RCTA
Revista Colombiana de Tecnologías de Avanzada
UNIPAMPLONA

Digital Object Identifier: 10.24054/rcta.v1i43.2912

Pattern for identifying the deployment points of artificial intelligence algorithms on the internet of things (IoT)

Patrón para identificar en la web de las cosas los puntos de despliegue de algoritmos de inteligencia artificial

MSc. Camilo Enrique Romero Parra [1], PhD. Carlos Alberto Cobos Lozada [1]
PhD. Miguel Ángel Niño Zambrano [1]

Universidad del Cauca, *Facultad de Ingeniería Electrónica y Telecomunicaciones, Grupo de investigación y desarrollo en tecnologías de la información - GTI, Popayán, Cauca, Colombia.*

Correspondence: *ceromero@unicauca.edu.co*

Received: *November 8, 2023.* ***Accepted:*** *January 10, 2024.* ***Published:*** *May 7, 2024.*

How to Cite: *C. E. Romero Parra, C. A. Cobos Lozada, and M. A. Niño Zambrano, "Pattern for identifying the deployment points of artificial intelligence algorithms on the internet of things (IoT)", RCTA, vol. 1, no. 43, pp. 144–154, May 2024.*
Recovered from https://ojs.unipamplona.edu.co/index.php/rcta/article/view/2912

Copyright 2024 Colombian Journal of Advanced Technologies.
This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License.

Abstract: When developing IoT solutions, there are limitations in hardware and software capabilities. In addition, the developer must select the location within the ecosystem that best suits the development needs, having three possible locations for processing: the edge of the network, the fog, and the cloud. This article proposes an architectural pattern to guide the selection of the deployment point for applications based on artificial intelligence algorithms based on the needs of the developed technological solution. Pratt's Iterative Research Pattern was used to obtain the proposed pattern. A real-world example was used, and the proposed step-by-step approach was applied to demonstrate the usefulness of the pattern. It was concluded that the selection of the processing location must consider the end user's needs and the potential limitations.

Keywords: Internet of Things, Architectural Pattern, Web of Things, Software Engineering, Artificial Intelligence.

Resumen: Al desarrollar soluciones IoT se presentan limitaciones en las capacidades de hardware y software. Además, el desarrollador deberá seleccionar la ubicación dentro del ecosistema que mejor se adecua a las necesidades del desarrollo, teniendo tres posibles ubicaciones para el procesamiento: el borde de la red, la niebla y la nube. El presente artículo propone un patrón arquitectónico que permite guiar la selección del punto de despliegue de aplicaciones basadas en algoritmos de inteligencia artificial con base en las necesidades de la solución tecnológica desarrollada. El patrón se obtuvo utilizando como metodología el Patrón de Investigación Iterativa de Pratt. Se usó un ejemplo del mundo real y se aplicó el paso a paso propuesto para demostrar la utilidad del patrón. Se concluyo que la selección de la ubicación del procesamiento debe tener en cuenta las necesidades del usuario final y las limitaciones que se puedan presentar.

Palabras clave: Internet de las Cosas, Patrón Arquitectónico, Web de las Cosas, Ingeniería de software, Inteligencia Artificial.

1. INTRODUCCIÓN

In recent years, there has been increasing interest in the development of IoT ecosystem solutions, composed of smart objects that connect to create interoperable services. One of the main obstacles in creating these systems is the high heterogeneity, i.e., the vast number of technologies and protocols used by different manufacturers [5].

Additionally, the solution developer in this environment must select a location to perform the processing (and analysis) of the information provided by the intelligent objects. The application may have different advantages and disadvantages depending on the selected location.

There are three deployment points to perform processing: the Edge, the Fog, and the Cloud [6][7]. The Edge can process the acquired data at the point of origin if the IoT application has the processing and communication capabilities in the smart device itself [8]. In the Fog there is a single centralized device responsible for processing (and analyzing) data from different endpoints in the network, i.e., it takes data from smart devices located at the Edge [7]. In the cloud, there are servers with the highest processing, storage, and user authentication capabilities to which the data from the different capture points should be reported [6].

The main advantage of Edge computing over Fog and Cloud computing is in the response times, which can be reduced because the processing is done locally as soon as the smart device acquires the data. In contrast, the main disadvantage is the limited hardware capabilities compared to Cloud-hosted servers. On the other hand, the advantage that computing in the Cloud presents over computing in the Fog is in the data processing and storage capacity, having as a disadvantage a greater difficulty in accessing the data provided by the smart objects [9]. Given the above, IoT application designers have drawbacks when deciding where to deploy the different data processing algorithms in the network. They usually make the decision empirically, which in many cases generates delays in the development of solutions or, in the worst case, applications that do not fully meet the functional or efficiency requirements [10][11]; the latter are of particular concern in IoT applications that in most cases are solutions that must act in real-time, have a low memory consumption, perform the right amount of processing to save battery and use low bandwidth.

For this reason, it is important to find a tool that supports IoT application developers to make the best decision in defining the location of the deployment of processing algorithms (including data analysis) within an Intelligent Objects Ecosystem of Web of Things (IOEoWoT), seeking to take advantage of the benefits proposed by the three locations previously mentioned.

In this context, the present study proposes an architectural pattern that allows defining the deployment point of a computational intelligence algorithm in a solution that is framed in an ecosystem of smart objects of the web of things to guide the developer in the solution that best fits the requirements of the IoT application.

The paper is organized as follows: Section 2 describes the methodology used based on the Iterative Research Pattern proposed by Pratt [12]. Section 3 introduces the concept of architectural pattern for IoT and then details the architectural pattern. Section 4 presents an example of using the pattern. Section 5 presents the discussion, conclusions, and future work the research group expects to develop on the topic in the short term.

2. METHODOLOGY

The architectural pattern proposed in this paper was obtained using Pratt's Iterative Investigation Pattern (IIP) as a methodology [12]. The IIP stands out for being an iterative and incremental process in which four stages are developed in each iteration: the observation of the problem, the identification of the problem, the development of a solution to the problem, and the evaluation of the developed solution. Three iterations were required to define the proposed pattern, which was learned and changed until the version presented in the following section of this document was obtained.

In the first iteration, we sought to experimentally compare the performance obtained when running artificial intelligence algorithms (fuzzy logic algorithms, genetic algorithms, among other tests) in different locations of intelligent ecosystems in the IoT, testing between devices with different hardware capabilities in each location (for example, in the Edge an ESP8266 card was used as the device with the lowest computational capacity and the Raspberry Pi 4 card as the device with the highest computational capacity). It was noted that, to guide the choice of location it would be convenient to present the user with a comparison of the response time obtained when executing the same algorithm in different locations, as well as to identify which devices do not have the necessary capabilities to carry out the processing of specific algorithms. The large number and variety of devices available to perform the proposed experiments and how artificial intelligence solutions should be implemented depending on the user's needs were identified as a research problem. After performing a performance comparison between devices in different locations with different artificial intelligence algorithms, it was concluded that a change in the way the project was being developed was necessary since the experimental approach was not feasible because it would require implementing an indeterminate and very large number of tests that could continue to grow in quantity and variety over time. It was also evident that little literature is available on implementing computational intelligence algorithms on Arduino-based boards.

In the second iteration, the experimental approach was changed, and a general solution was sought to guide the location of the processing depending on the user's needs. It was noted that to select the location, the functional and non-functional requirements of the application to be implemented (response time, effectiveness, availability of acquired data, storage, and processing capacity, security, and scalability) had to be identified. The research problem was identified as the user's need to delimit the location possibilities for the applications to be developed and how to select the best location among the possible solutions. In this sense, it was proposed to describe the usage scenarios of the pattern and its usage requirements. Taking as a reference the characteristics of a design pattern defined by Gamma et al. [13], together with the characteristics mentioned by Bloom et al. [14], the characteristics of the pattern were proposed, and a draft of the pattern was created in which the conditions to be met by the user to use the pattern were stipulated. The development was guided based on the CRISP-DM methodology [15]. When evaluating the pattern that had been developed so far with an expert, it was observed that it was necessary to improve its level of detail, to provide a guide that would offer greater clarity on the locations that are suitable for displaying the processing and that would allow choosing the most appropriate one for each scenario of use.

In the third iteration, the final version of the architectural pattern for the location of processing in IoT ecosystems was created, its components were described, and examples were made to test its usefulness.

3. PROPOSED ARCHITECTURAL PATTERN

Next, the concept of architectural patterns for IoT is introduced. Then, the components of the proposed processing location architectural pattern are presented, which are organized into context, problem, and solution.

3.1. Architectural Patterns for the IoT

For a solution to be considered a pattern, it must capture a common practice (which implies that it must have at least three known uses), and, at the same time, the pattern's solution must not be obvious. The description of architectural patterns is usually based on the context-problem-solution triplet, and this works synergistically in the context of a pattern language with numerous interdependencies with other patterns [16].

Within the IoT field, there are different categorizations for patterns, as can be seen in the article proposed by Washizaki [17], where a systematic review of the literature is carried out in which they seek to describe in a general way the current panorama of IoT design and architecture patterns, to identify deficiencies and suggest improvements when creating new patterns. Within this same article, patterns are classified depending on the level of abstraction, domain specificity, and non-functional requirement to be addressed.

Following the categorization proposed by Washizaki, it was determined that: 1) the level of abstraction of the proposed pattern is medium, seeking to ensure interoperability between heterogeneous devices, knowing the context of development needs, recurring problems, and their corresponding solution; 2) the domain specificity of the proposed pattern corresponds to general, since it applies to any IoT system; and 3) The non-functional requirements taken into account by the proposed pattern were response time, deployment and update of the model or algorithm, data acquisition, processing, security, mobility, energy consumption and deployment cost.

On the other hand, the proposed pattern includes all artificial intelligence approaches: fuzzy systems, neural networks, metaheuristics, machine learning, and expert systems [18].

3.2. Context

The pattern is aimed at developers implementing an IoT application. It seeks to guide the choice of the processing location (including data analysis) using artificial intelligence algorithms in the cloud, fog, or edge. It is necessary to comment that a basic level of experience of the user of the pattern is required to answer correctly (according to the needs of the application being developed) the questions that must be solved and that support the decision of the location of the IoT application/solution being developed.

Within an ecosystem of smart objects of the Web of Things, there are different non-functional requirements (security, response time, connectivity, scalability, among others) on which the design of the solution to be implemented can be focused. Depending on this approach, the decision can be made to perform the processing at a location within the ecosystem [19].
On the other hand, depending on the artificial intelligence technique (machine learning, neural networks, expert systems, fuzzy logic, and metaheuristics) that needs to be implemented to solve a specific problem in a specific application, it is necessary to use hardware components with minimum processing and storage capacities to be able to perform the required task.

Given the complexity of striking the right balance between non-functional requirements and the processing objectives of the application, this pattern supports the decision-making of the most appropriate location (Edge, Fog, or Cloud), considering the specific needs of the IoT application to be deployed.

Depending on the processing location within an ecosystem of smart objects, there will be advantages and disadvantages from the end user's point of view. In the Edge, the response speed is higher than in other locations, and there are greater mobility capabilities and energy autonomy. In the Fog, there are superior user authentication capabilities, which influence the application's security, superior storage capabilities, and ability to concentrate data from smart devices on the Edge, among others. Finally, in the Cloud, there is a greater processing capacity, optimization, and simulation [19]. Table 1 compares the three possible locations with some non-functional requirements.

An example of a motivating scenario to use the present pattern could be an IoT application focused on translating sign language to text using machine vision and artificial intelligence algorithms, specifically, a neural network. In this example, it is difficult for the application designer to select the best location for processing the data acquired by the sensor (in this case, the hand positions captured with the camera) since the data capture and training of the neural network may take much longer (if at all possible) on a device located at the Edge (such as an Arduino board) compared to a device in another location. Additionally, in the Cloud, an internet connection will be needed to consume the necessary processing services with the advantage of higher processing speed and a possible disadvantage in connection latency.

Table 1: *Location advantages and disadvantages*

Requirement	Cloud	Fog	Edge
Latency	Disadvantage	Neutral	Advantage
Distance between location and devices	Disadvantage (Far from the edge)	Neutral (Near the edge)	Advantage
Storage	Advantage	Neutral	Disadvantage
Computing power	Advantage	Neutral (device dependent)	Disadvantage
Mobility capacity	Disadvantage	Neutral	Advantage
User authentication	Advantage	Neutral	Disadvantage

Source: *own elaboration*

3.3. Problem

Where should a software application based on artificial intelligence be placed in an architecture that contemplates the Edge, Fog, and Cloud in a WoT smart object ecosystem?

3.4. Solution

Before defining the location of the algorithm to be used (Edge, Fog, or Cloud), the developer must perform the algorithm's training. In the case of solutions based on expert systems, fuzzy logic, and metaheuristics, although there is no phase called training, the developer must build the rules or algorithms based on the experience of the experts of the domain or the specific problem being solved, which is like training a model but is not done automatically from data. The training is usually performed at the location with the highest processing capabilities available, where the necessary tools for its development are also available. The developer must consider that the model obtained from the training, in the case of this example, a neural network, can be located at the selected deployment site.

Once the prior training of the algorithm has been performed, the user (developer or architect of the solution) must evaluate the viability (feasibility) of each location (Edge, Fog, or Cloud) for the implementation of the solution. Then, if two or more locations are considered viable, they must be evaluated according to the solution context to determine the best alternative.

Given the above, a set of questions was proposed to define the viability (feasibility) of locating the processing in each location, as shown in Table 2 (see attachments). The questions should be answered with YES or NO. It starts by evaluating the feasibility of locating the solution at the Edge; if one of the answers is negative, this location will be defined as unfeasible to perform the processing. This process is repeated for the location in the Fog and then for the Cloud.

Once the viable locations have been defined, the next step is to determine the best alternative. To achieve this, the non-functional requirements of the application (response time, latency, scalability, ease of training the artificial intelligence model and its update, ease of data acquisition, processing capacity, interoperability, autonomy, and security) must first be weighted. Table 3 (see attachments) shows in its first column the non-functional requirements to be considered and in the second column their corresponding importance, which should be scored regardless of the location where the application will be deployed. The importance is a subjective evaluation, which depends on the identified requirements of the IoT solution to be developed. It is up to the experience of the developer and the requirements engineer to correctly assess this importance.

Subsequently, Table 4 (see attachments) presents the evaluation of the non-functional requirements. In this table, the same score of the importance of the requirements that were previously assigned in Table 3 should be copied, and a rating will be given to each of the complete questions that appear in the first column (according to the location to be evaluated) with a value on a scale from 0 to 5 (where 0 refers to the worst performance and 5 to the best performance) based on the results that the algorithm is expected to provide when executed in a given location, This rating is assigned based on the previous experience of the application developer (which can be acquired by performing experiments on the devices in the locations or consulting with experts) or based on reports found in the literature, which in both cases establishes an estimated rating.

Once the score for each viable location is obtained, the weighting (importance) of each non-functional requirement is multiplied by its rating; these values are added and finally divided by the sum of the weightings to obtain the final rating for each location. The location with the highest score is selected as the most appropriate location (1).

$$Decision = ArgMax_{u_J \in U}\left(\sum_{r=1}^{8}\left(I_r \times Cal_{r,u_J}\right) \Big/ \sum_{r=1}^{8} I_r\right) \quad (1)$$

Where, U corresponds to the locations (Edge, Fog or Cloud), u_J corresponds to each of the locations, r Index of each of the eight non-functional requirements, I_r Importance given to a non-functional requirement according to Table 3, Cal_{r,u_J} Rating of the non-functional requirement r in the location u_J according to Table 4.

The non-functional requirements are explained in detail below:

Response time: It is the time resulting from the sum of data acquisition time, processing, response delivery, and latency. To assess this requirement in each location, the developer should ask himself whether, with the devices he has available and the range that these devices have, he can meet an appropriate response time for the user.

Deployment and updating of the model or algorithm: If the application to be located is based on a model that, in addition to being deployed, must be updated with a certain periodicity, it must be clear that this task can be performed in the selected location (effectiveness), but also that the development of these two processes can be performed with software resources, hardware, time and cost, which is by the project or company's budget (efficiency).

Data Acquisition: This is an estimated rating of how efficient it is to collect (acquire) the data required to run the AI-based application. Evaluating this efficiency may include collection time from different sources and energy consumption with the different resources available at the location.

Processing: The available processing capacity (processor, RAM, and external storage) at the location will allow obtaining the result of the execution of the model or algorithm in a time that is within the range of what is expected by the user.

Security: The location where the application is going to be hosted has devices that come with integrated security tools; these are correctly configured, the passwords for access to the devices are initialized and robust, the network definition protects from unauthorized access to the devices, the devices have their firmware, operating system and other applications updated, protected and are constantly updated, the functionalities or features that are not used in the devices are disabled; This ensures that the application is installed in an environment where security is at the level required by the organization that will use it.

Mobility: The user has sufficient capacity to move around the selected location to use the application and develop the task for which it was designed.

Energy consumption: When the application is installed in the selected location, it will have enough energy capacity to be used as long as necessary, or it will need batteries or other energy consumption mechanisms.

Cost: Purchasing, replacing, or renting updated equipment and software at the selected location corresponds to the budget established for the application deployment.

4. EXAMPLE OF USE OF THE PROPOSED PATTERN

The following is an example of the decision to locate an application that performs image processing for automatically selecting defective parts in a production line using the proposed pattern. In that example, there is a camera (smart device) that transmits the captured images to a computer (hub device) with an updated operating system with security patches, with an 8th generation Intel Core i5 processor, without a graphics card, with 4GB of RAM, in a factory, which has processing services in the Cloud for its intranet and extranet. The developers consider that the application should be developed with a deep neural network.

Following the indications of the solution, the developer must perform training to obtain the neural network model to be implemented. Once a model that will be compatible with the three possible locations is available, the feasibility of locating the processing in those three locations is evaluated. Table 5 (see attachments) below lists the data required for the application and the location where this data is housed. This information is maintained for all three possible locations. The guiding questions and actions to answer the question are shown in Table 2.

Next, Table 6 (see attachments) evaluates the feasibility of deploying the application on the Edge. As can be seen, this location is not feasible and is therefore discarded.

Next, in Table 7 (see attachments), the assessment of the feasibility of deploying the application in the Fog is performed. Therefore, this location is feasible and passes to the non-functional requirements qualification phase.

Next, in Table 8 (see attachments), the assessment of the feasibility of deploying the application in the Cloud is performed. Therefore, this location is viable and passes to the non-functional requirements qualification phase.

Thus, it is known that the viable locations are the Fog and the Cloud. Table 3 and Table 4 determine which locations will be the best alternative to locate the IoT application processing. The evaluation of the importance is shown below in Table 9 (see attachments).

The result of the evaluation in the Fog example is shown below in Table 10 (see attachments). As can be seen, the rating does not match the importance of the processing. However, the other non-functional requirements match adequately.

The evaluation result in the Cloud of the example below is in Table 11 (see attachments). As can be seen, the deployment and update of the model or algorithm and the data acquisition must match the importance assessment required for these non-functional requirements.

Finally, a rating of 4.38 points is obtained for Fog, surpassing the 3.77 points rating of Cloud, which indicates that for the application to be implemented, Fog is the best alternative.

In the evaluation of the pattern, two proofs of concept were developed and presented to a focus group composed of experts in agile methodologies and experts in IoT, agile development methodologies, and programming environments in IoT; however, these are not included in this article so as not to exceed the length limit defined by the journal. If required, they can be requested via e-mail to the corresponding author.

5. DISCUSSION, CONCLUSIONS AND FUTURE WORK

During the development of the pattern, it was possible to understand that the selection of the processing location (including data analysis) is not limited to the hardware capabilities of a location. However, it is necessary to analyze the needs of the end-user and the limitations that may arise concerning response time, model or algorithm deployment and update, data acquisition, processing, security, mobility, energy consumption, and cost.

The pattern proposed in this study helps guide the choice of processing in intelligent ecosystems when using artificial intelligence algorithms, as evidenced in the example.

Based on the identified limitations of the proposed pattern, future work to be developed is presented below:

- Compare whether the location selected using the proposed pattern corresponds to the experimentally selected location.
- Extend the scope of the pattern to guide the location of the processing of multiple algorithms or multiple functionalities within a single solution.
- Add other non-functional requirements within a new standard version, such as scalability, interoperability, portability, other aspects of maintainability, performance (the system must handle the required number of users without degrading performance), availability, and compatibility. Also, include the analysis of the semantic context in the IoT.

5. ACKNOWLEDGMENTS

To the Universidad del Cauca for partially financing the research.

REFERENCES

[1] P. Sethi and S. R. Sarangi, "Internet of Things: Architectures, Protocols, and Applications," *J. Electr. Comput. Eng.*, vol. 2017, p. 9324035, 2017, doi: 10.1155/2017/9324035.

[2] G. Bai, L. Yan, L. Gu, Y. Guo, and X. Chen, "Context-aware usage control for web of things," *Secur. Commun. Networks*, vol. 7, no. 12, pp. 2696–2712, 2014, doi: 10.1002/sec.424.

[3] A. Wagner, J. L. V. Barbosa, and D. N. F. Barbosa, "A model for profile management applied to ubiquitous learning environments," *Expert Syst. Appl.*, vol. 41, no. 4 PART 2, pp. 2023–2034, Mar. 2014, doi: 10.1016/j.eswa.2013.08.098.

[4] L. Yao, "A Propagation Model for Integrating Web of Things and Social Networks," in *Service-Oriented Computing - ICSOC 2011 Workshops*, G. Pallis, M. Jmaiel, A. Charfi, S. Graupner, Y. Karabulut, S. Guinea, F. Rosenberg, Q. Z. Sheng, C. Pautasso, and S. Ben Mokhtar, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 233–238.

[5] M. Niño Zambrano, "Interacción Semántica de Objetos en la Web de las Cosas," 2013.

[6] N. B. Ruparelia, *Cloud computing*. Mit Press, 2016. [Online]. Available: https://books.google.com.co/books?hl=es&lr=&id=umIsDAAAQBAJ&oi=fnd&pg=PR5&dq=Definition+of+Cloud+Computing&ots=k8BtuSpQ29&sig=FyIvnPN0V2lO12qQlPRrSekcaMk&redir_esc=y#v=onepage&q=Definition of Cloud Computing&f=false

[7] Z. Mahmood and M. Ramachandran, "Fog Computing: Concepts, Principles and Related Paradigms," in *Fog Computing: Concepts, Frameworks and Technologies*, Z. Mahmood, Ed., Cham: Springer International Publishing, 2018, pp. 3–21. doi: 10.1007/978-3-319-94890-4_1.

[8] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile Edge Computing: A Survey," *IEEE Internet of Things Journal*, vol. 5, no. 1. Institute of Electrical and Electronics Engineers Inc., pp. 450–465, Feb. 2018. doi: 10.1109/JIOT.2017.2750180.

[9] C. Mechalikh, H. Taktak, and F. Moussa, "A Scalable and Adaptive Tasks Orchestration Platform for IoT," in *2019 15th International Wireless Communications Mobile Computing Conference (IWCMC)*, 2019, pp. 1557–1563. doi: 10.1109/IWCMC.2019.8766744.

[10] B. Rababah, T. Alam, and R. Eskicioglu, "The Next Generation Internet of Things Architecture Towards Distributed Intelligence: Reviews, Applications, and

Research Challenges," *SSRN Electron. J.*, 2020, doi: 10.2139/ssrn.3640136.

[11] M. Aazam, S. Zeadally, and K. A. Harras, "Offloading in fog computing for IoT: Review, enabling technologies, and research opportunities," *Futur. Gener. Comput. Syst.*, vol. 87, pp. 278–289, Oct. 2018, doi: 10.1016/J.FUTURE.2018.04.057.

[12] K. S. Pratt, "Design Patterns for Research Methods: Iterative Field Research," in *AAAI Spring Symp. Exp. Des. Real*, 2009, pp. 1–7.

[13] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, "Design Patterns: Abstraction and Reuse of Object-Oriented Design," 1993, pp. 406–431. doi: 10.1007/978-3-642-48354-7_15.

[14] G. Bloom, B. Alsulami, E. Nwafor, and I. C. Bertolotti, "Design patterns for the industrial Internet of Things," *IEEE Int. Work. Fact. Commun. Syst. - Proceedings, WFCS*, vol. 2018-June, pp. 1–10, Jul. 2018, doi: 10.1109/WFCS.2018.8402353.

[15] P. Chapman *et al.*, "CRISP-DM 1.0: Step-by-step data mining guide," 2000.

[16] P. Avgeriou and U. Zdun, "Architectural Patterns Revisited - A Pattern Language.," 2005, pp. 431–470.

[17] H. Washizaki, S. Ogata, A. Hazeyama, T. Okubo, E. B. Fernandez, and N. Yoshioka, "Landscape of Architecture and Design Patterns for IoT Systems," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 10091–10101, Oct. 2020, doi: 10.1109/JIOT.2020.3003528.

[18] A. Dounis, "Artificial intelligence for energy conservation in buildings," *Adv. Build. Energy Res.*, vol. 4, pp. 267–299, 2010, doi: 10.3763/aber.2009.0408.

[19] J. Queiroz, P. Leitão, J. Barbosa, and E. Oliveira, "Distributing Intelligence among Cloud, Fog and Edge in Industrial Cyber-physical Systems," in *ICINCO*, 2019.

[20] "(PDF) Internet of Things (IoT): Definitions, Challenges, and Recent Research Directions." https://www.researchgate.net/publication/320532203_Internet_of_Things_IoT_Definitions_Challenges_and_Recent_Research_Directions (accessed Nov. 27, 2022).

ATTACHMENTS

***Table 2:** Rule base*

Guiding Question	Actions to answer the question.
GQ1: Is the data required to provide the service available at the location, or can it be acquired from the location?	A1: Make a list of the data and the data attributes required for training and use of the application (This list is the same for all locations; do not repeat it).
	A2: Define where each of these data is located (This list is the same for all locations; do not repeat it).
	A3: Ensure each data is available from the location evaluated as a deployment option.
GQ2: Are there tools for developing artificial intelligence processing and analysis, such as frameworks or libraries with the required implementations according to the selected AI approach (machine learning, neural networks, expert systems, fuzzy logic, or metaheuristics)?	A4: Based on the type of AI implementation, list the frameworks or libraries that will support the deployment of the solution available at this location and that have similar ones in the training, development, or modeling environment to be used.
GQ3: Does the device have the hardware capabilities to perform the processing (processor and RAM), store the data, and connect to other data sources if required (network)?	A5: Make an inventory of the devices available at this location to be used as deployment points for the AI-based implementation. Suppose the company does not have the devices and will be acquiring them. In that case, a review of which ones provide the best capabilities (processing, memory, storage, power consumption, etc.) will have to be done within the budget constraints of the company or project.
	A6: Ensure that each device has the capabilities (processing, memory, storage, network) to use the frameworks and libraries available for the application deployment.

Source: *own elaboration*

***Table 3:** Importance assessment by requirement*

Non-functional requirement	Importance (0 least important and 5 most important)
Response time	
Deployment and updating of the model or algorithm	
Data acquisition	
Processing	
Security	
Mobility	
Energy consumption	
Cost (scored based on available resources)	

Source: *own elaboration*

***Table 4:** Evaluation of non-functional requirements*

Non-functional requirement with associated question	Importance	Location rating (Edge, Fog, or Cloud) from 0 to 5	Weighted Rating
Is the response time of the solution at this location appropriate for the developer?			
Is deploying and updating the model or algorithm at this location efficient and effective for the developer?			
Can data acquisition be done from this location, and can it be developed efficiently here?			
With the available processing power, is the response time appropriate for the task?			
Is the security provided by the location sufficient to meet the application requirements?			
Does the mobility capability of the devices in the location meet the user's needs? [20]			
Does the power consumption of the devices at the location meet the application's needs? [20]			
How feasible is it to use the location based on its deployment cost?			
CUMULATIVE TOTAL	Sum of the importance rating		Sum of weighted score
RATING			Sum of weighted rating / Sum of importance rating

Source: *own elaboration*

***Table 5:** Description of data*

GQ1	
A1	Data required: Images of the products of the production line. Data required for training: Images and identifiers of the products with and without defects with which the model will be trained. Required attributes: Name of the image with its status (defective or correct) and image dimensions.
A2	The data will be acquired using a photograph, which must be formatted appropriately. Training data is obtained from a company database.

***Source:** own elaboration*

***Table 6:** Assessment of the feasibility of deployment at the Edge*

GQ2	
A3	Yes, the data is available at the location.
GQ3	
A4	Integrated development environment (IDE) for computers such as VSCode or PyCharm (used for model training). Integrated development environment for Raspberry Geany (Used for model training). Usually, this option is not recommended. Libraries: OpenCV, Scikit-image, PIL/pillow, NumPy, Mahotas
GQ4	
A5	Smart device: camera
A6	The device does not have the hardware capabilities to perform data processing and storage. There are no frameworks or libraries for development on the device manufacturer's proprietary operating system.

***Source:** own elaboration*

***Table 7:** Assessment of the feasibility of deployment in the Fog*

GQ2	
A3	The data is not at the location but is easily acquired by connecting the camera to the computer.
GQ3	
A4	Integrated development environment (IDE) for computers such as VSCode or PyCharm (for training and model execution). Libraries: OpenCV, Scikit-image, PIL/pillow, NumPy, Mahotas
GQ4	
A5	Desktop computer, with an updated operating system with security patches, with 8th generation Intel Core i5 processor, without a graphics card, with 4GB of RAM.
A6	The device has the necessary hardware capacity to carry out data processing and storage and the ability to connect to different devices on the network. Frameworks are available to develop the required processing algorithm.

***Source:** own elaboration*

***Table 8:** Assessment of the feasibility of deployment in the Cloud*

GQ2	
A3	The data can be sent to the cloud because it has internet connectivity, so the information from the computer to the Fog can be transferred.
GQ3	
A4	Programming environment Google Colab, PyCharm, VSCode, etc. (For training and model execution). Libraries: OpenCV, Scikit-image, PIL/pillow, NumPy, Mahotas.
GQ4	
A5	It depends on the servers contracted in the specific cloud (AWS, GCP, Azure, etc.), or hosting.
A6	The location has the necessary capabilities to perform the processing. Tools are available to carry out the processing.

***Source:** own elaboration*

***Table 9:** Evaluation of importance by requirement*

Non-functional requirement	Importance
Response time	4
Deployment and updating of the model or algorithm	5
Data acquisition	5
Processing	4
Security	1
Mobility	1
Energy consumption	1
Cost	5

***Source:** own elaboration*

***Table 10:** Example of evaluation of non-functional requirements in Fog*

Non-functional requirement with associated question	**Importance**	**Location rating (Edge, Fog, or Cloud) from 0 to 5**	**Weighted Rating**
Response time	4	5	20
Deployment and updating of the model or algorithm	5	5	25
Data acquisition	5	5	25
Processing	4	3	12
Security	1	5	5
Mobility	1	1	1
Energy consumption	1	1	1
Cost	5	5	25
CUMULATIVE TOTAL	26		114
RATING			4,38

***Source:** own elaboration*

***Table 11:** Example of evaluation of non-functional requirements in the Cloud.*

Non-functional requirement with associated question	**Importance**	**Location rating (Edge, Fog or Cloud) from 0 to 5**	**Weighted Rating**
Response time	4	4	16
Deployment and updating of the model or algorithm	5	3	15
Data acquisition	5	3	15
Processing	4	5	20
Security	1	5	5
Mobility	1	1	1
Energy consumption	1	1	1
Cost	5	5	25

CUMULATIVE TOTAL	26		98
RATING			3,77

Source: *own elaboration*