS services will be simulated in the controlled environment with artificial data.
- Testing – a so called "white box" testing is planned, when the logic of the designed artifact is tested during its execution / application.
- Descriptive – a Smart Home scenario will be used to demonstrate the utility of the designed artifacts.

Acknowledgments. The authors acknowledge the Portuguese FCT program UIDB/00066/2020 and UIDP/00066/2020 for providing financial support for this work.

References

1. Xu, X., Lu, Y., Vogel-Heuser, B., Wang, L. Industry 4.0 and Industry 5.0 – Inception, conception and perception. In J. of Manufacturing Systems, Vol. 61, pp. 530-535, (2021)
2. Leng, J., Sha, W., Wang, B., Zheng, P., Zhuang, C., Liu, Q., Wuest, T., Mourtzis, D., Wang, L. Industry 5.0: Prospect and retrospect. In J. of Manufacturing Systems, Vol. 65, pp. 279-295, (2022)
3. Camarinha-Matos, L.M., Rocha, A.D., Graça, P. Collaborative approaches in sustainable and resilient manufacturing. In J of Intelligent Manufacturing, Vol. 35, pp. 499-519 (2024)

4. Nazarenko, A.A., Camarinha-Matos, L.M. (2019). Basis for an Approach to Design Collaborative Cyber-Physical Systems. In: Camarinha-Matos, L., Almeida, R., Oliveira, J. (eds) Technological Innovation for Industry and Service Systems, DoCEIS. IFIP Advances in Information and Communication Technology, Vol. 553. Springer, Cham, (2019)
5. Hevner, A.R., March, S.T., Park, J., Ram, S. Design Science in Information Systems Research. In MIS Quarterly, Vol. 28(1), pp. 75-105, (2004)
6. Wang, B., Zheng, P., Yin, Y., Shih, A., Wang, L. Toward human-centric smart manufacturing: A human-cyber-physical systems (HCPS) perspective. In J. of Manufacturing Systems, Vol. 63, pp. 471-490, (2022)
7. Harkat, H., Camarinha-Matos, L.M., Goes, J., Ahmed, H.F.T. Cyber-physical systems security: A systematic review. In Computers & Industrial Engineering, Vol. 188, (2024)
8. Haug, A. Emergence patterns for client design requirements. In Design Studies, Vol. 39, pp. 48-69, (2015)
9. Cascini, G., Fantoni, G., Montagna, F. Situating needs and requirements in the FBS framework. In Design Studies, Vol. 34 (5), pp. 636-662, 2013
10. O'Donovan, P., Gallagher, C., Leahy, K., O'Sullivan, D.T.J. A comparison of fog and cloud computing cyber-physical interfaces for Industry 4.0 real-time embedded machine learning engineering applications. Computers in Industry, Vol. 110, pp. 12-35, (2019)
11. Bourhis, P., Reutter, J.L., Vrgoč, D. JSON: Data model and query languages. Information Systems, Vol. 89, (2020)
12. Nazarenko, A.A., Camarinha-Matos, L.M. Mechanisms for Service Composition in Collaborative Cyber-Physical Systems. In: Camarinha-Matos, L.M. (eds) Technological Innovation for Digitalization and Virtualization. DoCEIS 2022. IFIP Advances in Information and Communication Technology, Vol 649. Springer, Cham, (2022)
13. Simpkin, C., Taylor, I., Harborne, D., Bent, G., Preece, A., Ganti, R.K. Efficient orchestration of Node-RED IoT workflows using a Vector Symbolic Architecture. Future Generation Computer Systems, Vol. 111, pp. 117-131, (2020)
14. Kenter, T., de Rijke, M. Short Text Similarity with Word Embeddings. In Proc. of the 24th ACM International on Conference on Information and Knowledge Management (CIKM '15). Association for Computing Machinery, New York, NY, USA, pp. 1411-1420, (2015)
15. spaCy 101: Everything you need to know – https://spacy.io/usage/spacy-101#vectors-similarity
16. Sidorov, G., Gelbukh, A., Gomez-Adorno, H., Pinto, D. Soft similarity and soft cosine measure: Similarity of features in vector space model. Comput. Y. Sist. Vol. 18(3), pp. 491-504, (2014)
17. Peltz, C. Web services orchestration and choreography. In Computer, Vol. 36(10), pp. 46-52, (2003)
18. Zheng, Q., Gou, J., Camarinha-Matos, L. M., Zhang, J. Z., Zhang, X. Digital capability requirements and improvement strategies: Organizational socialization of AI teammates, *Information Processing & Management*, Vol. 60, Issue 6, (2023).
19. Sovacool, B.K., Furszyfer Del Rio, D.D. Smart home technologies in Europe: A critical review of concepts, benefits, risks and policies. In *Renewable and Sustainable Energy Reviews*, Vol. 120, (2020)
20. Fang, Y., Lim, Y., Ooi, S.E., Zhou, C., Tan, Y. Study of Human Thermal Comfort for Cyber–Physical Human Centric System in Smart Homes. Sensors. Vol. 20(2):372, (2020)
21. Pezoa, F., Reutter, J.L., Suarez, F., Ugarte, M., Vrgoč, D. Foundations of JSON Schema. In Proceedings of the 25th International Conference on World Wide Web (WWW '16). International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, pp. 263-273, (2016)

22. Liu, Z.H., Hammerschmidt, B., McMahon, D., Chang, H., Lu, Y., Spiegel, J., Sosa, A.C., Suresh, S., Arora, G., Arora, and V. Native JSON datatype support: maturing SQL and NoSQL convergence in Oracle database. In Proc. VLDB Endow, Vol. 13(12), pp. 3059-3071, (2020)