1. Grzegorz RYBAK[1], 2. Krzysztof KRÓL[1,2]

Research and Development Center, Netrix S.A., Lublin (1), WSEI University, Lublin (2)
ORCID: 1. 0000-0003-1250-2115; 2. 0000-0002-0114-2794

doi:10.15199/48.2025.01.48

Integration of the industrial tomography platform components in the IOT concept and cloud architecture

***Abstract**. W celu zwiększenia efektywności diagnostyki i kontroli procesów technologicznych zaprojektowano i wdrożono przemysłową platformę tomografii elektrycznej Platom. System jest zgodny z koncepcją Przemysłu 4.0 oraz tak zwanego Internetu Rzeczy (ang. Internet of Things, IoT). Platforma została opracowana w szczególności dla procesów: separacji płynów, fermentacji i krystalizacji. Realizacja projektu w jego końcowej fazie obejmowała przygotowanie planów wdrożeniowych, które mogą być realizowane przy wykorzystaniu zaawansowanych technik „ciągłego dostarczania" (ang. continuous delivery). W niniejszym artykule prezentujemy wybrane rozwiązania zarządzania urządzeniami pomiarowymi oraz metody wdrażania modułów platformy tomograficznej utworzonej w koncepcji architektury chmury obliczeniowej w modelu PaaS*

***Streszczenie.** In order to increase the diagnostic and control efficiency of technological processes, the industrial electrical tomography platform Platom was designed and implemented. The system complies with Industry 4.0 and the so-called Internet of Things (IoT). A platform has been developed for fluid separation, fermentation and crystallization processes. The realization of the project in its final phase included deployment plan preparation, which may also be executed with advanced 'continuous delivery' techniques. This work presents selected measurement device management solutions and deployment methods created in the cloud architecture PaaS model tomographic platform (**Integracja komponentów platformy tomografii przemysłowej z koncepcją IOT i architekturą chmury**).*

Keywords: ECT, EIT, UST, PaaS, IoT, software deployment, containerization
Słowa kluczowe: ECT, EIT, UST, PaaS, IoT, wdrażanie oprogramowania, konteneryzacja

Introduction

Nowadays, many industrial processes use tomographic techniques like Electrical Capacitance Tomography (ECT), Electrical Impedance Tomography (EIT) or Ultrasound Tomography (UST). Thanks to them, it is possible to measure the inside of system components such as tanks or pipelines [1-4]. Often, many computer system components require advanced management and integration methods, communication solutions such as message brokers (ex. Kafka, JMS, Mosquitto), orchestration, configuration services, service discovery, modules monitoring and load balancing or infrastructure security assurance. The complexity of the system, in turn, implies several problems and challenges related to the integration of components, their management and deployment [5-8].

The tomographic platform Platom, developed by a group of scientists cooperating within the consortium, is characterized by decentralization, heterogeneity and high modularization, where its individual deployment units require different installation processes and configurations [9-11].

The authors present an industrial system based on the PaaS model in which the deployment process takes advantage of containerization and fleet management techniques. Measurement devices were adapted by integrating them with other microservices in the IoT concept. A complex control and measurement system consists of numerous components, which include, among others, physical sensors and their software, measurement data acquisition modules, communication bus, communication protocol, visualization modules, user interfaces and advanced measurement data processing components (Fig.1). In order to deal with the set of deployment problems, containerization techniques are often used in industrial solutions, which is widely described in the literature. It excellently reduces the cost of rebuilding the cloud development platform and is better than virtual machines [12]. In [13], the multi-parameter new iterative method of CT image reconstruction development was based on the Grid platform and docker containers. Other studies also emphasize the advantage of using containers in microservices architecture along with resource optimization models. Considering the features and advantages of containerization, as part of research [14], a resource allocation model for microservice-based applications was created using a docker container. An efficient communication and scalable resource allocation algorithm, EPTA, has been proposed in [14]. Other solutions are IoTDoc and Dep-Pipe. The first is a viable option for cloud computing and is a more affordable, cost-effective alternative to large platform cloud computing services than Amazon EC2 [15]. The second one enables flexible and scalable deployment and orchestration of Big Data pipelines over the Computing Continuum resources [16].

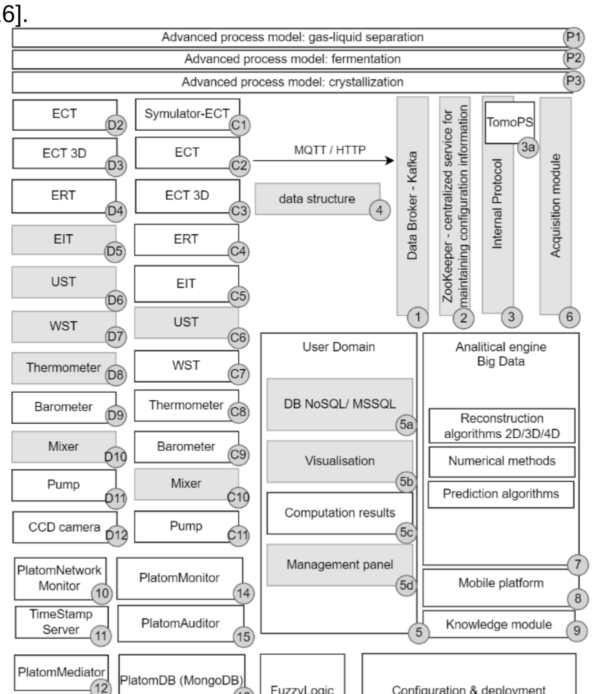


Fig. 1. Diagram of the main modules in the industrial process control platform; elements of the basic deployment process are marked in green (sections: D – devices; P – process models; C – connectors)

A different situation concerns the deployment of software to measuring devices. It is essential to ensure a consistent automatic deployment across the IoT-edge-cloud continuum, covering different types of devices. A model-based approach to deploying devices to solve the fleet deployment problem is presented in [17]. This approach is called Generation and Deployment of Smart IoT Systems (GeneSIS).

The main problem constituting the need to develop the solution described in this article was to design a model and prepare an implementation of mechanisms thanks to which a complex platform could be managed in an automated or semi-automated manner.

System components deployment method

The Platom laboratory setup with several devices is presented in Fig. 2. It contains an electrical impedance tomography (EIT), ultrasound tomography (UST), M-Bus controller, thermometer, mixer, pump and server. From now on, the collection of devices is referred to as a fleet. Additionally, the system was made in the concept of edge computing, where key algorithms are performed on edge devices; thus, measurement, preliminary computations, and image reconstruction can be done without server engagement, which affects subsequent modules installed on individual devices.

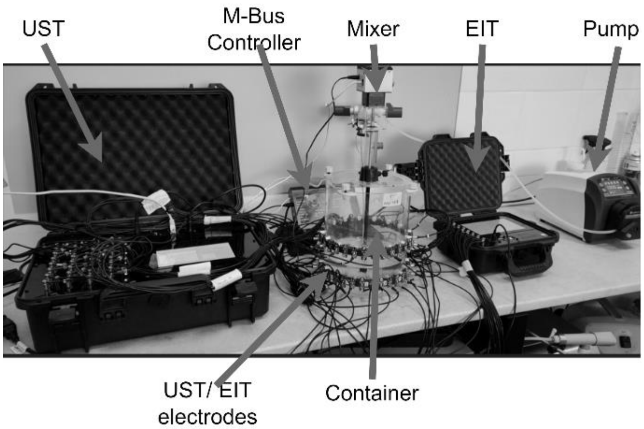


Fig. 2. The core setup of industrial electrical impedance tomography (EIT) and ultrasound tomography (UST) systems physical components for crystallization process

Automatic software deployment is key to adopting DevOps tools such as Puppet, Ansible, and Chef17 and cloud frameworks such as Azure IoT Edge or AWS IoT that provide services to maintain a fleet throughout its lifecycle. The main concept of the presented solution is based on the IaC approach (Infrastructure as Code) [18]. This is a key DevOps practice that assumes the existence of scripts describing the infrastructure. The concept evolved to solve the problem of environment drift in release pipelines. In the Platom system, configuration scripts for tools such as Puppet, Ansible, Docker, and Docker Compose were used. The idea of separating the deployment process's meta-model and assigning deployment plans to system components was also applied.

The process assumes the existence of management software, thanks to which it is possible to control devices connected to the system and dynamically update the target software. Additionally, an agent must be installed on each device. This software should be prepared first on a physical or virtual device (container, virtual machine). During the installation of the target software, it uses the deployment unit repository (installation packages repository) and the repository of deployment scripts, e.g. Ansible, Docker Compose scripts etc.

The process begins with the installation of the deployment management software. Each device managed by the system should have network access and access to the infrastructure and Python environment installed (the requirement of an Ansible tool). Each device should report its ID and address/port to the manager, or this information should exist in the deployment plan. The deployment plan contains a sequence of installation and configuration actions and requirements. Fig. 3 and eq. 1 presents the basis of the designed model.

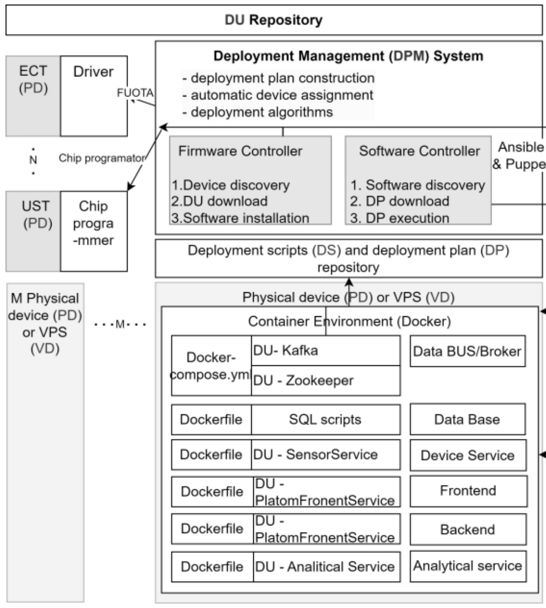


Fig. 3. Deployment model for tomographic platform

$$M = \{DPM,\ DPU,\ D,\ DP,\ DS,\ DU,\ R,\ C\} \tag{1}$$
$$D = \{PD,\ VD\}$$

where: *M* – prepared model; *DPM* – deployment manager, *DPU* – set of deployment management units (agent), *D* – set of devices, *PD* - physical devices, *VD* - virtual devices, *DU* – set of deployment units, *DP* – set of deployment plans, *DS* – set of deployment scripts, *R* – set of requirements, *C* – set of capabilities.

In order to automatically assign deployment plans to devices, it was necessary to verify whether the target device met all requirements declared in the plan. The analysis space is declared a matrix *A* (eq. 2), the Cartesian product of a set of devices and a set of plans combined with device capabilities and implementation plan requirements, respectively.

(2)
$$A=(DU\times C)\times(DP\times R)=\begin{vmatrix} DU_0C_0\ DP_0R_0 & \cdot\ \cdot & DU_0C_0\ DP_oR_s \\ \cdot & \cdot\ \cdot & \cdot \\ \cdot & \cdot\ \cdot & \cdot \\ \cdot & \cdot\ \cdot & DU_mC_n\ DP_oR_s \end{vmatrix}$$

All elements in matrix A should be verified against rules adequacy and compliance. Analysis space *A* is reduced only to selected elements where a given constraint exists in the deployment plan, and the device declares the given capability.

(3) $A` = reduce(A)$

where: reduce() – function that limits the analysis space to a valid range; A` is a valid analysis space for deployment tasks.

When preparing the result of the automatic assignment (matrix *I*), the *aq* and *com* functions are performed for each element of the matrix *A*`. The first of them allows us to verify whether the requirement type in the deployment plan corresponds to the capabilities type of the device. The second function allows to verify whether the rule has been met. Both functions return binary values: {0,1}. The elements of the submatrix of devices and deployment plans pair are summed. Then, the ratio of fulfilled rules to all declared rules is calculated (eq. 4,5,6). The plan is automatically assigned to the device if all conditions are met. The first one is selected when several plans can be deployed on one device.

(4)
$$T(dpu, dp) = \sum_{k=0}^{|C`_{dpu}|-1} \sum_{l=0}^{|S`_{dp}|-1} aq\left(A`_{fr(dpu)+k, fc(dp)+l}\right) \cdot com\left(A`_{fr(dpu)+k, fc(dp)+l}\right)$$

(5)
$$W(dpu, dp) = \sum_{k=0}^{|C`_{dpu}|-1} \sum_{l=0}^{|S`_{dp}|-1} aq\left(A`_{fr(dpu)+k, fc(dp)+l}\right)$$

(6)
$$I(dpu, dp) = \frac{T(dpu, du)}{W(dpu,\ dp)}$$

where: *dpu* – a device with an agent, *dp* - deployment plan, *fr()* – first row of the selected device in matrix A`; *fc()* – first column in matrix A` for the selected plan.

Finally, when a set of deployment plans is matched, all required actions are performed sequentially or concurrently, resulting in a consistent production environment. This solution also allows software updates for battery-powered devices using a network connection (firmware Update Over-The-Air or "FUOTA") in a similar way as presented in [19].

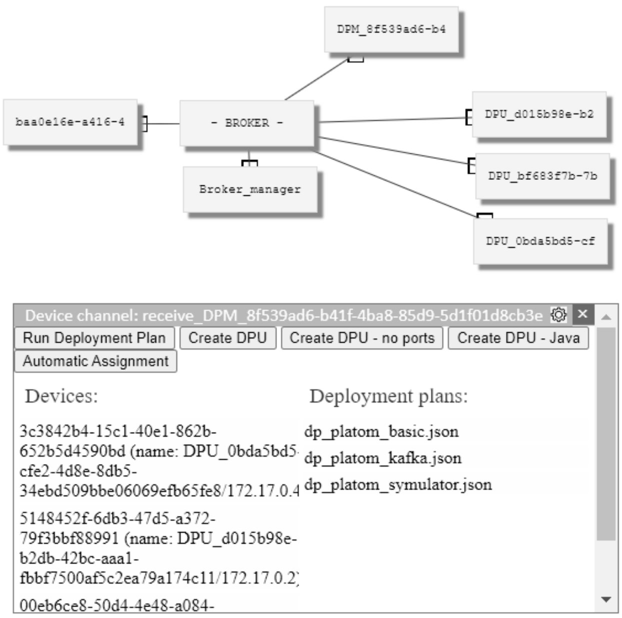
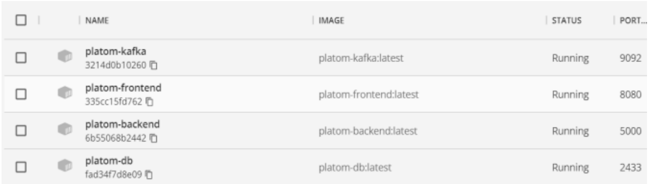


Fig. 4. Deployment management control panel (web application)

Results and analysis

The realization of the project and implementation of the software involved a number of time-consuming works consisting in the preparation of the infrastructure, which included the necessary protocols and modules: broker based on the WebSocket protocol, a broker manager allowing to manage the connected clients, deployment manager module (DPM) with rules solver, an agent that is responsible for target deployment, as well as the user interface. The web-application-based control panel is presented in Fig. 4.

It contains two sections. The first one shows the currently active modules connected to the system. Below is a dynamic window containing a list of DPUs with agents and deployment plans. The system allows manually selecting and installing the pair (fig. 5).



NAME	IMAGE	STATUS	PORT...
platom-kafka 3214d0b10260	platom-kafka:latest	Running	9092
platom-frontend 335cc15fd762	platom-frontend:latest	Running	8080
platom-backend 6b55068b2442	platom-backend:latest	Running	5000
platom-db fad34f7d8e09	platom-db:latest	Running	2433

Fig. 5. Example of platform deployment in docker environment - direct installation

One of the platform's capabilities is the creation of containers in the Docker environment with predefined parameters, such as runtime environments or available communication ports. The last option is to assign deployment plans to devices and run installation automatically. The result of the automatic adjustment is presented in Fig. 6 and Fig. 7.

DPlan\PD||VD	DPU_0bda5bd5-cfe2-4d8e-8d && ARCH	DPU_0bda5bd5-cfe2-4d8e-8d && FREE_GPIO	DPU_0bda5bd5-cfe2-4d8e-8d && FREE_PORTS	DPU_0bda5bd5-cfe2-4d8e-8d && FREE_RAM	DPU_0bda5bd5-cfe2-4d8e-8d && JAVA_SUPPORT	DPU_0bda5bd5-cfe2-4d8e-8d && SYS	DPU_0bda5bd5-cfe2-4d8e-8d && TYPE	DPU_bf683f7b-7b0a-4dd0-97 && ARCH	DPU_bf683f7b-7b0a-4dd0-97 && FREE_GPIO	DPU_bf683f7b-7b0a-4dd0-97 && FREE_PORTS	DPU_bf683f7b-7b0a-4dd0-97 && FREE_RAM	DPU_bf683f7b-7b0a-4dd0-97 && JAVA_SUPPORT	DPU_bf683f7b-7b0a-4dd0-97 && SYS	DPU_bf683f7b-7b0a-4dd0-97 && TYPE	DPU_d015b98e-b2db-42be-aa && ARCH	DPU_d015b98e-b2db-42be-aa && FREE_GPIO	DPU_d015b98e-b2db-42be-aa && FREE_PORTS	DPU_d015b98e-b2db-42be-aa && FREE_RAM	DPU_d015b98e-b2db-42be-aa && JAVA_SUPPORT	DPU_d015b98e-b2db-42be-aa && SYS	DPU_d015b98e-b2db-42be-aa && TYPE
dp_platom_basic.json && FREE_PORTS	false	false	false	false	false	false	false	false	false	true	false	false	false	false	false	false	false	false	false	false	false
dp_platom_basic.json && FREE_RAM	false	false	false	true	false	false	false	false	false	false	true	false	false	false	false	false	false	true	false	false	false
dp_platom_basic.json && SYS	false	false	false	false	false	true	false	false	false	false	false	false	true	false	false	false	false	false	false	true	false
dp_platom_kafka.json && NVIDIA_SUPPORT	false	false	false	false	false	false	false	false	false	false	false	false	false	false	false	false	false	false	false	false	false
dp_platom_symulator.json && FREE_RAM	false	false	false	true	false	false	false	false	false	false	true	false	false	false	false	false	false	true	false	false	false
dp_platom_symulator.json && JAVA_SUPPORT	false	false	false	false	true	false	false	false	false	false	false	false	false	false	false	false	false	false	false	false	false
dp_platom_symulator.json && SYS	false	false	false	false	false	true	false	false	false	false	false	false	true	false	false	false	false	false	false	true	false

Fig. 6. Deployment plan auto assignment visualization (rows are deployment plan requirements and columns are device capabilities; cell shows if requirement is fulfilled)

a)

DPlan\PD||VD	DPU_0bda5bd5-cfe2-4d8e-8d	DPU_d015b98e-b2db-42be-aa	DPU_bf683f7b-7b0a-4dd0-97
dp_platom_basic.json	false	false	true
dp_platom_kafka.json	false	false	false
dp_platom_symulator.json	true	false	false

b)

DPlan\PD||VD	DPU_8e60e958-70f1-4cdd-bd	DPU_86b09694-4cfa-4498-93	DPU_5f64f6ad-cffc-4501-af
dp_platom_basic.json	false	false	true
dp_platom_kafka.json	false	true	
dp_platom_symulator.json	true		

Fig. 7. Deployment plan auto assignment (rows: deployment plan; columns: device; cell shows if a plan may be implemented in the device) a) refers to fig. 7.; b) adjusted capabilities

The result in the docker desktop is presented in Fig. 8. Container set consists of Apache Kafka, Zookeeper, MSSQL, Backend, Frontend, Sensor-service, and ECT-simulator installed inside previously prepared containers with deployment agents.

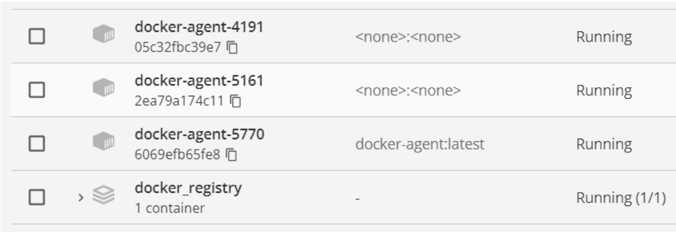


Fig. 8. Example of platform deployment result in docker environment - installation with agents

Finally, a coherent environment is obtained with installed and active services that allow for efficient measurement and control of industrial processes.

Using the developed solution, a significant acceleration of task implementation was noticed, which also resulted from the analysis of implementation time. The time of launching containers with agents was measured, as well as the deployment time of the entire tomography platform. Kafka and Zookeeper deployment from scratch took 7min12 sec., ECT simulator 2min 15sek and whole platform deployment took 10min 12sec. The authors did not perform a run-time workload analysis. Container preparation times are presented in Table 1.

Table 1. Docker containers configuration and deployment times

Container without ports [s]	Container with ports [s]	Container without ports and Java support [s]	3 containers at once [s]
3,85	2,12	5,90	7,34
2,83	1,84	3,47	4,53
3,13	2,74	8,37	4,65
2,94	6,73	2,83	7,63
2,92	4,42	10,18	11,43
2,99	2,76	3,90	5,04
4,20	2,17	9,84	5.33

Summary

The implementation, integration and delivery of innovative tomographic platform components are a complex task that requires the ability to use advanced tools and extensive specialist knowledge in computer system architecture and DevOps methodology. The authors have prepared a solution based on containerization and fleet management techniques, proposing a partial adaptation of the GeneSIS model and a deployment engine in the tomography system, finally obtaining fully automated continuous integration and deployment processes for an industrial monitoring platform. However, the complexity of the problem requires further research.

***Authors**: Grzegorz Rybak, Ph.D. Eng., Netrix S.A., Research and Development Center, Związkowa 26, 20-148 Lublin, e-mail: grzegorz.rybak@netrix.com.pl; Krzysztof Król, WSEI University, Projektowa 4, Lublin,, Research&Development Centre Netrix S.A., Lublin Związkowa 26. Email: krzysztof.krol@netrix.com.pl;*

REFERENCES

[1] Rymarczyk T., Kłosowski G., Hoła A., Hoła J., Sikora J., Tchórzewski P., Skowron Ł., Historical Buildings Dampness Analysis Using Electrical Tomography and Machine Learning Algorithms, Energies, 14 (2021), No. 5, 1307.

[2] Kłosowski G., Rymarczyk T., Niderla K., Rzemieniak M., Dmowski A., Maj M., Comparison of Machine Learning Methods for Image Reconstruction Using the LSTM Classifier in Industrial Electrical Tomography, Energies 2021, 14 (2021), No. 21, 7269.

[3] Rymarczyk T., Kłosowski G., Hoła A., Sikora J., Tchórzewski P., Skowron Ł., Optimising the Use of Machine Learning Algorithms in Electrical Tomography of Building Walls: Pixel Oriented Ensemble Approach, Measurement, 188 (2022), 110581.

[4] Kłosowski G., Rymarczyk T., Kania K., Świć A., Cieplak T., Maintenance of industrial reactors supported by deep learning driven ultrasound tomography, Eksploatacja i Niezawodnosc – Maintenance and Reliability; 22 (2020), No 1, 138–147.

[5] Gnaś, D., Adamkiewicz, P., Indoor localization system using UWB, Informatyka, Automatyka, Pomiary W Gospodarce I Ochronie Środowiska, 12 (2022), No. 1, 15-19.

[6] Styła, M., Adamkiewicz, P., Optimisation of commercial building management processes using user behaviour analysis systems supported by computational intelligence and RTI, Informatyka, Automatyka, Pomiary W Gospodarce I Ochronie Środowiska, 12 (2022), No 1, 28-35.

[7] Kropidłowska P., Irzmańska E. Korzeniewska E., Tomczyk M., Jurczyk-Kowalska M., Evaluation of laser texturing in fabricating cut-resistant surfaces for protective gloves, Textile Research Journal, 93 (2023), No. 9-10), 1917–1927.

[8] Pawłowski S., Plewako J., Korzeniewska E., Field Modeling of the Influence of Defects Caused by Bending of Conductive Textronic Layers on Their Electrical Conductivity, Sensors, 23 (2023), No. 3, 1487.

[9] Koulountzios P., Rymarczyk T., Soleimani M., A triple-modality ultrasound computed tomography based on full-waveform data for industrial processes, IEEE Sensors Journal, 21 (2021), No. 18, 20896-20909.

[10] Koulountzios P., Aghajanian S., Rymarczyk T., Koiranen T., Soleimani M., An Ultrasound Tomography Method for Monitoring CO2 Capture Process Involving Stirring and CaCO3 Precipitation, Sensors, 21 (2021), No. 21, 6995.

[11] Kłosowski G, Rymarczyk T, Niderla K, Kulisz M, Skowron Ł, Soleimani M., Using an LSTM network to monitor industrial reactors using electrical capacitance and impedance tomography – a hybrid approach. Eksploatacja i Niezawodnosc – Maintenance and Reliability, 25 (2023), No. 1, 11.

[12] Bashari R.B., Bhatti H., AhmadiM., An Introduction to Docker and Analysis of its Performance. *IJCSNS International Journal of Computer Science and Network Security*, (2017), 173.8.

[13] Pérez C. et all., Combining Grid Computing and Docker Containers for the Study and Parametrization of CT Image Reconstruction Methods. *Procedia Computer Science;* (2017); 108. 1195-1204. 10.1016/j.procs.2017.05.065

[14] Wan, Xili & Guan, Xinjie & Wang, Tianjing & Bai, Guangwei & Choi, Baek-Young. (2018). Application deployment using Microservice and Docker containers: Framework and optimization. *Journal of Network and Computer Applications.*

[15] Noor S. et. all., *IoTDoc:* A Docker-Container Based Architecture of IoT-Enabled Cloud System, (2020).

[16] https://datacloud-project.github.io/toolbox.

[17] Song H. et all., Model-based fleet deployment in the IoT–edge–cloud continuum, *Software and Systems Modeling, Springer*, 2022.

[18] https://learn.microsoft.com/pl-pl/devops/deliver/what-is-infrastructure-as-code,(access: 24.03.2023)

[19] Abdelfadeel K. et. all., How to Make Firmware Updates over LoRaWAN Possible, 2020, *IEEE 21st International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM)*, Cork, Ireland, (2020), 16-25, doi: 10.1109/WoWMoM49955.2020.00018