BULLETIN OF THE POLISH ACADEMY OF SCIENCES
TECHNICAL SCIENCES, Vol. 68, No. 2, 2020
DOI: 10.24425/bpasts.2020.131849

A methodology for cloud manufacturing architecture in the context of Industry 4.0

E. TALHI[1], J.-C. HUET[2*], V. FORTINEAU[3], and S. LAMOURI[3]

[1] EIGSI, La Rochelle, France
[2] EFREI Paris, Villejuif, France
[3] LAMIH, CNRS, Arts et métiers ParisTech, Paris, France

Abstract. This paper deals with a methodology for the implementation of cloud manufacturing (CM) architecture. CM is a current paradigm in which dynamically scalable and virtualized resources are provided to users as services over the Internet. CM is based on the concept of coud computing, which is essential in the Industry 4.0 trend. A CM architecture is employed to map users and providers of manufacturing resources. It reduces costs and development time during a product lifecycle. Some providers use different descriptions of their services, so we propose taking advantage of semantic web technologies such as ontologies to tackle this issue. Indeed, robust tools are proposed for mapping providers' descriptions and user requests to find the most appropriate service. The ontology defines the stages of the product lifecycle as services. It also takes into account the features of coud computing (storage, computing capacity, etc.). The CM ontology will contribute to intelligent and automated service discovery. The proposed methodology is inspired by the ASDI framework (analysis–specification–design–implementation), which has already been used in the supply chain, healthcare and manufacturing domains. The aim of the new methodology is to propose an easy method of designing a library of components for a CM architecture. An example of the application of this methodology with a simulation model, based on the CloudSim software, is presented. The result can be used to help the industrial decision-makers who want to design CM architectures.

Key words: cloud manufacturing, cloud-based services, service-oriented architecture, Industry 4.0, ontology.

1. Introduction

Industry 4.0 increases the digitization of manufacturing, in which connected networks of humans and robots interact and work together with shared and analyzed information, supported by big data and cloud computing (CC) throughout the industrial value chains [1]. In this context, a paradigm based on technologies such as service oriented architecture (SOA) and CC has emerged: cloud manufacturing (CM). The resources and services are provided to users in a pay-as-you-go manner. The users can request services concerning any stage of a product lifecycle (product design, manufacturing, testing, management, etc.). CM covers all stages of a lifecycle that are provided as services, although the literature gives the impression that it only treats the manufacturing level.

Xu et al. [2] define CM as "*a model for enabling ubiquitous*, *convenient*, *on-demand network access to a shared pool of configurable manufacturing resources (e.g.*, *manufacturing software tools*, *manufacturing equipment*, *and manufacturing capabilities) that can be rapidly provisioned and released with minimal management effort or service provider interaction*" and propose a multi-layer framework including:

- manufacturing resource layer (MRL): where the needed resources during the lifecycle are contained;
- virtual service layer (VSL): where identified manufacturing resources are virtualized and packaged as a service;
- global service layer (GSL): where the manufacturing resources are located, allocated, and monitored;
- application layer/user domain (UD): where an interface between the user and manufacturing cloud resources are done.

Our goal is to define a CM architecture to support this framework, applied to industrial information systems in the context of Industry 4.0, and more precisely, in the context of product lifecycle management (PLM). PLM is an approach that enables the management of the product's data in such environments. Terzi [3] described PLM as "an integrated, Information and Communication Technology (ICT) supported approach to the cooperative management of all product-related data along the various phases of the product lifecycle". To implement such an architecture, we propose a structured methodology based on the ASDI (analysis–specification–design–implementation) framework.

The paper is structured as follows. Section 2 presents a review about industry 4.0 and synthetic review of the cloud manufacturing models and modelling approaches. The aim is to position cloud manufacturing as a technological lever of Industry 4.0. It is followed by a presentation of the main CC simulation software. Section 3 depicts the adaptation of ASDI framework to our domain and provides a CM model, along with its validation. An example of utilization is shown in Section 4. Then, a case study using CloudSim simulation software is explained in Section 5. Section 6 exposes the limits of the methodology and suggests research prospects. Finally, Section 7 presents conclusions.

*e-mail: jean-charles.huet@efrei.fr

Manuscript submitted 2019-06-30, revised 2020-01-17, initially accepted for publication 2020-03-08, published in April 2020

2. Related work

2.1. Industry 4.0. Many applications in manufacturing systems require a combination of new technologies, which is giving rise to the emergence of Industry 4.0. Such technologies come from different disciplines including cyber-physical systems [4], IoT, Coud Computing [5], enterprise process control [6], software defined network (SDN) [7], industrial integration, enterprise architecture, SOA, business process management, industrial information integration and others. At this present moment, the lack of powerful tools is still a major obstacle for exploiting the full potential of Industry 4.0 [8]. The authors concluded that formal methods and system methods are crucial for realising Industry 4.0, which poses unique challenges. In addition to this observation, the lack of a conceptual framework of interoperability regarding Industry 4.0 is underlined [1].

A prerequisite in Industry 4.0 is that the factories, workshops, and all the manufacturing capacities must have an informational part so that they can be integrated first, and then communicate with services. Therefore they must be parts of a holon manufacturing system (HMS) [9]. A holon [10] is an autonomous entity having a physical part and a logical one, communicating and cooperating with its environment. A holonic manufacturing system (HMS) is an autonomous and co-operative building block of a system for transforming, transporting, storing and/or validating information and physical objects [11]. The concept of holon is closed to the concept of cyber-physical systems included in Industry 4.0 [4].

2.2. Relationship between CC and CM. Before going further into exploring the state of the art, a brief comparison between CC and CM is presented. CM is a paradigm that includes the notion of CC, but is wider. Tao et al. [12] explained that in CM, in addition to the information technology (IT) resources, all manufacturing resources and abilities involved in the whole life cycle of manufacturing aim to be provided for the user in different service models: design as a service (DaaS), manufacturing as a service (MFGaas), experimentation as a service (Eaas), simulation as a service (SIMaaS), management as a service (Maas), maintain as a service (MAaas) and integration as a service (INTaas). For instance (Fig. 1), if a user asks for the design-as-a-service (from CM), this may include other sub-services that are specific to CC. A user can demand a design software (Saas) and a storage capacity to store the data (Iaas), all provided as Design_aas. A transaction is initiated by the client who requests a design service that includes a storage media to process the data. The cloud system then grants the requested services on the basis of other features such as: price, duration of use, etc. When allocating services, both services that are labelled CM (for the design software) and those that are labelled CC (for the storage media) are chosen.

2.3. Relationship between CM and Industry 4.0. According to Danjou et al. [13] (Fig. 2) ten technology groups can be considered to to implement Industry 4.0, completed by [14].

- **Big data and analytics.** In an Industry 4.0 context the collection and comprehensive evaluation of data from many different sources is a standard solution for supporting real-time decision making.
- **Simulation** will leverage real-time data to mirror the physical world in a virtual model, which can include machines, products and humans.
- **Internet of Things.** With IoT, more devices will be enriched with embedded computing and connected using standard technologies to allow field devices to communicate and interact both with one another and with more centralized controllers, if necessary.
- **Cyber-physical systems** – mechanisms that allow for monitoring using communication, data storage and computational capabilities directly incorporated into objects.
- **Cloud.** In Industry 4.0, more production-related undertakings will require increased data sharing across sites and

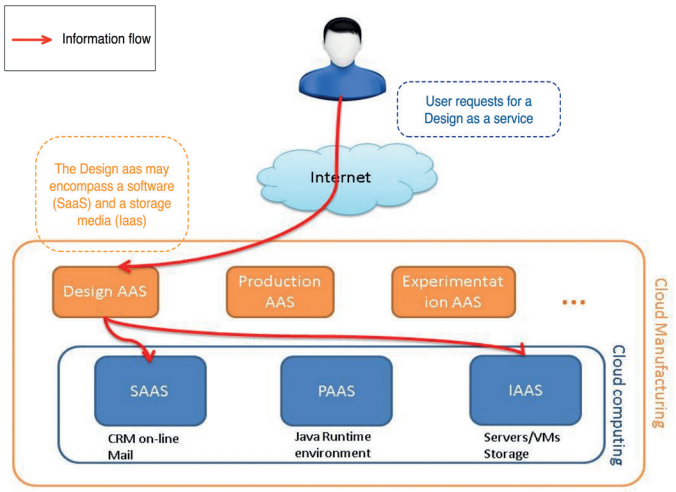

Fig. 1. Usage example

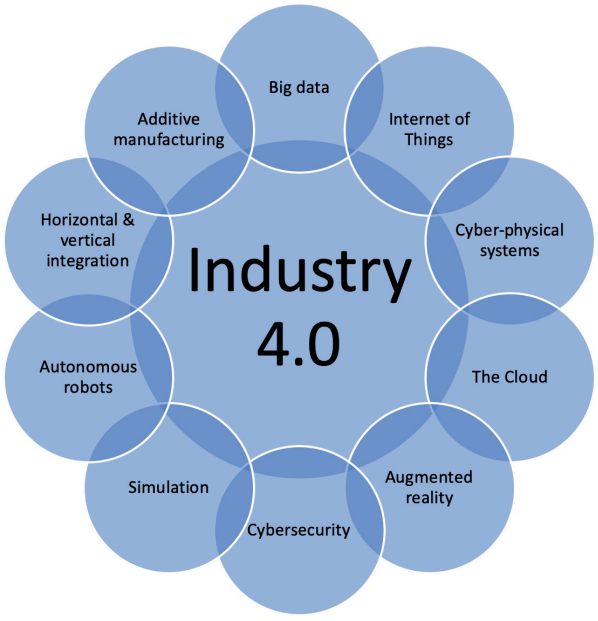

Fig. 2. Technology groups of Industry 4.0

company boundaries. At the same time, the performance of cloud technologies will improve, achieving reaction times of just several milliseconds.

- **Augmented reality.** Companies will make much broader use of augmented reality to provide workers with real-time information to improve decision making and work procedures. For example, workers may receive repair instructions on how to replace a particular part as they are looking at the actual system needing repair.
- **Cybersecurity.** In view of increased connectivity and use of standard communications protocols that come with Industry 4.0, the need to protect critical industrial systems and manufacturing lines from cybersecurity threats increases dramatically. As a result, secure, reliable communications as well as sophisticated identity and access management of machines and users are essential.
- **Autonomous robots.** Robots were first used in industry to tackle complex assignments, but are evolving to become more autonomous, flexible and cooperative to interact with one another.
- **Horizontal and vertical system integration.** In Industry 4.0, companies, departments, functions and capabilities will become much more cohesive as cross-company, universal data-integration networks evolve and enable truly automated value chains.
- **Machine-to-machine communication (M2M).** M2M communication facilitates direct exchange between machines within high scaled parks.
- **Additive manufacturing.** Companies have just begun to adopt additive manufacturing, such as 3-D printing, which they use mostly to prototype and produce individual components. In Industry 4.0, these additive-manufacturing methods will be widely used to produce small batches of customized products that offer construction advantages, such as complex, lightweight designs.

As the CM is a model based on CC, in which all the steps of the product lifecyle are provided to users as services, it can be considered as a technological group of Industry 4.0. Indeed, as explained in Section 2.2, CM provides, in a scalable and pay-as-you-go way IT resources as well as manufacturing resources and capabilities. Therefore, we believe that CM is a relevant model that enables the implementation for example of smart factories. The next section proposes an analysis of how to deploy a robust methodology.

2.4. Methodological aspects. A methodology has been defined by Zellner [15] as a set of five mandatory elements:

1. Procedure model: the order of activities to be performed when using the method.
2. Technique: way to generate results; supports an activity.
3. Results: an artefact (e.g. a document, etc.) created by an activity.
4. Role: the individual who carries out the activity and is responsible for it.
5. Information model: consists of the above-described elements and their relationships. Information models are also employed to represent the results.

We have found in the literature a methodological framework that fulfills these features and meets our needs: ASDI (analysis–specification–design–implementation) [16]. This framework is used for the design and the implementation of modelling, simulation and piloting software environments dedicated to a domain (class of systems). To obtain knowledge models of complex systems, ASDI recommends a systemic decomposition of the studied system in three communicating subsystems: (1) the physical subsystem (PSS) defines the physical entity set (which could concern different fields, such as production, storage, handling and transport), their geographical distribution and the links between them; (2) the logical subsystem (LSS) (called also informational subsystem in some simulation methodologies) represents the flow of entities that have to be handled by the system, along with the set of operations concerning these flows, and the nomenclatures that refer to this set; (3) the decision-making subsystem (DSS) contains the management and working rules of the system. The analysis and specification phases allow us to obtain the generic modelling of a domain while the design and implementation phases allow us to create a library of reusable software components for the domain being studied. ASDI has already been used in different areas: Health care [11], simulation of traffic [17] and manufacturing [18] systems, energy management of data-centers [19] and more recently in the context of lean management [20].

This methodology has two important and interesting parts linked together:

- the capitalization of reusable components into a library,
- generic decision-making tools can be built by evaluating the performance of a scenario.

2.5. Cloud manufacturing modelling approaches. Ghomi and al. [21] analysed the literature on CM and classified the CM works in five categories, i.e., (1) studies focused on the architecture and platform design of CM, (2) studies concentrated on resource description and encapsulation, (3) studies focused on service selection and composition, (4) studies aimed at resource allocation and service scheduling, and (5) studies aimed at service searching and matching.

In this paper, we contribute mainly to (1) and (3) with a methodology to design cloud-based manufacturing architecture. In fact, the modelling methods found in the literature [22, 23] focus on the provider with the aim to model resources and operators on a shop floor or to provide a framework of knowledge integration in networked manufacturing. In fact, the user point of view is generally neglected and no modelling methodology proposes a global service layer standpoint. Furthermore, our goal is to implement an architecture where all phases of the product's lifecycle are provided as services. This makes our model service-centeric because it is the concept that connects users and providers. Wang and Xu [24] propose a methodology for virtualizing the existing manufacturing abilities and resources. The drawback is that the processing of data generated from multiple resources over the Internet is heterogeneous whereas the author used a language which is not suited for portability in heterogeneous environments . An ontology is used by Lu et al. [25] to manage user-defined

clouds within a cloud management engine. It has been used to instantiate companies on which the user executes a customised rule to create a cloud and defines users authorized to access this cloud. The authors do not mention the methodology used to build the architecture. In Lin et al. [26], the research results can support Coud Manufacturing to effectively deal with the challenge of management and allocation for increasingly large-scale and distributed manufacturing resources and capabilities. The authors do not propose the methodology that allows the reuse of such a system. Some work has been undertaken to deal with service composition. For example Bouzary et al. [27] worked on the service composition and its characteristics in CM environments by providing a review of most algorithms for optimization in this field. Lu and Xu [28] proposed a semantic web-based framework in a CM environment. This framework is of interest for use with an ontology, but there is no structured methodology to implement it. The studies above characterize models that represent the related domains without any methodology to explain how to use the models efficiently. To build a CM architecture that respects the multi-layer framework proposed by Xu et al. [2], the ontology seems of interest in order to benefit from a semantic advantage and an inference mechanism.

2.6. Cloud simulation tools. Recently, many cloud environment simulation tools have been proposed to enable the reproducible and controlled evaluation of new algorithms for the management of cloud resources and applications [29].

GreenCloud [30] is a cloud simulator focusing on energy efficiency research. It extends the NS2 simulator[1], and is also able to estimate not only the power consumption of computing resources but also from network resources.

MDCsim (multi-tier data center simulation) is used to measure power and analyse each layer of the 3-layer architecture of the simulator [31] and can modify any layer without impacting any other layer. It has Ethernet and IBA communication protocols over a TCP/IP protocol. It allows for the addition of any other type of communication protocol. This simulator is commercial; in other words, the user needs to buy it for its full functionality. This is the main limitation as opposed to other simulators.

CloudSim [32] is a discrete event-based cloud simulator implemented in Java, enabling the simulation of data centers with a number of hosts. CloudSim is a famous simulator for cloud parameters developed in the CLOUDS Laboratory, at the Computer Science and Software Engineering Department of the University of Melbourne. It is the most popular tool available for simulating a cloud environment. It has become more and more popular because of its extendibility. The programming language is Java, which is a portable language. All that is required is to import the CloudSim package to the IDE (integrated development environment) and start coding an algorithm. When this is complete, the user runs the file and the simulation begins; at the end, the results are provided. It lacks several features, such as a GUI (graphical user interface). Recently, some authors used CloudSim[2] in a multiformalism modeling approach for cloud based systems [33] or to work on IoT-based fog computing issues [34].

In order to make a CM simulation model, we have selected CloudSim because it is the most flexible and adaptable cloud computing simulation software.

6. Proposed methodology

3.1. ASDI adaptation: ASDI-Onto. In order to establish a CM architecture, we propose ASDI-Onto methodology, being is an extension of ASDI. Next, we will explain how we adapted ASDI to our domain of study and to our objective. Using ASDI provides an opportunity to study the systems that belong to the domain with the aim of building a CM architecture. To do so, ASDI-Onto proposes dealing with two abstraction levels: the analysis/validation and design/implementation levels for the domain of study. To illustrate it, ASDI-Onto uses ontologies to develop the generic knowledge model and its implementation using Cloud simulation systems to build the library components. We do not take into account the specification step, where functions and the behavioural model of system objects are normally described. In fact, the behaviour of the components is the responsibility of service providers because ASCI-Onto deals with the CM from the global service layer point of view. Instead of a specification step, we propose a validation step. The operation principle is to test different scenarios in order to check:

- if the ontology takes into account all possibilities;
- the performance of the inference mechanism;
- if the concepts are well formulated;
- any existing ambiguities (the consistency of the ontology).

The proposed adaptation of ADSI to the CM domain is shown in Fig. 3. The modified and added elements when compared to the original framework [16] are written in orange and in italic. The CM Ontology (CMO) is built during the analysis and validation steps and allows for the generic knowledge model to be obtained. The build of the generic domain model is based on enumerating all of the entities needed to describe the domain and the relationships between them (analysis). Our ontology responds to these requirements since it includes all the related concepts found in the literature, which can be used to describe the CM and their relationships.

Furthermore, the CM architecture corresponds to the library of components of ASDI-Onto. The definition of CM system proposed by Xu [2] is used to build the architecture. Our objective is to develop a CM architecture that will match providers and users in a way in which users will find the best suited service for their request. The design step consists of modifying existing cloud computing implementation in order to add concepts related to CM and all the steps of the product lifecycle. Implementation is the final step, based on the design step; it enables the imple-

[1] http: //www.isi.edu/nsnam/ns/ (available June, 2019)

[2] The list of simulation software based on CloudSim can be found at: http://www.cloudbus.org/cloudsim/ (available June, 2019)

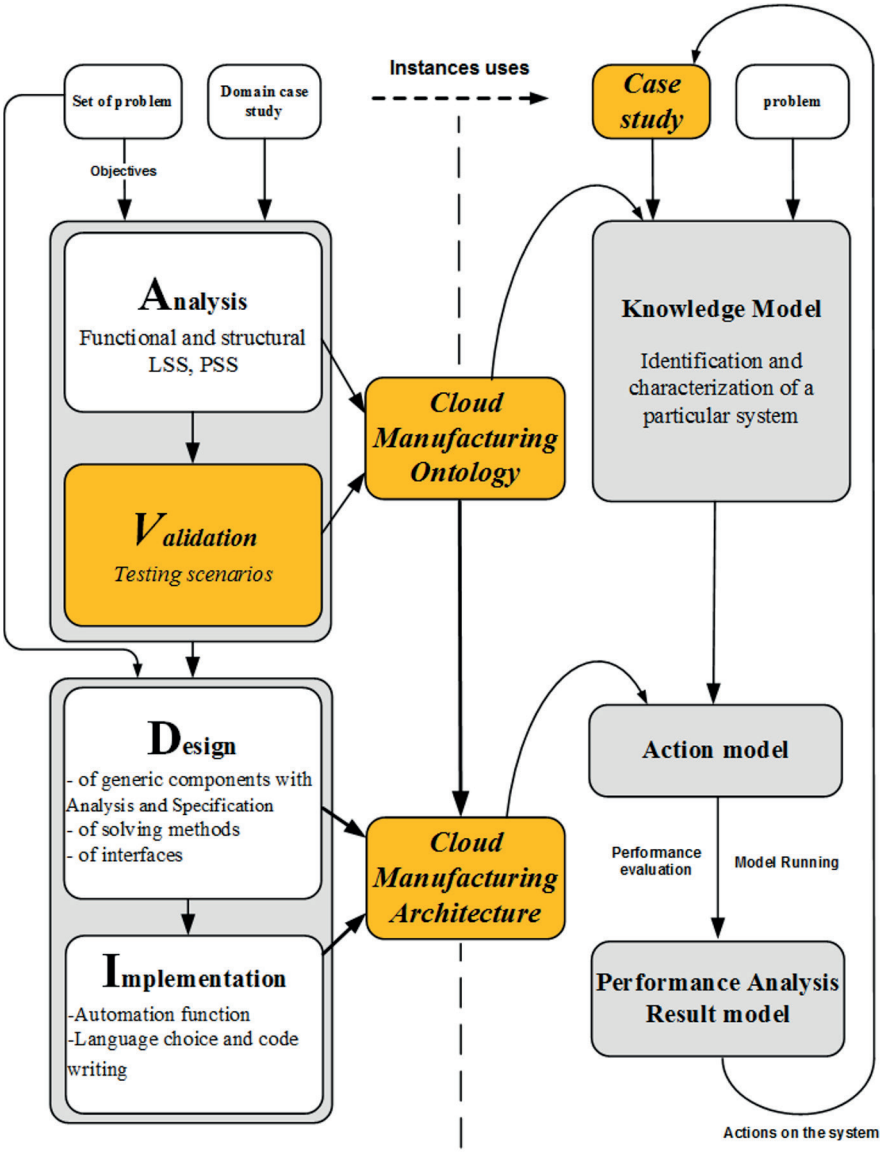

Fig. 3. ASDI-Onto

mentation of CM architecture, which is the action model in the ASDI methodology. Since in the CM everything is viewed as a service, our implementation has to guarantee the integration of various resources organized as a set of services offering the advantages of modularity and reusability in the system.

In the following sections, the analysis/validation steps are explained in order to provide a generic knowledge model: the CMO. We have used Protégé to model the CMO, which is a free, open-source ontology editor and framework for building intelligent systems and is supported by a strong community of academic, government, and corporate users.

3.2. Domain analysis. During the analysis and validation steps, which are dedicated to the domain, the generic model knowledge is built. It brings together the systems of the studied domain by identifying the common entities they contain and their interactions. Simon [35] defines a complex system as "a system made up of a large number of parts that interact in a non simple way". We believe that the Coud Manufacturing system is a complex once, since it maps a set of actors who, according to their statutes, are providers or users of resources virtualized in services. In addition, these services handle all stages of product lifecycle. According to the decomposition proposed by ASDI framework, CMO includes the PSS and LSS. Specifically, it models the different resources as well as the services offered by the resources' owners or used by the consumers. The DSS is not included in the ontology since it must be as generic as possible and in our case, the management of rules is specific to each company. Indeed, management rules can be, for example, the company's decisions concerning the visibility of its services to the system's actors. This company can restrict access to its services if the user is a competitor and the choice of confidentiality rules are not always the same for the entire domain system.

The choice of ontology as a modelling tool is motivated by the following reasons [23]:

- Ontologies provide a simplified and understandable view of the domain since they explicitly represent the meaning of concepts and their relationships. Moscato et al. [36] explained that different cloud systems and vendors have different ways of describing their services, to specify requirements and to communicate. Therefore, the definition of a CM ontology allows us to overcome this problem because it represents the common concepts suitable to be used by the different cloud providers.
- Ontologies automate the discovery services by performing a semantic mapping between the user query and the described services because they use the inference mechanism. Tsai et al. [37] argued that the reasoning abilities provided by the ontology systems in oriented-service frameworks can facilitate the service matching process, provide a certain level of flexibility by returning the most compatible services when a perfect match cannot be found, and reduce the manual work for a user.
- Inference ontologies treat non canonic data, which allows having individuals in different classes at the same time. Therefore, it can merge different viewpoints of the same object [38]. It is a very important advantage of inference ontologies in our study: let the individual X be a design software proposed by a provider, and let the concepts "DesignaaS" (design as a service), "Saas" (software as a service) be in the ontology. X must belong to the two classes so the mediator returns this service as responsive if the user employs one of these terms in his request.

Typesetting conventions used in this paper are: **Class**, individual and *property*. The top level of the ontology that constitutes the generic knowledge model (Fig. 4) is composed of **Actor** class, which represents the actors that interact with the Coud Manufacturing system [23]. **Deployment Model** for the Coud Manufacturing mode of deployment this class contains the following instances: Community, Hybrid, Private, Public. **Semantic_elements** class contains all the elements needed to allow and facilitate the communication with the service and its use. It consists of: **Data_language**, **Programming_language**, **Protocol**}. **SLA** class includes all the elements needed to define the SLA. **State** describes the state of the service and contains the following individuals: free, unavailable, used. **Property** class is divided into two subclasses: **functional_property** and **Non_functional_property**. The **functional_property** class describes the elements needed to run a service and **Non_functional_property** class describes properties that will be considered, in addition to functional ones, to decide which service is most suited for a particular user such as **Availability**, **Qos**, **reliability**. **Resource** class describes the resources that will be packaged as a service and contains two subclasses **Hard_resource** that combines **Manufacturing_equipment** and **Computational_resources**, and **Soft_resource** that include non material resources like **Software**, **Experience**, **Skill**. **Service** is the cloud system's centric entity. In cloud manufacturing, all of the resources are virtualized and packaged as a service. The **Service** class contains subclasses **PLMaas**, **ServiceModel**. The **PLMaas** class includes PLM stages like: design as a service (**DaaS**), manufacturing as a service (**MFGaas**), simulation as a service (**SIMaas**), etc. The **ServiceModel** class describes the service's categories: **Saas**, **Paas**, **Iaas**, **Holonaas**, where the latter is used to describe the services that encompass manufacturing resources. Indeed, according to the holon architecture introduced by Bussmann [39] we notice that a holon has a "logical" part that allows for control and communication with the physical part. We believe that this definition is suitable for the "manufacturing" part of cloud manufacturing since our aim is to make physical entities interact with other services and be remotely accessible.

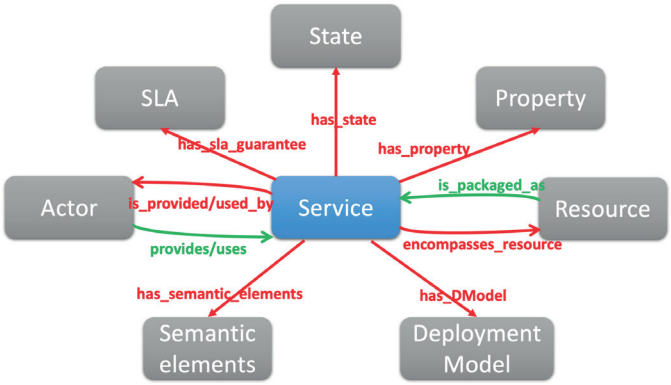

Fig. 4. The top-level concepts and their semantic links

Since the service is the most important entity that maps the users and the providers, the model is service-centric (Fig. 4). The **Actor** individuals can either provide or consume a **Service**. The **State** individuals specify the state of the **Service**, whether it is used, free or in maintenance. The **Property** class and subclasses define the functional and non-functional characteristics of the **Service**. The **Deployment_model** addresses the deployment mode of the Cloud (public, private or hybrid) where the **Service** is provided. **Semantic_elements** specifies the means of communication with the **Service** such as the data format handled and generated. Resources belong to an **Actor** (enterprise for instance) and are packaged and virtualized as a **Service**.

In addition to semantic relationships between concepts, the CM ontology also includes SWRL (Semantic Web Rules Language) rules. SWRL is a language that completes the expressivity of OWL with rules based on the RuleML paradigm [40]. These rules make explicit the implicit links between instances using logical axioms. In our case, between a resource, a service, and a supplier:

$$is\_owned\_by(?y, ?x), is\_packaged\_as(?y, ?z) \rightarrow$$
$$\rightarrow is\_provided\_by\ (?z, ?x).$$

This rule means that if a resource "y" belongs to a supplier "x", and the resource "y" is virtualized as a service "z" then service "z" is provided by "x" (To view the details of the complete ontology, see [41]).

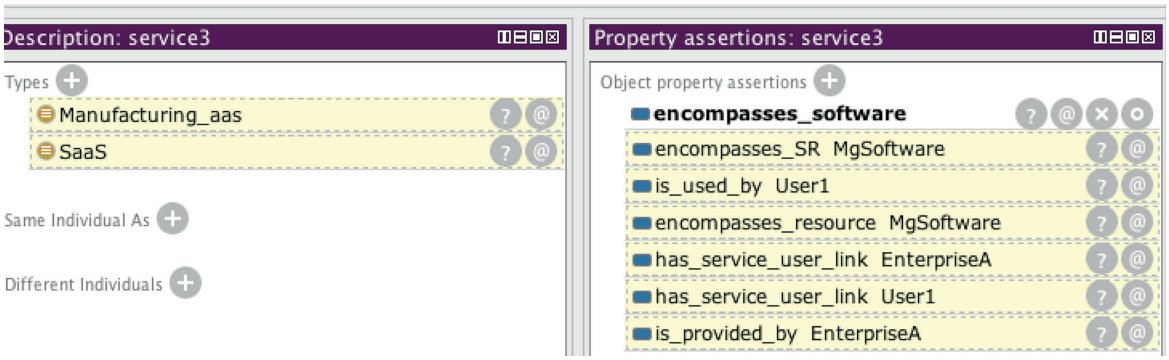

Fig. 5. Inference results on individual Service3

3.3. Model validation. Our generic knowledge model is validated by populating the ontology with a number of instances and performing an inference mechanism on them. This mechanism allows us to infer implicit information with only some explicit knowledge. This is one motivation for choosing ontologies since this mechanism allows for the reduction of the modelling commitment and therefore saves time and facilitates future model integration. In the following section, we present one of the unit tests performed on the ontology.

We have defined an **Enterprise** type individual named EnterpriseA, a software individual: Mgsoftware that is **Manufacturing_software**. We have also defined individuals service3 and User1 without specifying their types in order to verify whether the inference mechanism will figure out whether they are **service** or **Consumer** types, respectively, to validate the CMO. The individuals are defined as follows:

- EnterpriseA *owns_resource* Mgsoftware
- User1 *use_cm_service* service3
- service3 *encompasses_software* Mgsoftware

We have also used the rule presented in Section 3.2 and have defined two classes as:

- Consumer ≡ **Actor** and (*use_cm_service* some **Service**)
- **Manufacturing_aas** ≡ **Service** and (*encompasses_resource* some **Manufacturing_resource**)

The information has been inferred using the definitions above. For example, for the individual service3, we have only specified that it *encompasses_software* Mgsoftware and we see from Fig. 5 that the information inferred is:

- service3 ∈ **Manufacturing_aas**
- service3 ∈ **Saas**
- service3 *is_used_by* User1
- service3 *encompasses_resource* MgSoftware
- service3 *is_provided_by* EnterpriseA

Complete validation of our ontology as well as a complete review of CM ontologies are presented in [41]. Next, an example of design and implementation is illustrated with an action model based on CloudSim simulation software. It corresponds to the steps in the bottom left-hand corner of Fig. 3. The next sub-sections follow these steps.

4. Methodology validation using a literature scenario

This section describes a proof-of-concept scenario to explain how to implement CM architecture using the steps presented in the methodology shown above.

4.1. Design phase. In this paper, we present an example based on CloudSim simulation software. To do that, we use the component library that can be reused in different systems within the ASDI framework.

The UML language is an international reference of modelling software used by the builder of CloudSim. Moreover, the existing technical tools do not allow the correct implementation of OWL-based models. In fact, the tools of the ontology are not mature enough to allow full implementation including the reasoning part. Hence, we translate the static elements of the CMO to UML. The translated classes that form the CMO are shown in UML language in Fig. 6. This is just an extract and not all details are shown. The translation is based on the work of Atkinson and Kiko [42] who have made a detailed comparison of UML and OWL. The classes with boxes around them are integrated into the CloudSim environment.

4.2. Implementation phase. Our aim is to create a library of reusable components. We present the architecture we considered in Fig. 7. In this architecture, the UML model derived by the CMO is integrated with the CloudSim tool. Initially, CloudSim was developed in order to simulate CC; we propose using it in our architecture.

The CMO is the main element in the architecture because it is used to match the users with the providers. The first step is the publication of existing services. The communication process starts with action from the providers: they register and publish the services they wish to present. Processing the user's request consists of a set of steps:

1. Service request: the user sends a request to the module that holds the knowledge base. This base contains all of the information about available services and it implements the CMO as a knowledge base;
2. CMO response: users are sent the response to the request with a list of appropriate services;

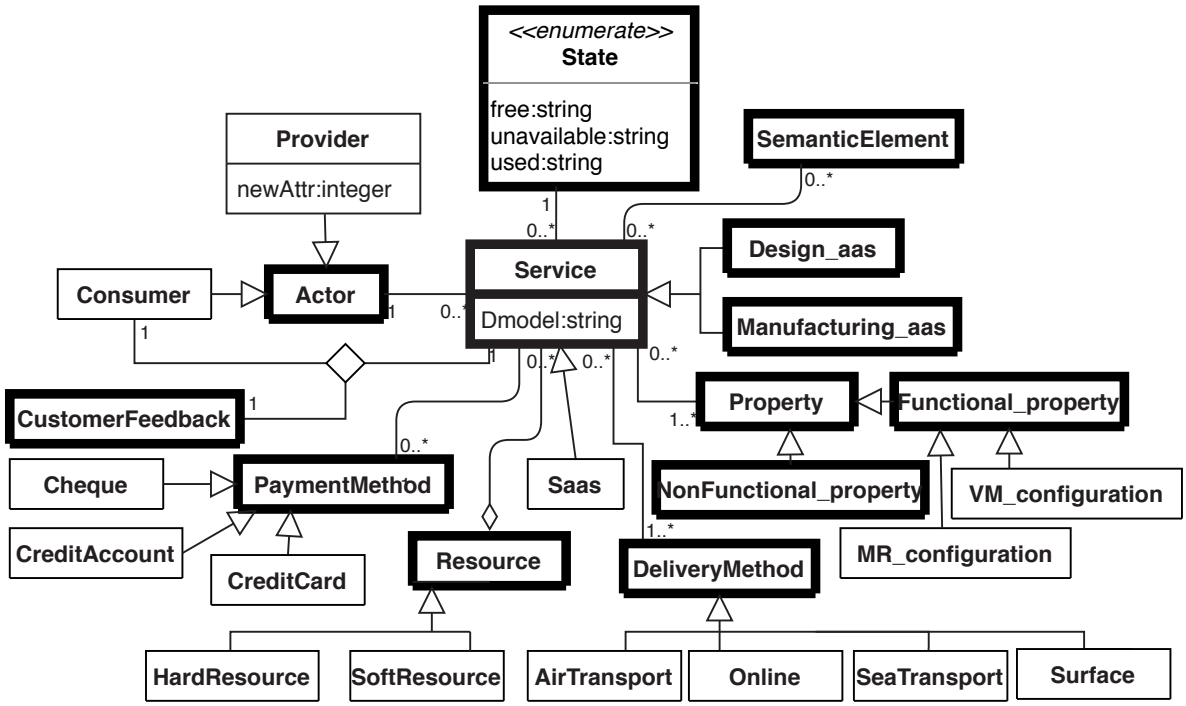

Fig. 6. Class integrated in CloudSim environment

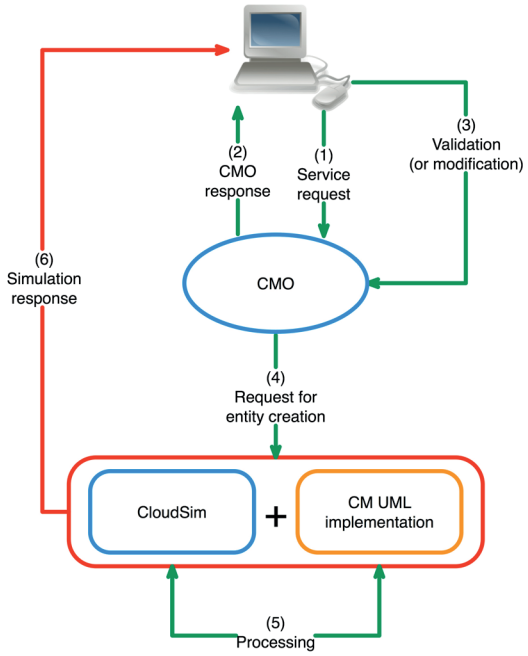

Fig. 7. The architecture considered

3. Validation: the user can choose among the list of services or decide to modify the request;
4. Request for entity creation: after confirmation by the user, a request for entity creation (virtual machine, machine-tool simulation, etc.) is sent;
5. Processing: the entity is created and the service execution starts;
6. Simulation response: the results of the process are sent to the user.

The CMO module is detailed in Fig. 8. It is divided in two parts:

1. The knowledge base represents the domain: services, providers, users, etc. It is built using XML files that describe all the entities with respect to the UML model.
2. The reasoner, which is the implementation of the reasoning part of the CMO, (i) acts like a search engine that links the information from the users, (ii) parses the knowledge base to find the appropriate services and sends a request to the simulator to create the requested entities.

5. Case study of a particular system

5.1. Presentation. This example consists of the use of the implemented classes shown in the previous section. The goal is to represent a situation in which the user sends a request to choose the service needed among different possibilities. The services proposed are described in an XML file. In order to do this simulation, we have hosted data in local servers, including WAMP. WampServer is a Windows web development environment that allows web applications to be created with Apache2, PHP and a MySQL database. Along with this, PhpMyAdmin enables one to easily manage databases. Figure 9 shows the use case presented in this paper and its relationship with the multi-layer framework of CM already presented in the introduction. This means that the *Company-X* provides services *S-A2* and *S-A3*, *EntrepriseA* provides *service2* and *EnterpriseB* provides *Service3* and *S-A1*. The services are described using an

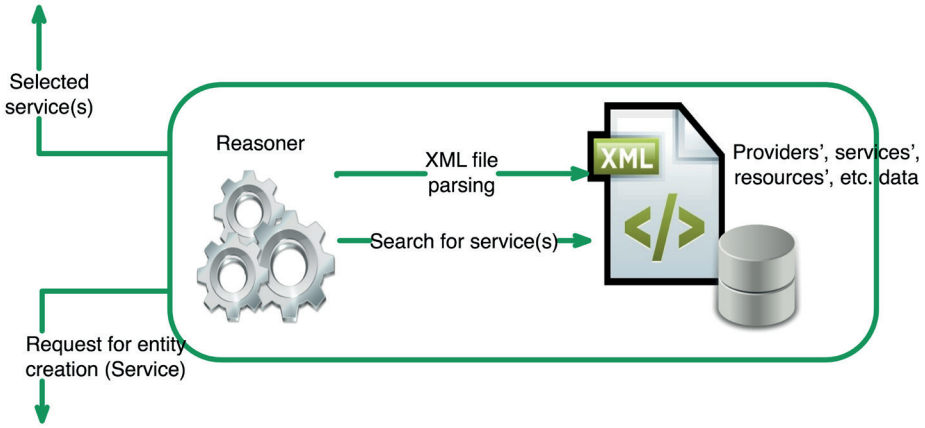

Fig. 8. Detail of the CMO module from Fig. 7

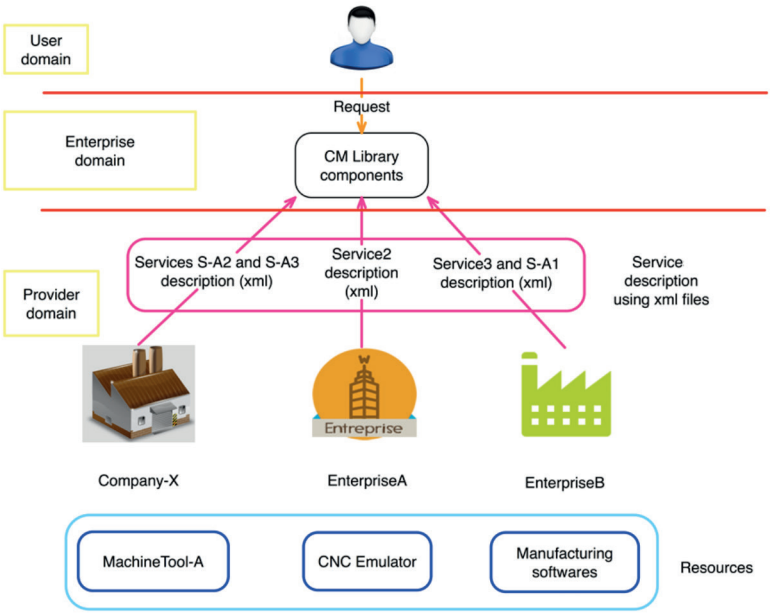

Fig. 9. Use case and the reference architecture

XML file and presented in Fig. 11. For instance, we described *service2* in the XML file as a Holon as a service (**Holonaas**) because it represents a CNC emulator with:

- a logical part: the interface developed using HTML5 for accessing the machine-tool;
- a physical part: the emulator itself.

The next sub-sections follow the steps to the right of Fig. 3, which represent the entire methodology.

5.2. Knowledge model. We present the utilization of the methodology on a particular CM system with the simulation software CloudSim. The knowledge model used is based on CM ontology. We identify the elements of the ontology that will be used in the simulation model. Figure 10 shows the identified elements from the CM ontology. It is an extract from the whole CMO. It also explicitly depicts the semantic links between the top-level concepts. The model is service-centric since the service is the most important entity that maps the users and the providers.

5.3. Action model. Several action models can be deduced starting from the same knowledge model. Here, we used a computer numerical control (CNC) emulator developed by William HILTON during his thesis[3] in Drexel, Philadelphia, USA (emulation of a DYNA MYTE 2400 machine). We installed a MAMP[4] server to host the service, which represents a CNC machine. MAMP is an acronym for Mac OS X, the operating system; Apache, the web server; MySQL, the database management system; and PHP, Perl, or Python, all programming languages used for web development.

[3] https://sites.google.com/site/wmhilton/projects/javascript-cnc-machine (available June, 2019)

[4] https://www.mamp.info/en/ (available June, 2019)

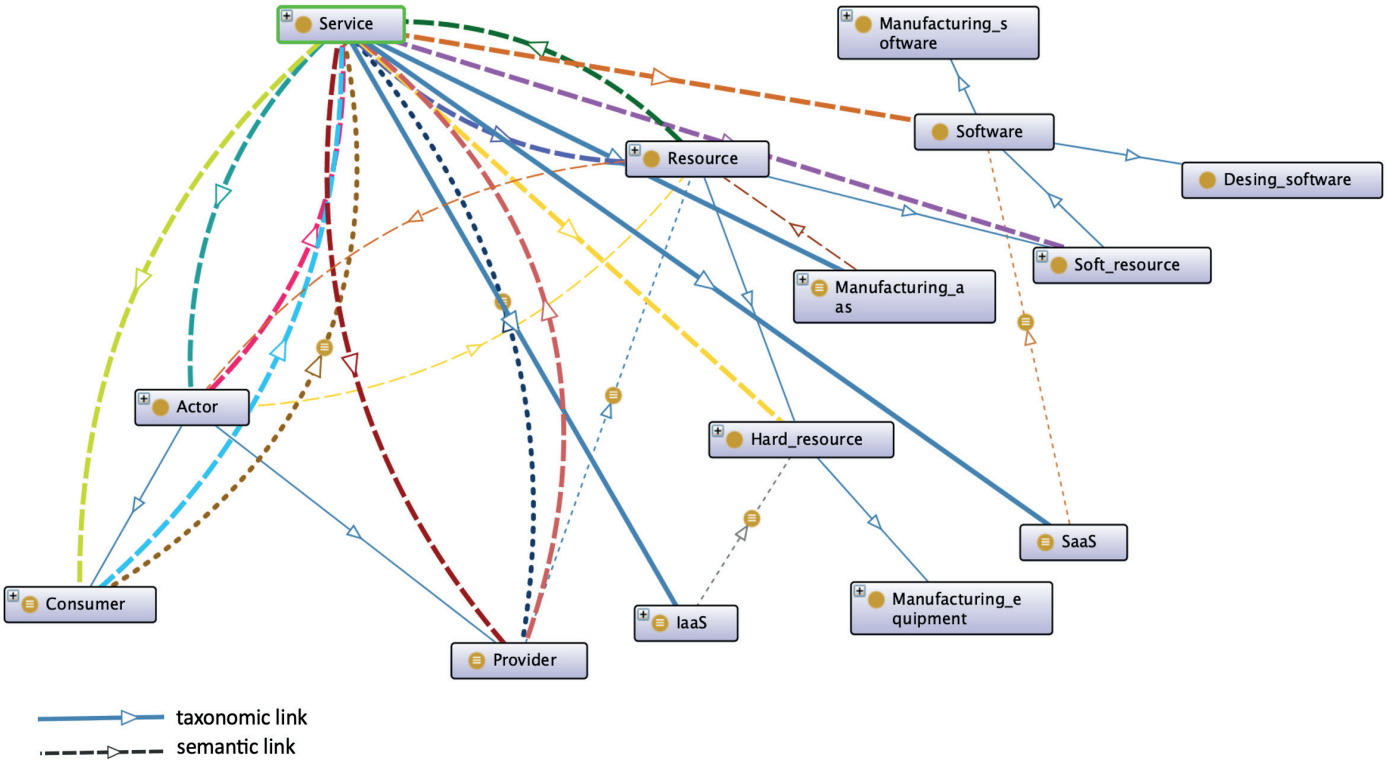

Fig. 10. Knowledge model elaborated from CMO

The CM classes and the reasoner are developed using Java language. They parse the XML file which contains the description of services (Fig. 11) and for each service displayed the name, the type and the name of the provider. The next step is up to the user; here, the user can choose *service 2*. In order to simulate the machine tool and to illustrate the function of *service 2* we used a CNC simulator developed by William HILTON[5]. Figure 12 represents the *service 2* selected by the user in the example which uses the class **Holonaas**. The interface shown in the figure depicts the logical part of the *Holon*. The web page is written in HTML5 and JavaScript.

5.4. Result model. Figure 13 depicts the results after performing the simulation on our use case. This means the result model is fed by the action model. In the result model, we can analyse the performance of the action model. Here, it is made with the results of the simulation. For example, the service used to perform the case study produces a log file where each line represents the time spent with the machine tool to make the different movements for the milling cutter and the coordinates of the movements. At this step, an evaluation of the action model has to be done (analysis of the data). Here, the most important data is the time spent by the CNC machine to make the movement. After interpreting the result model, different actions on the system can be made. Thanks to the result model, the user has a few different choices:

services.xml

```
<Individuals>
    <Individual uid="service2">          Name of the resource
        <Type>IaaS</Type>
        <Type>Holonaas</Type>
        <Type>Service</Type>             Types and semantic links of the resource
        <Resource>resource1</Resource>
        <Provider>EnterpriseA</Provider>
        <uri>http://localhost:8888/CNC/CNC_Simulator.php</uri>
    </Individual>                        URI for identifying the resource

    <Individual uid="service3">
        <Type>SaaS</Type>
        <Type>Manufacturing_aas</Type>
        <Type>Service</Type>
        <Resource>MgSoftware</Resource>
        <Provider>EnterpriseB</Provider>
        <uri>uri2</uri>
    </Individual>

    <Individual uid="S-A1">
        <Type>IaaS</Type>
        <Type>Manufacturing_aas</Type>
        <Type>Service</Type>
        <Resource>MgSoftware</Resource>
        <Provider>EnterpriseB</Provider>
        <uri>uri3</uri>
    </Individual>

    <Individual uid="S-A2">
        <Type>IaaS</Type>
        <Type>Manufacturing_aas</Type>
        <Type>Service</Type>
        <Resource>machineTool-A</Resource>
        <Provider>company-X</Provider>
        <uri>uri4</uri>
    </Individual>

</Individuals>
```

Fig. 11. List of services in the Use Case

[5] https://sites.google.com/site/wmhilton/home (available June, 2019)

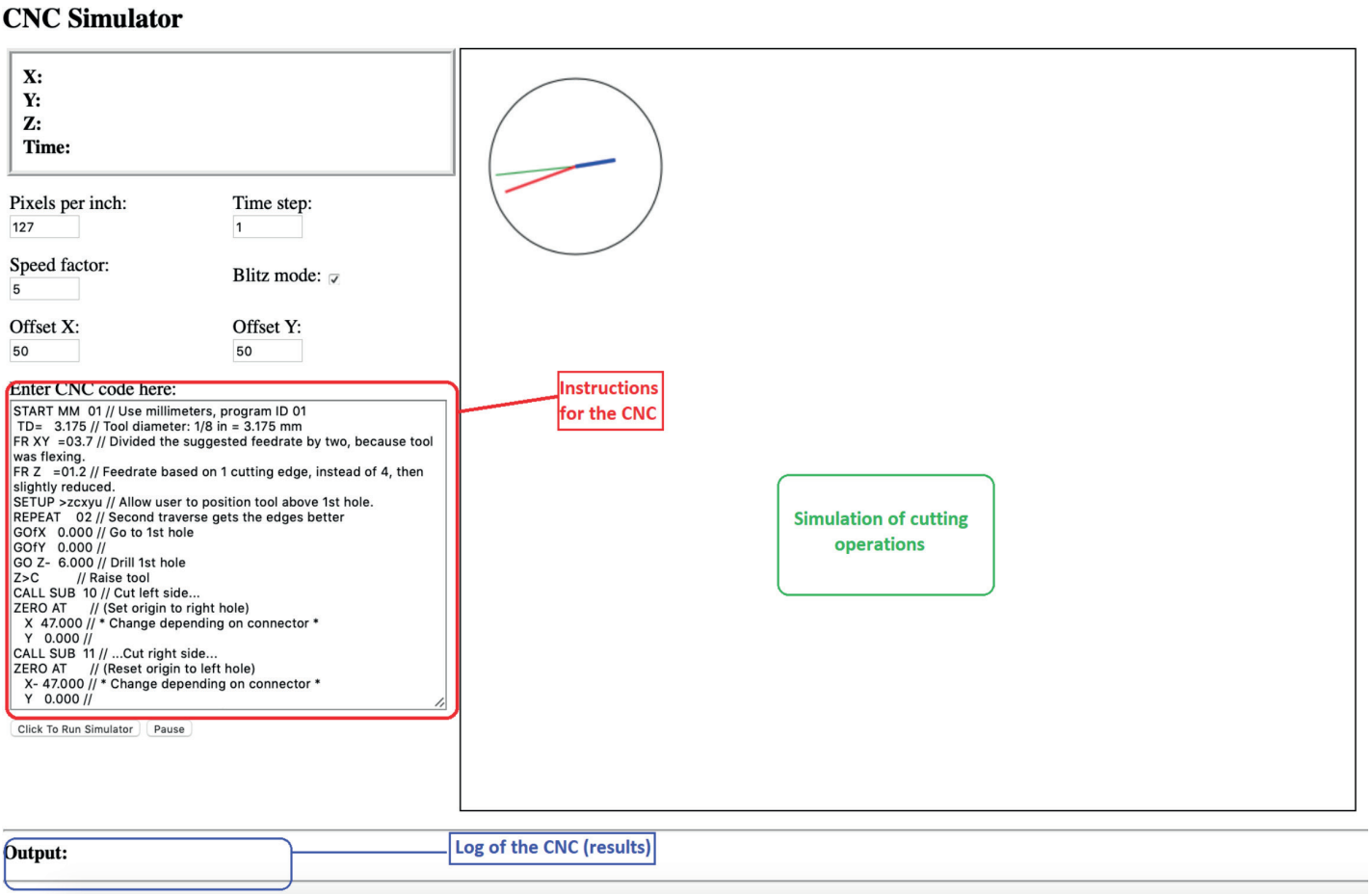

Fig. 12. Logical part of the holon – CNC simulator

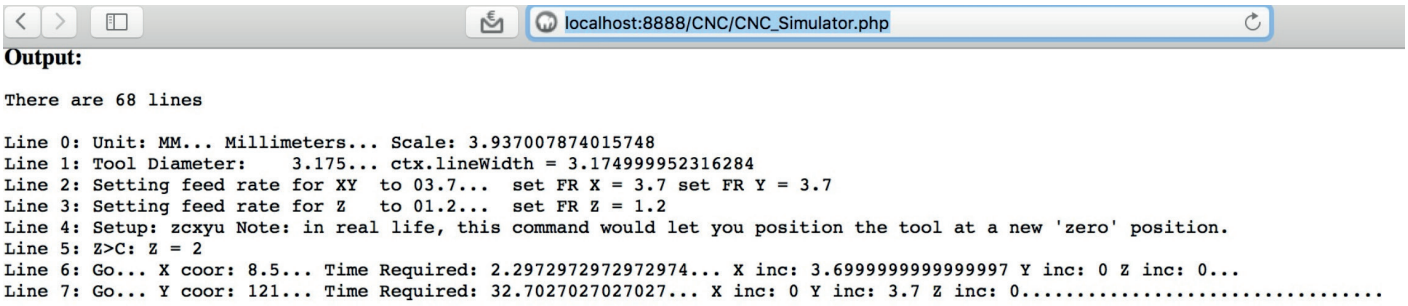

Fig. 13. Log of the CNC – result model

1. maintain the settings of the machine and therefore continue to use this service.
2. modify the settings of the machine in order to obtain a better execution time.
3. select another service.

A major change that can be made by the user is to modify the system that was studied, and, consequently, elaborate upon another knowledge model and generate another action model.

6. Discussion and perspectives

6.1. Proposal improvement and implementation limits. We studied the literature on CM to identify the concepts of the ontology proposed. Nevertheless, it remains non exhaustive and other concepts could be added to extend the model. In our case, the ontology is used as a mediator between suppliers and customers in order to provide decision making support. At this stage, the user already knows the workflow to run and use

our example to find services that will fit their needs. For this reason, the ontology does not include concepts such as process and workflow.

On the other hand, as was explained, ontology tools are not mature enough to enable a full implementation, including reasoning. Likewise, the number of responses returned by the reasoner depends on the number of entities in the system. Since CM environments contain millions of instances, it is impossible for the user to choose between thousands of services. This limitation of the model leads us to consider methods from the field of operation research to implement the reasoning. Indeed, these methods and algorithms could optimize the responses under users' constraints, such as: costs, processing time, etc.

Additionally, the security of hosted data is a crucial concern of customers and suppliers. Customers are concerned about confidentiality, backups and restoring their data and suppliers with regards to VM isolation methods and data protection and recovery procedures. Indeed, Främling et al. [43] explained that one of the challenges in network systems is where the data is stored and how to access and update it.

6.2. Application to a real use case. The ontology has been validated by performing unit testing and integration testing. Although the results remain positive, an application in a real industrial scenario provides more definitive validation by evaluating the model in cloud-based environments, which include a growing number of highly connected entities. These inter-connections can be illustrated by the fact that one user is connected to various suppliers via a set of services. With this increasing volume of information, the validation of the ontology involves application in a number of scenarios that are not easy to imagine. In our case, this has led to a modification of the CM architecture and to the creation of a new one thanks the CMO, in order to use a real CM architecture without a simulation. Another potential route could be the use of this approach to model another domain with ontology and OWL.

6.3. Licenses and user rights. A company needs to know what software or other infrastructure they have access to, how it is used and how it will be impacted by the process of migrating it to the cloud. This allows businesses to manage their assets and have strong knowledge and strict policies for access control in order to supervise the information system and avoid vulnerabilities. On the other hand, from the suppliers' perspective, the software licenses can create misunderstanding. In fact, for some software vendors, the software license is specific to a given company and cannot be purchased on behalf of another one. In this case, suppliers do not have the right to purchase such licenses in order to provide a SaaS offer.

6.4. Moving towards green cloud manufacturing. James Glanz from "The New York Times" estimates that "the industry uses 30 billion watts of electricity, roughly equivalent to 30 nuclear power plants". It is therefore imperative to find new ways and methods to optimize the energy requirements of a data centre. Power consumption is the cause of many problems as a result of excessive heat. Service providers in CC are focused on methods of VM isolation and data privacy neglects power consumption management. Service providers may take action and ensure that their profit margin is not reduced because of the high cost of electric consumption. Some studies have been conducted on this subject: Huet and El Abbassi [19] used ASDI framework in order to build a software environment for green cloud computing, Lu et al. [44] proposed an energy management architecture in the Industry 4.0 area.

6.5. Coud manufacturing and edge computing. The distance between cloud platforms and the end devices might be an issue for latency sensitive applications such as content delivery applications. The idea of the edge and fog computing is to place small servers near the end users [45]. In this way, some of the computational and data storage load is transferred from a cloud platform to the edge servers. The use of this kind of service could be a powerful complement to the Coud Manufacturing architecture. In this case, a part of the ontology would be integrated in the edge level.

7. Conclusion

This paper presents a new methodology for implementing a cloud manufacturing architecture. Coud manufacturing is one of the aspects of the Industry 4.0 trend. This methodology is inspired by ASDI framework (analysis-specification-design-implementation). ASDI is based on two major aspects: the study of a domain and the study of a system of this domain.

In order to study a domain, we used a generic knowledge model based on ontologies. With the inference mechanism offered by ontologies, we showed that the reasoner has inferred implicit information with less knowledge. At this level, the choice of ontology helps us build a generic knowledge model that depicts CM concepts and the relationships between them, taking into account the semantic aspect of this approach. The other advantage of ontologies and semantic web is to allow the communication between the different stages of the product lifecycle.

A study of a system for the domain is explained in this paper. An action model that represents the system studied is implemented with cloud computing simulation software: CloudSim. We modified this well-known software in order to adapt it to the cloud manufacturing paradigm. To do so, we followed the steps of the ASDI methodology. We then created a library of reusable components. In our case, this library represents the CM architecture components that will be used to implement a CM simulation system based on CloudSim. The advantages and benefits of our methodology are:

- propose a reusable framework to build CM platforms to map CM users and providers in such a way that a user can choose between a set of services and select the ones that will meet his or her needs. This platform will help to enhance collaboration in PLM by reducing costs and development time,

- contribute to the technological levers of industry 4.0. Indeed, this methodology explains the implementation of a platform allowing the connection of different actors collaborating on product development and thus poses a generic framework that can be applied to any company, any domain, wishing to implement a cloud manufacturing and the underlying architecture,
- have a global approach in order to provide a guide for users to migrate their systems towards the cloud.

This methodology will be further validated by applying it to a real case scenario in order to help industrial decision-makers in the process of moving towards cloud-based solutions. However, an example of a case study setting up a cloud platform has been implemented in the oil and gas industry in U.S. firms [46] and it highlights the emergence and importance of such a solution.

A. Acronym glossary

ASDI	analysis-specification-design-implementation
CC	cloud computing
CM	coud manufacturing
CMO	coud manufacturing ontology
CNC	computer numerical control
DAAS	design as a service
DSS	decision-making subsystem
GSL	global service layer
GUI	graphical user interface
HMS	holonic manufacturing system
ICT	information and communication technology
INTaas	integration as a service
Iot	internet of thing
Lss	logical subsystem
MAaas	maintain as a service
Maas	management as a service
MDCsim	multi-tier data center simulation
MFGaas	manufacturing as a service
MRL	manufacturing resource layer
PLM	product lifecycle management
PSS	physical subsystem
SDN	software defined network
SOA	service oriented architecture
SIMaas	simulation as a service
SWRL	semantic web rules language
UD	user domain
UML	unified modeling language
VM	virtual machine
VSL	virtual resource layer

References

[1] Y. Lu, "Industry 4.0: A survey on technologies, applications and open research issues," *J. Ind. Inf. Integr.*, 6, 1–10 (2017). https://doi.org/10.1016/j.jii.2017.04.005

[2] X. Xu, "From Coud Computing to Coud Manufacturing," *Rob. Comput. Integr. Manuf.*, 28(1), 75–86 (2012).

[3] S. Terzi, "Elements of Product Lifecycle Management: Definitions, Open Issues and Reference Models," doctoral thesis, Université Henri Poincaré-Nancy I, May 2005. https://tel.archives-ouvertes.fr/tel-00009559

[4] O. Cardin, "Classification of cyber-physical production systems applications: Proposition of an analysis framework," *Comput. Ind.*, 104, 11–21 (2019). https://doi.org/10.1016/j.compind.2018.10.002

[5] Z. Zhang, X. Wang, X. Zhu, Q. Cao, and F. Tao, "Cloud manufacturing paradigm with ubiquitous robotic system for product customization," *Rob. Comput. Integr. Manuf.*, 60, 12–22 (2019). https://doi.org/10.1016/j.rcim.2019.05.015

[6] M. Zaborowski, "Data processing in self-controlling enterprise processes," *Bull. Pol. Ac.: Tech.*, 67(1), 3–20 (2019).

[7] D. Mazur, A. Paszkiewicz, M. Bolanowski, B.G., and M. Oleksy, "Analysis of possible SDN use in the rapid proto- typing process as part of the industry 4.0," *Bull. Pol. Ac.: Tech.*, 67 (1), 21–30 (2019).

[8] L. D. Xu, E. L. Xu, and L. Li, "Industry 4.0: state of the art and future trends," *Int. J. Prod. Res.*, 56 (8), 2941–2962 (2018). https://doi.org/10.1080/00207543.2018.1444806

[9] H. Van Brussel, J. Wyns, P. Valckenaers, L. Bongaerts, and P. Peeters, "Reference architecture for holonic manufacturing systems: Prosa," *Comput. Ind.*, 37 (3), 255–274 (1998).

[10] A. Koestler *et al.*, *The ghost in the machine*. Hutchinson London, 1967.

[11] J.-C. Huet, J.-L. Paris, K. Kouiss, and M. Gourgand, "A new reengineering methodology for the product-driven system applied to the medication-use process," *Decis. Support Syst.*, 55(2), 599–615 (2013), 1. Analytics and Modeling for Better Health Care 2. Decision Making in Healthcare.

[12] F. Tao, L. Zhang, V. Venkatesh, Y. Luo, and Y. Cheng, "Cloud manufacturing: a computing and service-oriented manufacturing model," *J. Engineering Manufacture*, 225(10), 1969–1976 (2011).

[13] C. Danjou, L. Rivest, and R. Pellerin, "Douze positionnements stratégiques pour l'industrie 4.0: entre processus, produit et service, de la surveillance à l'autonomie (twelve strategic positioning for industry 4.0: between process, product and service, from monitoring to autonomy.)," in *CIGI*, 2017.

[14] M. Rüßmann, M. Lorenz, P. Gerbert, M. Waldner, J. Justus, P. Engel, and M. Harnisch, "Industry 4.0: The future of productivity and growth in manufacturing industries," *Boston Consulting Group*, 9(1), 54–89 (2015).

[15] G. Zellner, "A structured evaluation of business process improvement approaches," *Bus. Process Manag. J.*, 17(2), 203–237 (2011).

[16] M. Gourgand and P. Kellert, "An object-oriented methodology for manufacturing system modelling," in *Summer Computer Simulation Conference*, Reno, Nevada, USA, July 1992, pp. 1123–1128.

[17] M. Chabrol, D. Sarramia, and N. Tchernev, "Urban traffic systems modelling methodology," *Int. J. Prod. Econ.*, 99, 156–176 (2006).

[18] H.E. Haouzi, A. Thomas, and J. Pétin, "Contribution to reusability and modularity of manufacturing systems simulation models: Application to distributed control simulation within {DFT} context," *Int. J. Prod. Econ.*, 112(1), 48–61 (2008).

[19] J.-C. Huet and I. El Abbassi, "Green Coud Computing modelling methodology," in *Utility and Cloud Computing (UCC), 2013 IEEE/ACM 6th International Conference on*, Dec 2013, pp. 339–344.

[20] A. Ahlam, "Construction of scheduling methods and tools for the phosphate mining industry in the context of Lean Management," doctoral thesis, Université Paris 10, December 2018. https://www.theses.fr/s148905

[21] E.J. Ghomi, A.M. Rahmani, and N.N. Qader, "Cloud manufacturing: challenges, recent advances, open research issues, and future trends," *Int. J. Adv. Manuf. Technol.*, 102 (9), 3613–3639, (2019). https://doi.org/10.1007/s00170–019–03398–7

[22] A. Talhi, J.-C. Huet, V. Fortineau, and S. Lamouri, "Toward an ontology-based architecture for Coud Manufacturing," in *Service Orientation in Holonic and Multi-agent Manufacturing*, ser. Studies in Computational Intelligence, vol. 594, pp. 187–195 T. Borangiu, Thomas, and D. Trentesaux, Eds. Springer International Publishing, 2015.

[23] A. Talhi, J. Huet, V. Fortineau, and S. Lamouri, "Towards a Coud Manufacturing systems modeling methodology," in *15th IFAC Symposium on Information Control Problems in Manufacturing*, 48 (3), 288–293 (2015). https://doi.org/10.1016/j.ifacol.2015.06.096

[24] X.V. Wang and X.W. Xu, "An interoperable solution for Coud Manufacturing," *Rob. Comput. Integr. Manuf.*, 29 (4), 232–247 (2013).

[25] Y. Lu, X. Xu, and J. Xu, "Development of a hybrid manufacturing cloud," *J. Manuf. Syst.*, 33(4), 551–566 (2014).

[26] T.Y. Lin, C. Yang, C. Zhuang, Y. Xiao, F. Tao, G. Shi, and C. Geng, "Multi-centric management and optimized allocation of manufacturing resource and capability in Coud Manufacturing system," *J. Engineering Manufacture*, 231(12), 2159–2172 (2016). https://doi.org/10.1177/0954405415624364

[27] H. Bouzary and F.F. Chen, "Service optimal selection and composition in Coud Manufacturing: a comprehensive survey," *Int. J. Adv. Manuf. Technol.*, 97, 795–808 (2018).

[28] Y. Lu and X. Xu, "A semantic web-based framework for service composition in a Coud Manufacturing environment," *J. Manuf. Syst.*, 42, 69–81 (2017). https://doi.org/10.1016/j.jmsy.2016.11.004

[29] J. Son, A.V. Dastjerdi, R.N. Calheiros, X. Ji, Y. Yoon, and R. Buyya, "Cloudsimsdn: Modeling and simulation of software-defined cloud data centers," in *Proceedings of the 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid 2015)*, Shenzhen, China, May 2015.

[30] D. Kliazovich, P. Bouvry, and S.U. Khan, "GreenCloud: a packet-level simulator of energy-aware Coud Computing data centers," *J. Supercomput.*, 62 (3), 1263–1283 (2012).

[31] S.-H. Lim, B. Sharma, G. Nam, E.K. Kim, and C.R. Das, "MDCSim: a multi-tier data center simulation, platform," in *Cluster Computing and Workshops, 2009. CLUSTER'09. IEEE International Conference on*. IEEE, 2009, pp. 1–9.

[32] R.N. Calheiros, R. Ranjan, A. Beloglazov, C.A. De Rose, and R. Buyya, "CloudSim: a toolkit for modeling and simulation of Coud Computing environments and evaluation of resource provisioning algorithms," *Software. Pract. Exper.*, 41(1), 23–50 (2011).

[33] E. Barbierato, M. Gribaudo, M. Iacono, and A. Jakóbik, "Exploiting cloudsim in a multiformalism modeling approach for cloud based systems," Modeling and Simulation of Cloud Computing and Big Data. *Simul. Model. Pract. Theory*, 93, 133–147 (2019), https://doi.org/10.1016/j.simpat.2018.09.018

[34] K. Ma, A. Bagula, C. Nyirenda, and O. Ajayi, "An iot-based fog computing model," *Sensors*, 19 (12), 2783 (2019). https://doi.org/10.3390/s19122783

[35] H.A. Simon, "The architecture of complexity," *Proceedings of the American Philosophical Society*, 106(6), 467–482, (1962). http://www.jstor.org/stable/985254

[36] F. Moscato, R. Aversa, B. Di Martino, T. Fortis, and V. Munteanu, "An analysis of mosaic ontology for cloud resources annotation," in *Computer Science and Information Systems (FedCSIS), 2011 Federated Conference on*, Sept 2011, pp. 973–980.

[37] W. Tsai, X. Sun, Q. Huang, and H. Karatza, "An ontology-based collaborative service-oriented simulation framework with microsoft robotics studio®," *Simul. Model. Pract. Theory*, 16(9), 1392–1414 (2008).

[38] V. Fortineau, T. Paviot, and S. Lamouri, "Improving the interoperability of industrial information systems with description logic-based models—the state of the art," *Comput. Ind.*, 64(4), 363–375 (2013).

[39] S. Bussmann, "An agent-oriented architecture for holonic manufacturing control," in *Proceedings of First International Workshop on IMS, 1998*, 1998. http://www.stefan-bussmann.de/files/ims98.ps

[40] S.V. Fortineau, X. Fiorentini, T. Paviot, L. Louis-Sidney, and Lamouri, "Expressing formal rules within ontology-based models using swrl: an application to the nuclear industry," *Int. J. Prod. Lifecycle. Manag.*, 7(1), 75–93 (2014).

[41] A. Talhi, V. Fortineau, J.-C. Huet, and S. Lamouri, "Ontology for Coud Manufacturing based product lifecycle management," *J. Intell. Manuf.*, 30(5), 2171– 2192 (2019). https://doi.org/10.1007/s10845–017–1376–5

[42] C. Atkinson and K. Kiko, "A detailed comparison of uml and owl," *Report from University of Mannheim*, 2008. https://ub-madoc.bib.uni-mannheim.de/1898/

[43] K. Främling, T. Ala-Risku, M. Kärkkäinen, and J. Holmström, "Design patterns for managing product life cycle information," *Communications of the ACM*, 50(6), 75–79 (2007).

[44] Y. Lu, T. Peng, and X. Xu, "Energy-efficient cyber- physical production network: Architecture and technologies," *Comput. Ind. Eng.*, 129, 56–66 (2019). https://doi.org/10.1016/j.cie.2019.01.025

[45] K. Bilal, O. Khalid, A. Erbad, and S.U. Khan, "Potentials, trends, and prospects in edge technologies: Fog, cloudlet, mobile edge, and micro data centers," *Comput. Networks*, 130, 94–120 (2018). https://doi.org/10.1016/j.comnet.2017.10.002

[46] V. Jain and B. Kolla, "Case study: Could costing discoveries for the end to end solution," in *COLLABORATE 19, Technology and Applications Forum for the Oracle Community*, 2019.