Meta-Model Approach of Applied Devops on Internet of Things Ecosystem

Badr El Khalyly, Abdessamad Belangour, Allae Erraissi, Mouad Banane
Laboratory of Information Technology and Modeling
Hassan II University, Faculty of sciences Ben M'sik, Casablanca, Morocco
Emails: badrelkhalyly92@gmail.com, belangour@gmail.com, allae.erraissi-etu@etu.univh2c.ma, mouad.banane-etu@etu.univh2c.ma

2020 IEEE 2nd International Conference on Electronics, Control, Optimization and Computer Science (ICECOCS) | 978-1-7281-6921-7/20/$31.00 ©2020 IEEE | DOI: 10.1109/ICECOCS50124.2020.9314552

***Abstract*— DevOps is a software development and distribution process. It combines an entire culture of collaboration and fusion between management approach, technologies of coding and technics of integration. DevOps is a new paradigm and novel concept that is introduced in different industries that concerns software and embedded development like robotics and smart agent.The ecosystem of the Internet of Things is a set of physical devices such as sensors and actuators. It includes a set of servers and gateways that provide connectivity. These devices can be installed at three levels: the edge level, the fog level and the cloud level. Applications that monitor things and collect data from sensors are deployed at the edge, fog and cloud levels. These applications can be containerized and deployed in different devices by Docker, which takes advantage of its advantage to create executable containers that are isolated from each other. This paper presents a generic metamodel of the Internet of Things ecosystem based on microservices and supported by Docker as a containerization tool and Ansible as a monitoring tool. This metamodel allows us to generate a system of connected objects that can be deployed on three levels: Fog, Edge, Cloud. This metamodel can be used in different types of domains such as Smart Home - Smart City - Smart Vehicle - Smart Health - Smart Farm - Smart Factory. The metamodel proposed in this article is a fusion between 5 metamodels: Internet of Things - Microservices - Ansible – Docker - Kubernetes.**

Keywords—Model Driven Engineering, metamodeling, Microservices, Internet of Things, Devops, Docker, Ansible, Kubernetes.

I. INTRODUCTION

Companies are adopting agile and lean development processes in practises to improve the impact of their program process and to increase the quality of their products. They use the DevOps [1] tools, and set of tool that belongs to DevopsTool chain. Each group of tool belong to a step of continuous Integration [2] – Continous Deployement [3] – Continous Delivery [4] – Continous Testing [5].

Nowadays, developers and designers are adopting several technological trends in the field of the Internet of Things. These technologies include micro-services and Devops. Microservices are an architectural style in which the software system is built with standalone components. These components are separated from each other in terms of business functionality and have limited granularity.

Devops is a fusion of two words: Development and operation. Devops comprises the work of developers in parallel with the work of integrators. It allows developers to control the entire chain of integration and continuous deployment from development, through pishing code to the built environment, containerization and orchestration of deployed instances. Devops culture tools are at the service of micro-services. Among these Devops tools, which participate in the assurance of the deployment and continuous integration chain, we mention the following: Docker and Ansible [7,8].

Docker makes it possible to containerize the microservices developed in order to ensure their portability and load balancing between several microservice instances [8].

Ansible is an open-source DevOps [9] tool that can help the company in configuration management, deployment, provisioning, etc. It is simple to deploy; it uses SSH technology to communicate between servers [10]. It uses the playbook to describe automation tasks. Ansible participates in the continuous deployment of the Docker Containers realized. Playbook is a configuration in which we cite the name of docker image to be deployed, removed, or updated.

Model-Driven Engineering allows systems to be generated according to the designer's needs [11]. The objective of this article is to propose a PIM model [12] that gathers all the concepts that allow us to build an Internet of Things system based on microservices and supported by the two tools Devops Docker and Ansible. Namely, a metamodel is presented for each of the following ecosystems: IoT [13] – Microservices [14] - Docker [8]- Ansible [7] – Kubernetes [6]. Each ecosystem is modeled as a package and then these packages are linked together to form a global metamodel of the Internet of Things system based on microservices and supported by the two tools Devops Docker and Ansible.

This metamodel can be used in different domains. The third section talks about improving the Internet of Things metamodel. The fourth section talks about a metamodel enhancement of microservices. The fifth section talks about a proposed metamodel for the Docker containerization system. The sixth section discusses a proposed metamodel for the Ansible monitoring system. The seventh section discusses a proposed metamodel for Kubernetes. Finally, the seventh section merges these metamodels into a metamodel for a microservice-based Internet of Things system supported by the two tools Devops Docker and Ansible.

II. RELATED WORK

The metamodel of an Internet of Things system based on microservices and supported by the two tools Devops Docker [8] and Ansible [7] is a fusion of four metamodels: Internet of Things, Microservice, Docker, Ansible.

In the literature, there are proposals for metamodels for microservice ecosystems and the Internet of Things. The authors in [15] proposed a metamodel for the ecosystem of connected objects in the figure below:

Authors in [16] proposed a PIM Level for the microservice ecosystem:

The microservices in this model are part of the overall microservice architecture. They are divided into two types: functional microservices and infrastructure microservices. Microservices use load balancers and are deployed on containers. Microservices use service interfaces that are published at endpoints. Microservices depend on service operations. Data caching, storage, and asynchronous buses are part of service operations.

In the literature, there is a lack of metamodels for containerization Docker, thus Ansible.

The contributions of this article reside in:

- Improving the metamodel of connected objects.
- Improving the metamodel of microservices and proposing the appropriate type of microservice for IoT applications.
- Metamodel proposal for Docker.
- Metamodel proposal for Ansible.
- Metamodel proposal for Kubernetes.

III. CONSTRUCTING A METAMODEL

First, we produce a reference framework of a given subject; it contains important elements that should be included in the metamodel. Then a second step is to gather existing models of the subject, note that the more models we gather the best is the quality of our metamodel. Then we analyze the concepts using matching technics. Them, finally we optimize the concepts in one metamodel containing an aggregation to all models.

The following step must be executed to build a final metamodel [13]:

- Step 0: Reference Framework Definition

The reference framework definition in our case has to take into consideration 4 fields of search.

For each field of search, we define a set of rules & concepts that are interconnected to get as results in the relationship (Ri) and concepts (Cj).

- Step 1: Concepts Gathering

Each ecosystem has its PIM that has its specificities that are useful during implementation. PIMi is modeling of an ecosystem i.

$$i \in \{\text{Docker, IoT, Ansible, Microservices,Kubernetes}\}$$

Number equations consecutively. Equation numbers, within parentheses, are to position flush right, as in (1), using a right tab stop.

- Step 2: Metamodel Construction

The metamodel of an Internet of Things that is based on microservice and supported by Docker and Ansible is an intersection between the following PIM: PIM(IoT), PIM(Microservices), PIM(Docker), PIM(Ansible).

IV. METAMODEL OF THE INTERNET OF THINGS ECOSYSTEM

This metamodel is an improvement of the cited metamodel in the state of the art [18].

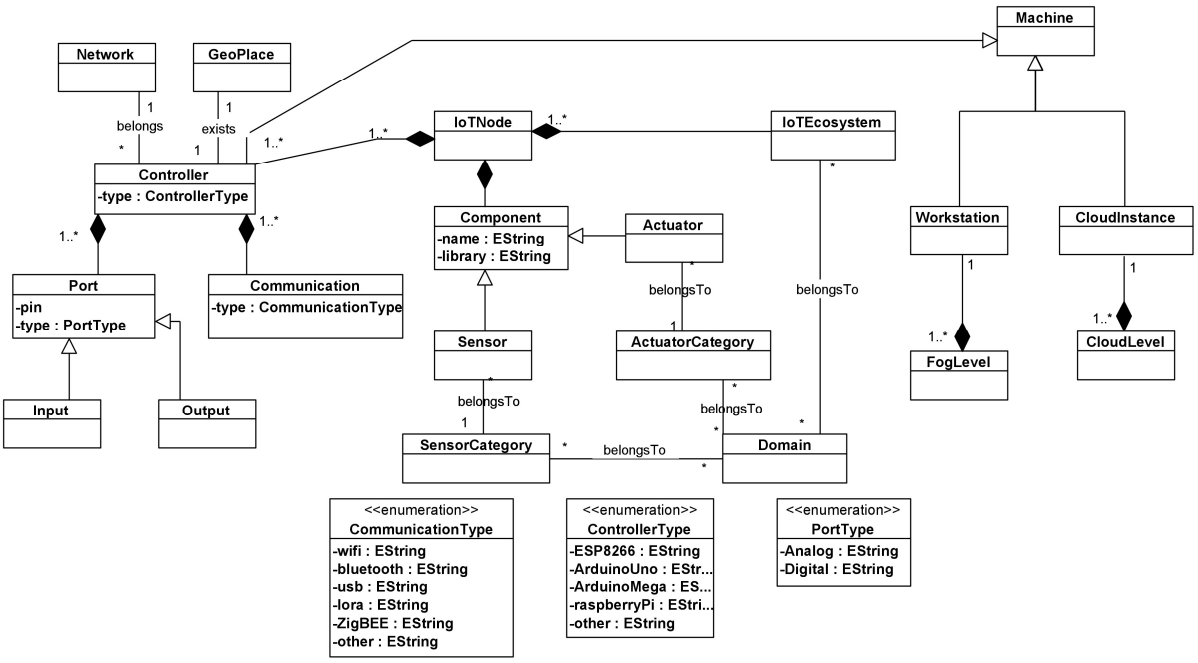

Fig. 1. Internet of Things Metamodel.

We have added the following classes:

- Network: The network to which the controller belongs manages the connected objects.
- GeoPlace: These GPS geographic coordinates determine the location of a controller in an ecosystem in order to locate them. This information is important in some ecosystems such as those belonging to the category: Smart Logistic - Smart Farm.
- SensorCategory: determines the category a sensor belongs to such as temperature, humidity, distance, etc.

- ActuatorCategory: determines the category an actuator belongs to such as motors, light bulbs, etc.
- Domain: Determines the domain to which the sensor and actuator categories belong. As an example, we have sensors that belong to the smart farm domain such as temperature, humidity, and acidity sensors. Also, some actuators belong to the smart vehicle domain such as engines.
- IoTEcosystem: identifies the ecosystem to be modeled.
- Machine: these are the devices and appliances that are used in an ecosystem, including controllers for the edge level, workstations for the fog level, and instances for the cloud level.

V. METAMODEL OF THE INTERNET OF MICROSERVICES

These microservices are programs that are deployed on machines belonging to the ecosystem [16].

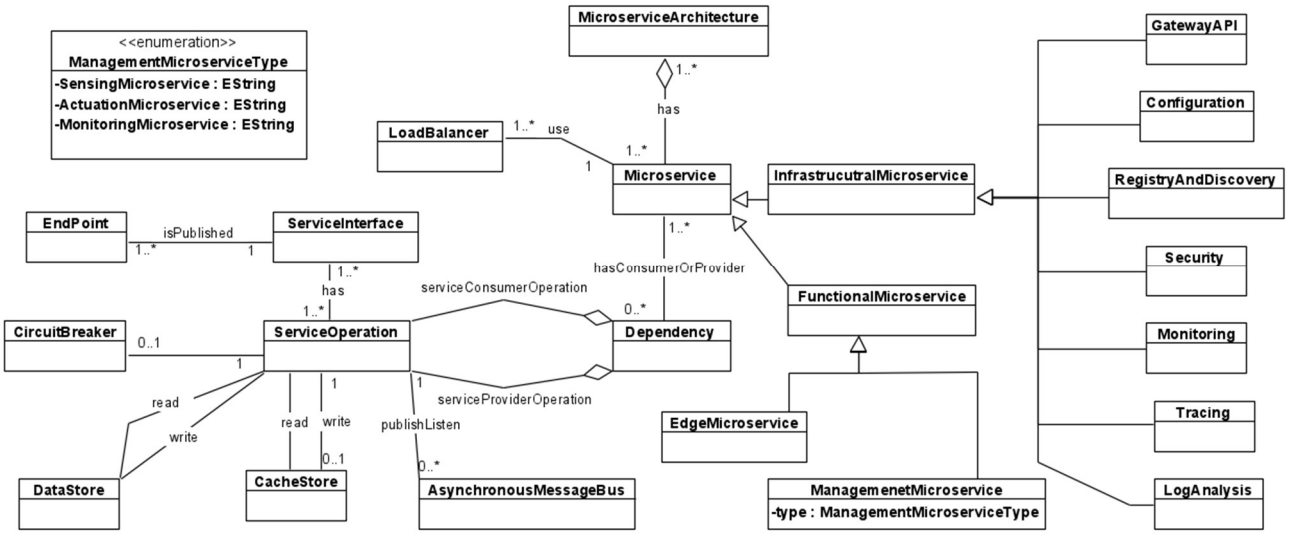

Fig. 2. Microservice Ecosystem Metamodel.

We were inspired by the meta-model cited in the state of the art. The components we used are as follows: Microservice is part of MicroserviceArchitecture by being the main component. Microservices are divided into two types: InfrastructuralMicroservice and FunctionalMicroservice. Among the InfrastructuralMicroservice there are API Gateway, Configuration, Registry and Discovery, Security, Monitoring, Tracing, Log Analysis. Microservices expose Service Interfaces that are published in EndPoint.

We have proposed types of microservices appropriate to the ecosystems of connected objects called FunctionalMicroservice they are divided into 3 categories: ActuationMicroservice which is responsible for sending commands to the actuator. SensingMicroservice which are responsible for sensing physical quantities from sensors. CompositeMicroservice are microservices that retrieve information from the SensingMicroservice and perform processing in order to make decisions to send commands to the ActuationMicroservice.

VI. METAMODEL OF DOCKER

Docker provides containerization services and it is based on Linux containers. Docker provides a standard runtime across Docker Engine. It allows us to build Image format. When the image is run in such an environment like Cloud, Fog, Edge it is called container. Microservices are organized in the form of containers are dedicated to being deployed on machines belonging to the Internet of Things ecosystem.

The metamodel of the Docker ecosystem is organized as follows:
DockerDaemon is the docker engine that manages a set of ControlGroup. Docker's ControlGroup manages DockerObject. DockerObjects are divided into two types, Image and Container. A Container is an instantiation of an Image. An Image is stored in a DockerRegistry which can be private or public. An Image has a set of ImageVersion versions. The DockerFile is a file that is responsible for creating the DockerImage in the DockerRegistry. DockerCompose is a file that is composed of several services, each service representing a DockerImage. DockerCompose is responsible for deploying an application composed of several DockerImages. The NameSpace are used to provide isolation to the Container. Docker creates a set of NameSpace for each Container.

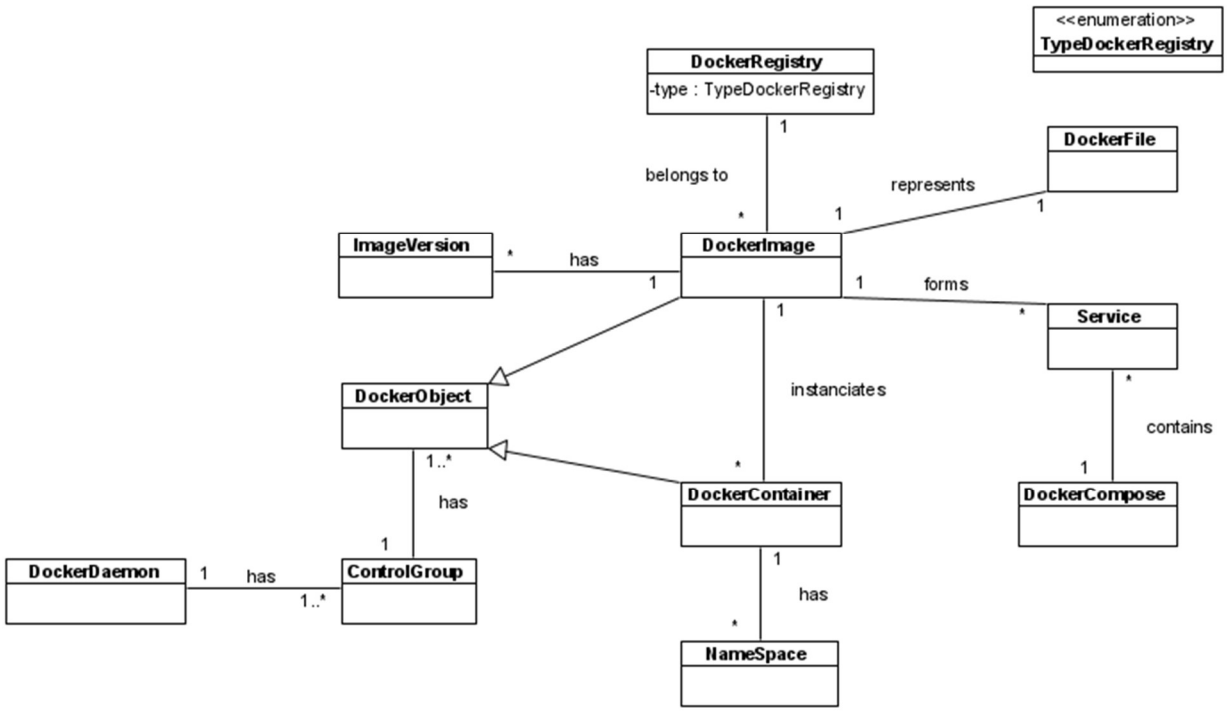

Fig. 3. Docker Ecosystem MetaModel.

VII. METAMODEL OF ANSIBLE

Ansible is an open-source DevOps tool that can help the enterprise in configuration management, deployment, provisioning, etc. It is simple to deploy; it uses SSH technology to communicate between servers. It uses the playbook to describe automation tasks, and the playbook uses a very simple language, YAML [22].

Ansible is responsible for deploying the Docker Containers on the machines. Its metamodel is organized as follows:
Ansible's architecture is a Master-Slave architecture. The AnsibleMaster is the machine that performs the administration of other machines that are no other than microcontrollers. The machines we want to monitor are stored in a file called Inventory. The AnsibleMaster generates an SSHKey that it shares with the other "microcontroller" machines to allow resource sharing with these machines. Resource sharing and container deployment is done through YAML files called PlayBook.

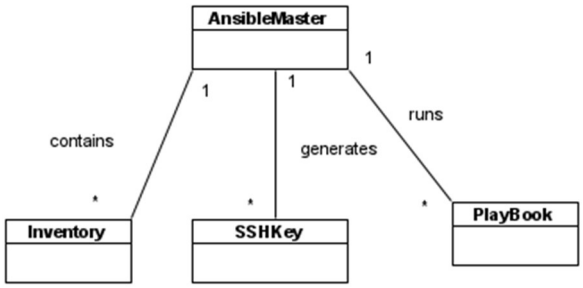

Fig. 4. Ansible Ecosystem MetaModel.

VIII. METAMODEL OF KUBERNETES

Depending on Kubernetes.io, Kubernetes cluster is an orchestrated set of nodes. Kubernetes offers two types of node is Master Node, Slave Node. The Master Node represents the server that performs the deployment of the Pods. A Pod is the basic execution unit of a Kubernetes application - the smallest and simplest unit of the Kubernetes object model that you create or deploy. A Pod represents the processes running on your cluster. A Pod encapsulates an application Container (or, in some cases, multiple Containers), storage resources, a single network IP, and the options that govern how the Container(s) should operate. A Pod represents a single deployment unit is a single instance of an application in Kubernetes, which may consist of either a single container or a small number of closely coupled and resource-sharing containers.

In particular, the Master Node is the server that performs the deployments on the microcontrollers. The microcontrollers are the Slave Nodes on which the Docker containers are deployed in the form of Pods.
Kubernetes is responsible for the orchestration of the Docker containers. Cluster is a set of machines. Node is the machine on which the Container Pods run. NonMaster is the node on which the Pods are deployed. Master is a node that controls the NonMaster Nodes. Pod is a group of containers deployed together.

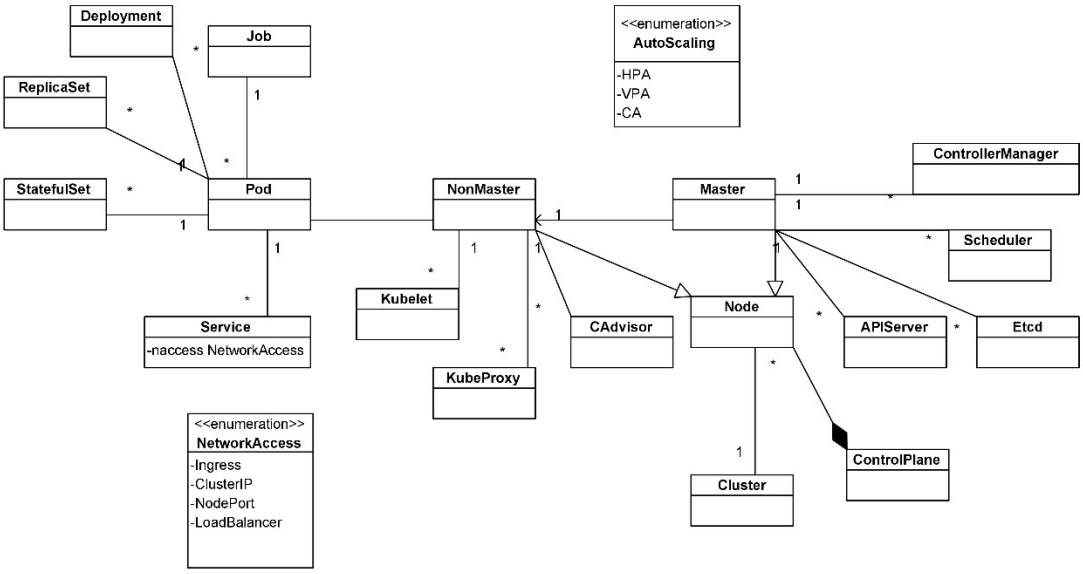

Fig. 5. Kubernetes Ecosystem MetaModel.

IX. Global Meta-Model

The principle of this Meta-model stipulates that the microservice is containerized using Docker. The Docker container is made in the form of a Pod in order to scale it up using Kubernetes. The Kubernetes Pods are started in the Kubernetes Node which forms a cluster. The cluster is created and managed by Ansible's playbooks. The AnsibleMaster machine monitors the existing microcontrollers in the IoT ecosystem by sharing SSH keys and launching playbooks to deploy the Kubernetes Pods.

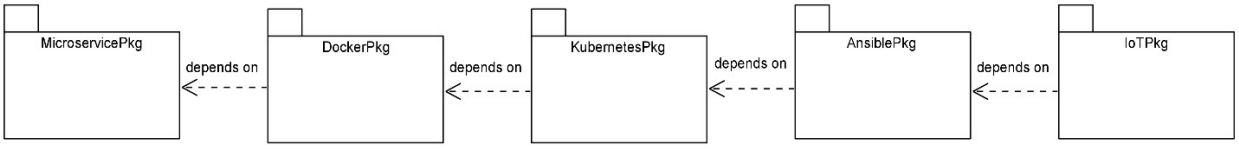

Fig. 6. Global MetaModel.

X. Discussion

In this paper, we have elaborated a metamodel which is a fusion of 4 metamodels: metamodel of connected objects - metamodel of microservices - metamodel of Docker - metamodel of Ansible. First of all, we improved the metamodel of connected objects quoted in the state of the art, this improvement will allow identifying the domain to which the objects belong as well as the categories of sensors and actuators, so it allows the geographical and network identification in an Internet of Things ecosystem. Secondly, we have realized an improvement for the metamodel of microservices by providing appropriate microservices for the management of connected objects and data processing and decision support for the Internet of Things agents. Besides, we have developed a metamodel for containerization Docker, in which we have grouped the concepts constituting an ecosystem that aims to dock the microservices that will be deployed in objects and machines, whether at the cloud or fog level. Also, we have developed an Ansible metamodel, which brings together the components of this system and whose objective is to monitor the containers in the machines and controllers. Finally, we have grouped these metamodels into a single one and realized a dependency between the layers that are represented by the packages. Each package represents a metamodel. The objective of this metamodel is to generate an Internet system of objects based on microservices and supported by Devops technologies: Docker & Ansible.

In future works, we will use this metamodel to realize a model transformation using ATL Language.

XI. Conclusion

The paper talks about a fusion of 4 metamodels: Internet of things metamodel, Microservice metamodel, Docker metamodel, Ansible metamodel. We have elaborated an amelioration of the existing metamodel regarding IoT, we have elaborated an amelioration of Microservice metamodel by adding enumerations that correspond to the Internet of Things need. Then we have made a new proposition regarding Docker and Ansible metamodel.

References

[1]. Len Bass. The software architect and DevOps. IEEE Software, 35(1):8–10, jan 2018.

[2]. Mojtaba Shahin, Muhammad Ali Babar, and Liming Zhu. Continuous integration, delivery and deployment: Asystematic review on approaches, tools, challenges and practices. IEEE Access, 5:3909–3943, 2017

[3]. Eero Laukkanen, Juha Itkonen, and Casper Lassenius. Problems, causes and solutions when adopting continuous delivery—a systematic literature review. Information and Software Technology, 82:55–79, feb 2017

[4]. Daniel Ståhl, Kristofer Hallén, and Jan Bosch. Achieving traceability in large scale continuous integration and delivery deployment, usage and validation of the eiffel framework. Empirical Software Engineering, 22(3):967–995, oct 2016

[5]. S. Stolberg, Enabling Agile Testing Through Continuous Integration, Agile 2009 Conference, Chicago, IL, 2009.

[6]. Kubernetes. https://kubernetes.io/. Accessed 19 Aug 2020

[7]. Hochstein, Lorin, and Rene Moser. Ansible: Up and Running: Automating Configuration Management and Deployment the Easy Way. " O'Reilly Media, Inc.", 2017.

[8]. Merkel, Dirk. "Docker: lightweight linux containers for consistent development and deployment." Linux journal 2014.239 (2014): 2.

[9]. Ebert, Christof, et al. "DevOps." Ieee Software 33.3 (2016): 94-100.

[10]. LIU, Bin, and Zui WANG. "Application of office automation based on ssh framework [j]." Computer Technology and Development 1 (2010): 39.

[11]. Schmidt, Douglas C. "Model-driven engineering." COMPUTER-IEEE COMPUTER SOCIETY- 39.2 (2006): 25.

[12]. Kleppe, Anneke G., et al. MDA explained: the model driven architecture: practice and promise. Addison-Wesley Professional, 2003.

[13]. Lee, In, and Kyoochun Lee. "The Internet of Things (IoT): Applications, investments, and challenges for enterprises." Business Horizons 58.4 (2015): 431-440.

[14]. Thönes, Johannes. "Microservices." IEEE software 32.1 (2015): 116-116.

[15]. D. Alulema, J. Criado, and L. Iribarne, “A Model-Driven Approach for the Integration of Hardware Nodes in the IoT,” in Advances in Intelligent Systems and Computing, 2019, vol. 930, pp. 801–811.

[16]. N. Alshuqayran, N. Ali, and R. Evans, “Towards Micro Service Architecture Recovery: An Empirical Study,” in Proceedings - 2018 IEEE 15th International Conference on Software Architecture, ICSA 2018, 2018, pp. 47–56.

[17]. H. Ibrahim and B. Abdessamad, “Project Management Metamodel Construction Regarding IT Departments,” Int. J. Adv. Comput. Sci. Appl., vol. 10, no. 10, p. 208, 2019.

[18]. Badr El Khalyly, et al. " A comparative study of Microservices-Based IoT platforms" International Journal of Advanced Computer Science and Applications (IJACSA) 11, no. 7 (2020) (in progress).

[19]. J. Turnbull, The Docker book. James Turnbull, 2014.

[20]. T. Lu and J. Chen, “Research of Penetration Testing Technology in Docker Environment,” 2017.

[21]. “(PDF) iBrownout: An Integrated Approach for Managing Energy and Brownout in Container-based Clouds.” [Online]. Available: https://www.researchgate.net/publication/323405045_iBrownout_An_Integrated_Approach_for_Managing_Energy_and_Brownout_in_Container-based_Clouds. [Accessed: 03-May-2020].

[22]. M. Zadka and M. Zadka, “Ansible,” in DevOps in Python, Apress, 2019, pp. 139–145.