Enhancing Smart Agriculture Scenarios with Low-code, Pattern-oriented functionalities for Cloud/Edge collaboration

1st Georgios Fatouros
Dept. of Digital Systems
University of Piraeus
Piraeus, Greece
Innov-Acts Ltd
Nicosia, Cyprus
0000-0001-6843-089X

2nd George Kousiouris
Dept. of Informatics and Telematics
Harokopio University
Athens, Greece
0000-0003-0961-3471

3rd Theophile Lohier
CybeleTech
Montrouge, France
theophile.lohier@cybeletech.com

4th Georgios Makridis
Dept. of Digital Systems
University of Piraeus
Piraeus, Greece
0000-0002-6165-7239

5rd Ariana Polyviou
Dept. of Management
University of Nicosia
Nicosia, Cyprus
Innov-Acts Ltd
Nicosia, Cyprus
0000-0002-9808-5422

6th John Soldatos
Innov-Acts Ltd
Nicosia, Cyprus
0000-0002-6668-3911

7th Dimosthenis Kyriazis
Dept. of Digital Systems
University of Piraeus
Piraeus, Greece
0000-0001-7019-7214

***Abstract*—The integration of cloud computing and Internet of Things (IoT) technologies has brought significant advancements in the agriculture domain. However, the implementation of such systems often requires significant time and resources, making it challenging for smart agriculture providers to offer optimized yet affordable services for small and medium-sized farmers at scale. Low-code development platforms can be a viable solution to address these challenges, enabling non-experts to adapt or enhance existing applications with minimal coding. This paper presents a low-code approach to enhance smart agriculture scenarios with pattern-oriented functionality blocks for cloud/edge collaboration. It highlights the usage of a pattern collection for redesigning the implementation of smart agriculture applications that can enhance the data collection process as well as real-time decision-making and efficient resource management in the continuum. The effectiveness of the presented approach is demonstrated through the implementation of a case study in smart agriculture greenhouses. Evaluation results show that this approach can significantly reduce the time and effort required to deploy smart agriculture applications and provide data resilience.**

***Index Terms*—smart agriculture, low code, cloud patterns, data resilience, IoT, function as a service**

I. Introduction

Agricultural greenhouses are considered the most prominent way to control plant environments and increase their production, reduce the impact of climate uncertainty, provide physical barriers to diseases, as well as substantially reduce chemical pesticides [1]. Over the last decades, the global area under greenhouses has significantly expanded towards addressing the increasing demand for nutrition [2], which is expected to be increased 30% by 2050 [3]. In this direction, various novel applications have emerged, leveraging recent advances in data analytics and Internet of Things (IoT) systems to increase and optimize greenhouses production. Notably, Allied Market Research reports that the global IoT in agriculture market size was valued at $27.1 billion in 2021 and is projected to reach $84.5 billion by 2031 [4], Such applications are part of a novel paradigm called smart agriculture or smart farming [5].

Smart agriculture applications related to greenhouses include the optimization of production and crop quality, the prediction of microclimate, and the reduction of energy consumption and costs [6]. To enable the realization of such applications, it is necessary to collect and analyze large amounts of data since multiple parameters need to be set in a greenhouse, based on the type of plants being grown, the climate in the region, and the specific needs of the plants. For instance, more than 200 parameters can be set in a standard soil-less glasshouse for tomato cultivation [7]. Some important general parameters to consider include temperature, humidity, light intensity and duration, CO2 concentration, water availability, nutrient content of fertilizer solutions, and soil pH [8]. However, in traditional farming these parameters are set to default values or updated only sporadically based on season-specific and site-specific trials [9].

Towards proper parameter setting in greenhouses, smart agriculture solution providers leverage environmental and spa-

Part of the research leading to the results presented in this paper has received funding from the European Union's funded Project PHYSICS under grant agreement no 101017047.

tial data as well as real time data from IoT sensors placed in the greenhouse to develop models for fine-grained crop management and yield estimation [10]. The models utilize all available data, including historical data, to predict the optimal set of parameters for each crop/greenhouse at a given time and provide recommendations on how these parameters should be changed in real time. These recommendation and decision support models are usually deployed on a cloud-based platform that integrates with sensors, irrigation systems, and other IoT devices [11]. The latter facilitate the collection of real time data from the greenhouse and placing it in the cloud-based application for processing by the developed models. In addition, a first-level pre-processing or lightweight analytics are usually performed at the edge to avoid expensive data transfer from the farm to the cloud while optimizing the systems' latency. Likewise, serverless computing, such as Function-as-a-Service (FaaS) [12], can potentially bring significant benefits to smart agriculture applications by enabling cost-efficient and scalable deployment of decision support models and analytics at the edge or in the cloud [13].

However, implementing, maintaining, and adapting such systems often require significant time, resources, and expertise in cloud computing, making it challenging for smart agriculture providers to offer affordable services for small and medium-sized farmers. For instance, the data collection process in such hybrid pipelines is subject to various limitations, including the network connectivity of the IoT devices [14]. As a result, ensuring that the data collected from the greenhouse is accurate and complete becomes a tedious task, as data loss can occur during unstable network connections. Hence, the decision support models are often unable to provide high-quality recommendations on how best to manage greenhouse operations in real-time. Furthermore, adaptation of the solution to different technologies or protocols used between different greenhouses creates a burden for on-boarding new customers and adapting the code to each individual sensor system. Collaboration and work splitting between edge and cloud can also be complicated and depend on runtime parameters of the system and according availability of each site.

To address these challenges, this paper explores the usage of the PHYSICS platform [15], a relevant low-code environment based on the popular Node-RED framework that supports code packaging and serverless deployment, as well as a collection of various functionality patterns [16] in a real-world smart agriculture use case. This collection includes patterns for parallelizing execution, monitoring performance, performing extract-transform-load (ETL) data collection, and aggregating requests that can be used to enhance existing scenarios involving agricultural greenhouses. The patterns are implemented as Node-RED (sub)flows, enabling their abstract reuse, adaptation, and generalized deployment, even in resource-constrained environments such as edge devices. The work investigates how these patterns can extend the existing smart agriculture application architecture and what is the benefit of their inclusion. The results indicate an 86% faster onboarding time for a new greenhouse while achieving zero data loss.

The remainder of this paper is organized as follows: Section II provides background information on IoT and Cloud-based systems in agriculture and discusses the challenges in data collection and processing. Section III presents an IoT-enabled real-world smart agriculture use case as well as the incorporation and adaptation of the patterns in the specific domain in a before and after approach. Section IV discusses the evaluation and derived results from the application use case, focusing on data loss prevention and onboarding efficiency. Finally, Section V concludes the paper and discusses possible future work.

II. Background and Related Work

This section elaborates on related works that discuss the significance of edge computing in smart agriculture, the obstacles faced in data collection and processing, and the use of Node-RED as a valuable tool in developing efficient IoT and edge computing solutions.

A. IoT and Cloud-based Systems in Smart Agriculture

Edge computing is a paradigm that focuses on moving computational processing and storage closer to end-users, devices, or sensors, rather than relying exclusively on cloud-based solutions. This approach is particularly relevant in smart agriculture, which involves automation and control systems, sensors, and monitoring devices distributed across various locations. These components need to efficiently communicate with computing and database servers typically hosted on the cloud. In this direction, various platforms support the hybrid deployment of applications by placing some workflows at the edge and others on the cloud according to user requirements [17].

Although some scenarios involve data flowing directly from IoT devices in the field to the cloud, several smart agriculture applications and services generate substantial volumes of data that require real-time processing. This can result in increased network load, extended response times, and subpar quality of service due to limited bandwidth [18]. To address these limitations, smart agriculture applications introduce an edge layer between the IoT sensors and the cloud. This layer is responsible for preprocessing data and sending only aggregated summaries to the cloud, thus reducing the cloud's load and improving Quality of Service (QoS) through reduced latency, as it is closer to the devices [19], [20].

Hence, the collaboration of IoT, edge computing, and cloud-based systems in smart agriculture enables the development of more sophisticated predictive models and decision support systems. These systems can provide actionable insights and recommendations to farmers, enhancing their decision-making process and ensuring that they can respond promptly to changing conditions and potential risks in the agricultural environment [21].

B. Challenges in Data Collection and Processing

Smart agriculture faces numerous challenges, including sensing, data gathering, and device management to adapt to real-world farming conditions. Furthermore, data storage and

processing are critical issues that encounter certain obstacles [22]. Conventional methods for data storage, organization, and processing are not viable due to the immense volume of collected data. To address this, [23] proposes the collaborative use of cloud and edge computing in smart agriculture, aiming to reduce latency and costs while supporting QoS through load balancing of data operations between the edge and the cloud.

The study conducted by [24] recognizes that the broader adoption of smart agriculture is hindered by factors such as cost-effectiveness and limited internet accessibility. Remote farms often depend on satellite internet, which can be prohibitively expensive for transferring large volumes of data and may also be unreliable. [25] proposes the use of serverless functions to enhance smart agriculture systems with a fine-grained pricing model. Furthermore, [26] mentions that harsh environmental conditions can cause communication loss between the field and the server or cloud, resulting in data loss or traffic and congestion.

OpenFog [27], a public-private ecosystem formed to solve bandwidth, latency and communications challenges associated with the IoT, highlights the need to remove the requirement for persistent connectivity between edge/fog nodes and the cloud to address many of today's emerging scenarios including smart agriculture use cases.

Various works aim to address connectivity issues between IoT devices and cloud/edge computing services since the IoT network is a crucial component of IoT applications in the agricultural sector. It assists in monitoring agricultural data and facilitating the transmission and reception of this information [28]. The Routing Protocol for Low Power and Lossy Networks (RPL) is regarded as the primary protocol for implementing routing on the 6LoWPAN (IPv6 over low power wireless personal area network) stack [29], which is a key part of IoT-based solutions [30]. RPL utilizes Destination Oriented Directed Acyclic Graphs (DODAG) to define routes for various traffic flows and adapt to network speed based on routing metrics such as battery status, link quality, and increased computational cost exchange.

The authors of [31] contend that the adoption of smart agriculture necessitates digitization and hyper-connection processes, which require expertise in various fields. This includes appropriate training and the capacity of certain stakeholders to adapt to and utilize technologies that enable the management of the entire complex system. As a result, the operations required by farmers, software engineers, and IT personnel should be minimal and straightforward.

C. Node-RED in IoT and Edge Computing

Node-RED [32] is a widely used, open-source visual programming tool that simplifies the process of wiring together hardware devices, APIs, and online services for IoT applications. In the context of edge computing, Node-RED can be employed as a lightweight, versatile middleware layer that enables the efficient processing and management of data generated by IoT devices [33]. By deploying Node-RED on edge devices such as Raspberry Pi, BeagleBone, or other single-board computers, developers can create custom data processing logic using a flow-based programming paradigm. This approach facilitates the rapid development and deployment of edge computing solutions, catering to the diverse needs of smart agriculture applications.

The combination of Node-RED with edge computing offers significant benefits for IoT-based applications. For example, Node-RED has been exploited in various industrial applications such as smart factories, homes [34] and cities [35] as well as e-health use cases performing data collection, real-time processing, and transmission.

Furthermore, Node-RED's user-friendly, visual interface allows system developers to easily design, implement, and modify data processing workflows, adapting to the ever-changing requirements of smart agriculture applications. As a result, Node-RED serves as a powerful tool in the integration of IoT devices and edge computing technologies and can foster the development and evolution of advanced smart agriculture solutions.

III. Edge/Cloud Patterns for Smart Agriculture

This section initially presents a smart agriculture use case implemented by CybeleTech[1], along with the respective challenges faced by the smart agriculture platform. Additionally, it elaborates on the application and adaptation of generic Edge/Cloud patterns in the case of smart agriculture, including aspects of data collection, simulation execution and monitoring.

A. Smart Agriculture Digital Twin Use Case

CybeleTech offers greenhouse modeling solutions that significantly improve crop management through accurate yield and quality estimations of the crops, enabling farmers to make informed decisions. This improvement is achieved by developing a digital twin of the greenhouse that simulates the growth and development of crops [31]. This digital twin can then be fed with real-time data from sensors installed in the greenhouse, as well as meteorological data, allowing it to predict how a crop will behave under various conditions.

In this smart agriculture use case, the data used by the digital twin consist of daily production management data provided by the farmer, in addition to intra-hour climate data. Specifically, the platform receives around 30 climate variables every 10 to 60 minutes from the greenhouse sensors monitoring temperature and humidity spatial heterogeneity. Then, the platform needs to process 500K to 1M simulations per day on each greenhouse to manage meteorological uncertainty and correct its trajectory with existing historical data. These simulations, constructing the digital twin of the greenhouse, can accurately represent its biology and environment, producing increased emergence (i.e., low input variation can have a significant output impact).

However, the industrialization of this use case requires the management and operation of multiple and complicated

[1]https://www.cybeletech.com/en/home/

pipelines. For instance, some modeling equations are solved in a numerical simulation approach, with each simulation requiring 1 to 5 seconds to be completed and roughly 10 MB of storage. Moreover, the system should be robust to network connectivity breakouts between the greenhouse and the platform running on the cloud. Thirdly, the system should be scalable, allowing for the convenient and timely onboarding of new greenhouses to the platform.

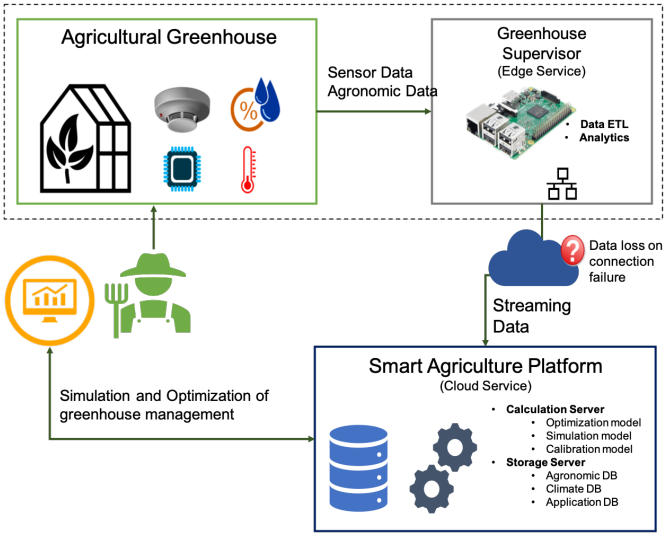

Fig. 1. Overview of the Smart Agriculture Digital Twin use case.

Currently, the greenhouse's digital twin use case relies on a hybrid (cloud/edge) approach. The computationally intensive simulations are performed in the cloud while data extraction from the sensors, transformation, and loading to the cloud-based database of the platform are performed on edge devices, such as Raspberry Pi 3/4 placed in the greenhouses. Figure 1 depicts CybeleTech's digital twin use case. However, the current setup lacks redundancy in connection failures between edge and cloud services, has limited scalability as it relies on typical IaaS, and requires extensive manual effort to add new greenhouses to the platform. These shortcomings result in poor QoS and increased costs for farmers.

To deal with these issues, the presented use case leverages several low-code patterns and the PHYSICS platform, which allows their timely parameterization through functional programming and scalable deployment through the FaaS cloud paradigm (based on OpenWhisk [36] open source FaaS platform). The patterns analyzed in the following subsection address critical aspects of the cloud/edge continuum. In addition, FaaS enables the deployment of functions (e.g., simulation or analytics tasks) as independent, event-driven, and autonomous entities with automatic scaling and fine-grained cost models, thereby improving the QoS and reducing costs.

B. Applicable Low Code Pattern Subflows for the Smart Agriculture Digital Twin

This section demonstrates how the generic patterns and subflows created within the PHYSICS project can be applied

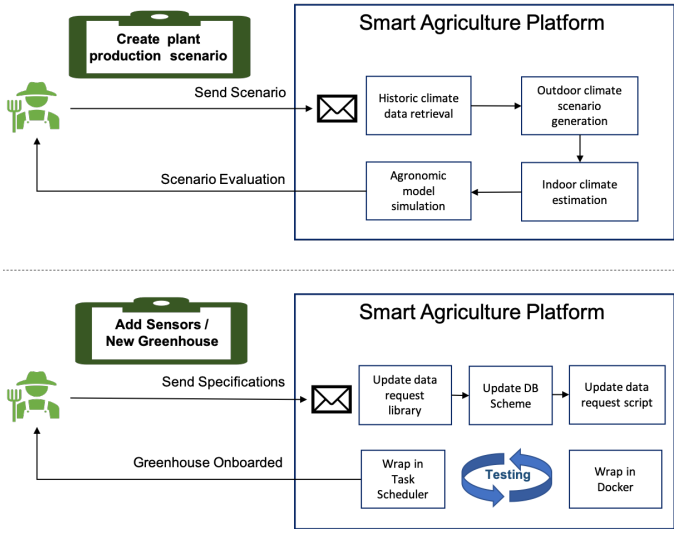

Fig. 2. AS-IS Smart Agriculture indicative use cases.

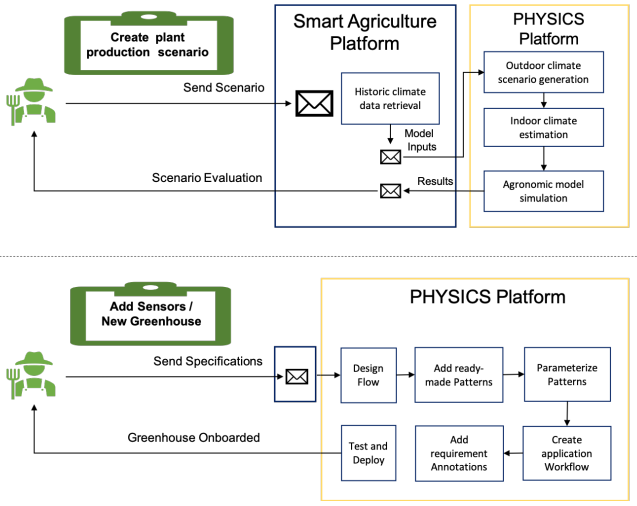

Fig. 3. Enhanced Smart Agriculture indicative use cases.

in the case of Smart Agriculture with Figure 4 illustrating the deployment overview of these patterns.

1) Transformation of the execution model to a functional programming style: As mentioned previously, the Smart Agriculture case applies typically a data collection process, followed by a data analysis part that is rather resource consuming. The latter is typically performed through High Performance Computing (HPC) code run on central cloud services. If the central infrastructure is for some reason not reachable, this creates an unavailability for the overall service. Furthermore, it creates issues maintaining the code as well as finding HPC experts and infrastructures for adapting it and executing it.

For this reason, a change to a more mainstream programming and execution model can help modernize and support the analysis application, without losing the benefits of scalability of the computations. The FaaS paradigm seems to be a good fit in this case, given that the agricultural simulation

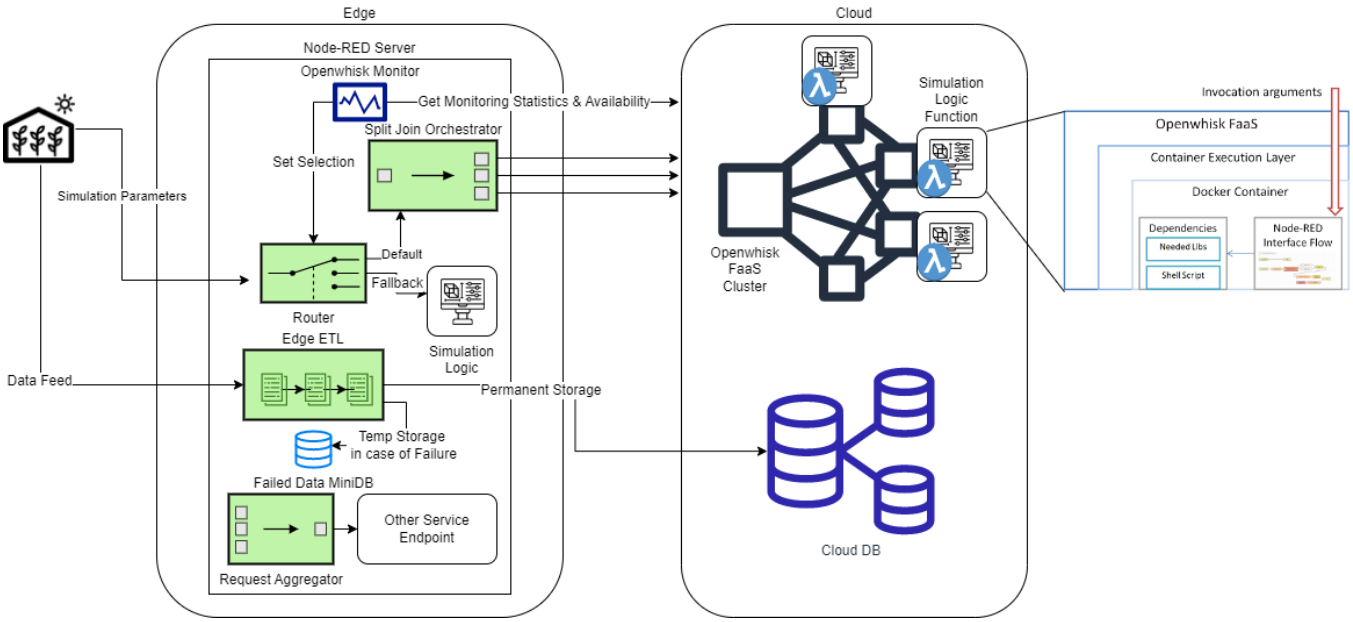

Fig. 4. Patterns enhancing Smart Agriculture.

is performed through numerous small individual executions of typically 1 to 5 seconds, without dependencies between the individual executions. Figures 2 and 3 depict the process view for migrating two indicative smart agriculture use cases (i.e., plant production scenario evaluation and onboarding of new greenhouse) to a functional-based execution model offered by PHYSICS.

In order to migrate the logic in FaaS, two layers of implementation are needed. Initially an orchestrator logic needs to exist, that will take the needed input (typically an array of values) and will undertake to split the work into smaller chunks (or parts of the array). For each chunk it will call an endpoint that provides the simulation logic per chunk and upon finalization of all parts it will re-assemble the overall output. For applying this, a relevant pattern is the Split Join parallelized execution [37].

In order to achieve maximum scalability, the simulation code can be encapsulated into a function and run on a central cloud FaaS platform. Thus for every split chunk, a separate function container will be launched, effectively parallelizing the execution. To adapt the simulation logic in a function format, a Node-RED function wrapper (OW Skeleton pattern [38]) can be used. The simulation logic can be executed either within the Node-RED environment (i.e. as Javascript code) or through calling a relevant shell script. The latter approach is better for migrating existing code and not having to re-implement it in Javascript.

Node-RED follows an event driven execution model, so a number of different functionalities can be applied as flows at the edge level, without consuming resources. Upon need, one or more of these functionalities can be triggered.

2) High Availability (HA) Scenario: One benefit of using Node-RED as the execution environment is that the created simulation flow can be executed as is in both ways, either as an OpenWhisk function, if registered on a relevant central cloud FaaS environment, or as a local service flow on the edge. This creates a number of possibilities, such as ensuring continuation of the offered functionality in cases of network or central cloud resources failures. In that case, a direct fall back on a local execution can be applied, although at a smaller scale (less detailed simulation, less parameters etc).

In applying this HA scenario, the FaaS time window monitoring pattern subflow [39] can be used. Through this, a number of metrics are obtained, such as average wait time, initialization time, execution time, success percentage etc. of the functions in the defined time window. Through this, a relevant routing logic can be applied, sending by default the requests in the central location and only using the local execution if the performance metrics of the former are not satisfactory or if the central location is unavailable.

3) Cost-Performance Tradeoffs: Depending on the available environment, one can regulate the split size applied on the SplitJoin pattern in order to enable more or less parallelization. For example, small split size results in more function containers (or service calls) to the simulation logic. This in general favors faster central execution but at higher costs. Also if the central resource is not that large, or if the execution is done at the edge, a smaller split size needs to be applied in order to avoid concurrency overheads from multiple containers or requests [40].

4) Reliable Data Collection at the Edge: As seen in Section 2, one of the main risks when operating at the edge is the unavoidable loss of connectivity at one or more points in time. This creates significant gaps in the data collection process, which also affects the quality and estimation error of the simulation models. To this end, the Edge ETL pattern

subflow [41] can be used. This tries to push data to the central repository on the cloud for a number of parametric retries. If that fails, then the data are stored locally and a general retry is performed in a cron job fashion, avoiding duplicate entries as well. A detailed analysis of the Edge ETL pattern can be found in [15], while an evaluation of its functionality within the given use case can be found in Section IV.

5) Request Aggregator Pattern: When the execution needs to fall back on the edge due to network failures, it is important that the device is not congested with a high number of relatively lightweight requests. In many cases the computational cost of serving each individual request may be higher than the actual useful computation needed inside the service. Furthermore, the division of the device resources in more than one concurrent requests may create bottlenecks and context switches. In the case of I/O frameworks, like Node.js (on top of which Node-RED is built), requests may not raise concurrent threads but they still may need to wait in the queue of the event loop for previous requests to finish, if the service implementation does not exploit the async nature of the framework. In all these cases, grouping together the incoming requests into one large batch can help alleviate the burden, as shown in [42]. The Request Aggregator [43] is the opposite of the SplitJoin pattern, so these two should not be used in sequence in the simulation logic. However the Request Aggregator can be used for other service endpoints that need to be offered on the edge.

IV. Evaluation and Results

This section analyzes the results and performance metrics used in the preliminary evaluation of the proposed pattern-based redesign of smart agriculture scenarios. The presented results focus on assessing the use of the ETL pattern for reliable data collection at the edge in smart agriculture. The evaluation of the other low-code patterns presented above is ongoing.

A. Performance Metrics

In order to assess the effectiveness of the proposed ETL Pattern, CybeleTech has established a set of key performance indicators (KPIs) that encompass various aspects of the system's performance. These KPIs are designed to evaluate the ETL Pattern's ability to minimize data loss, maintain the availability of service, and efficiently onboard new greenhouses to the platform:

- KPI#1 - Amount of data lost: The data collection pipeline currently deployed in greenhouses is highly sensitive to connection failures. When these failures occur, the data measured by the sensors and exposed by the greenhouse supervisor (i.e., edge service) are lost. This KPI measures the effectiveness of the ETL Pattern in reducing or eliminating data loss during network interruptions.
- KPI#2 - Update on plant status: The collected sensor data feed CybeleTech's plant growth model, which provides outputs to growers through visualization tools. These tools enable tracking of hourly changes in plant status and support decision-making in greenhouse management. In case of connection failures, environmental data become unavailable, and plant status updates cannot be provided, leading to service interruption. This KPI measures the duration of unavailability of service due to such interruptions.
- KPI#3 - Onboarding Efficiency: The onboarding process for new greenhouses on the platform may require several adjustments depending on the infrastructure of each greenhouse. This KPI measures the required effort, in person-days, to onboard a new greenhouse to the platform, showcasing the efficiency and adaptability of the ETL Pattern in accommodating different greenhouse setups.

B. Experimental Results

CybeleTech's experiments provide useful results on the performance of the proposed ETL Pattern on the basis of the predefined KPIs. The experiments were conducted in real-world scenarios, considering different types of greenhouses and simulated connection failures. The outcomes highlight the effectiveness of the ETL Pattern in improving data collection, onboarding efficiency, and maintaining service availability.

Specifically, the baseline evaluations of KPI#1 and KPI#2 were obtained by monitoring two types of greenhouses belonging to R&D partners of CybeleTech. Monitoring was conducted from October 2021 to June 2022 in the first greenhouse and from July 2021 to September 2021 in the second greenhouse. As shown in Table I, during the monitoring period six service interruptions occurred for greenhouse Type 1, with interruption duration ranging from 2 to 12 days and 27.4k data points lost. For greenhouse Type 2, three interruptions occurred with interruption duration ranging from less than one day to four days causing a loss of 8k data points. The total interruption duration is 38 days out of eight months (around 15%) for greenhouse Type 1 and four days out of two months (around 6.5%) for greenhouse Type 2.

On contrary, to evaluate the two first KPIs, the existing script for sensor data parsing was adapted and integrated into the Edge ETL pattern. This pattern was then parameterized and deployed on CybeleTech's infrastructure, acting as a greenhouse supervisor. Several simulations of connection failures were performed, with their durations ranging from 1 hour to 1 day by randomly deactivating the internet connection. After 2 weeks of experimentation, no data ware lost due to connection failure.

As a reference for evaluating KPI#3, a coarse estimation of the effort invested in adapting through PHYSICS the data collection procedure from a Type 1 to a Type 2 greenhouse was used. The results of this evaluation are shown in Table II which reports the time (measured in person-days) required to adapt and deploy the code scripts implementing the ETL process to a new greenhouse. These results show that depending on the data structure's complexity and the file format, the complete adaptation time ranged from a few minutes to a few hours when leveraging the proposed pattern. This is significantly less

TABLE I
KPI#1 & KPI#2 EVALUATION

Period	Number of interruptions	Interruption time[a]	Data Lost[b]
Greenhouse Type 1			
October 2021	0	0	0
November 2021	1	7	5000/40 kB
December 2021	1	4	2900/23 kB
January 2022	1	8	5800/46 kB
February 2022	1	12	8600/69 kB
March 2022	1	2	500/11 kB
April 2022	0	0	0
May 2022	1	5	3600/29 kB
Total	6	38	27400/220 kB
Greenhouse Type 2			
July 2021	2	< 1	500/4 kB
August 2021	1	3	7500/59 kB
Total	3	4	8000/63 kB

[a]Measured in days.
[b]Measured in data points/Kilobytes.

time ($\sim$86%) than what was required to perform the same adaptation with the existing implementation. The main reason for this improvement is that the Edge-ETL pattern eliminates the need to develop and configure bash wrappers while it can be directly deployed on the edge as a Docker container.

TABLE II
KPI#3 EVALUATION

Task	AS-IS[a]	With ETL Pattern[a]
Script for data collection and storage	1-2	< 1
Script for connection failure handling	3-5	0
Scripts for task scheduling	3-5	0
Deployment procedure	4-6	< 1
Total	11-18	< 2

[a]Measured in person-days.

Overall, the integration of the proposed ETL pattern at the edge layer of the smart agriculture use case enabled to achieve the objective regarding the data collection pipeline with no data lost during the experimentation period. Moreover, the time needed to adapt the data collection pipeline to new greenhouses is significantly lower with the presented approach allowing also a reduction in costs for the growers. The objectives and results of the KPI evaluations are summarized in Table III. It is also noted that the ETL pattern could be exploited in different use cases and sectors where resilient data collection from IoT devices is required.

TABLE III
KPIS OBJECTIVES AND RESULTS

KPI	Objective	AS-IS	With ETL Pattern
Amount of data lost	No data lost no matter how many connection failures occur	500 to 8000 data points lost/month	0
Update on plant status	No interruption of service	$\leq$10 days/month	0
Onboarding Efficiency	Onboarding time must be smaller than 2 person-days	$\sim$15 person-days	$<$2 person-days

V. CONCLUSION AND FUTURE WORK

This paper showed how a collection of adaptable low-code functional patterns implemented as Node-RED flows can enhance smart agriculture scenarios. Those patterns facilitating the parallel execution, performance monitoring, resilient data collection at the edge, and request aggregation can address various challenges of smart agriculture platforms, such as data loss, service high availability, scalability, and onboarding of new clients efficiently. By addressing these challenges, smart agriculture services could become more efficient in terms of performance and cost, alleviating the increasing nutrition demand.

In addition, the paper showcased experimental results of the ETL pattern for resilient data collection at the edge in the context of smart agriculture greenhouses. The results demonstrated the effectiveness of the ETL Pattern in significantly reducing data loss, maintaining service availability, and streamlining the onboarding process for new greenhouses. Based on the defined KPIs, the performance evaluation showed that this pattern eliminated data loss during connection failures, ensured no interruption in plant status updates, and reduced the onboarding time by about 86%.

The ETL Pattern's flexibility and adaptability make it a promising solution not only for smart agriculture use cases but also for other sectors requiring resilient data collection from IoT devices. Future work includes the evaluation of the other presented patterns in hybrid (cloud/edge) applications of the agricultural sector, as well as the development of additional patterns facilitating lightweight analytics at the edge for further enhancing the system's performance and capabilities.

ACKNOWLEDGMENT

Part of the research leading to the results presented in this paper has received funding from the European Union's funded Project PHYSICS under grant agreement no 101017047.

REFERENCES

[1] L. Lipper, P. Thornton, B. M. Campbell, T. Baedeker, A. Braimoh, M. Bwalya, P. Caron, A. Cattaneo, D. Garrity, K. Henry *et al.*, "Climate-smart agriculture for food security," *Nature climate change*, vol. 4, no. 12, pp. 1068–1072, 2014.

[2] I. Blanco, A. Luvisi, L. De Bellis, E. Schettini, G. Vox, and G. Scarascia Mugnozza, "Research trends on greenhouse engineering using a science mapping approach," *Horticulturae*, vol. 8, no. 9, p. 833, 2022.

[3] D. Tilman, C. Balzer, J. Hill, and B. L. Befort, "Global food demand and the sustainable intensification of agriculture," *Proceedings of the national academy of sciences*, vol. 108, no. 50, pp. 20 260–20 264, 2011.

[4] C. Samriddhi and D. Roshan. Iot in agriculture market by application: Global opportunity analysis and industry forecast, 2021-2031. [Online]. Available: https://www.alliedmarketresearch.com/internet-of-things-iot-in-agriculture-market

[5] X. Yang, L. Shu, J. Chen, M. A. Ferrag, J. Wu, E. Nurellari, and K. Huang, "A survey on smart agriculture: Development modes, technologies, and security and privacy challenges," *IEEE/CAA Journal of Automatica Sinica*, vol. 8, no. 2, pp. 273–302, 2021.

[6] A. Escamilla-García, G. M. Soto-Zarazúa, M. Toledano-Ayala, E. Rivas-Araiza, and A. Gastélum-Barrios, "Applications of artificial neural networks in greenhouse technology and overview for smart agriculture development," *Applied Sciences*, vol. 10, no. 11, p. 3835, 2020.

[7] J. Dyśko, M. Szczech, S. Kaniszewski, and W. Kowalczyk, "Parameters of drainage waters collected during soilless tomato cultivation in mineral and organic substrates," *Agronomy*, vol. 10, no. 12, p. 2009, 2020.

[8] A. Ali, T. Hussain, N. Tantashutikun, N. Hussain, and G. Cocetta, "Application of smart techniques, internet of things and data mining for resource use efficient and sustainable crop production," *Agriculture*, vol. 13, no. 2, p. 397, 2023.

[9] P. Oteng-Darko, S. Yeboah, S. Addy, S. Amponsah, and E. O. Danquah, "Crop modeling: A tool for agricultural research–a review," *Journal of Agricultural Research and Development*, 2013.

[10] K. G. Liakos, P. Busato, D. Moshou, S. Pearson, and D. Bochtis, "Machine learning in agriculture: A review," *Sensors*, vol. 18, no. 8, p. 2674, 2018.

[11] M. Danita, B. Mathew, N. Shereen, N. Sharon, and J. J. Paul, "Iot based automated greenhouse monitoring system," in *2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS)*. IEEE, 2018, pp. 1933–1937.

[12] S. Kounev, C. Abad, I. Foster, N. Herbst, A. Iosup, S. Al-Kiswany, A. A.-E. Hassan, B. Balis, A. Bauer, A. Bondi *et al.*, "Toward a definition for serverless computing," 2021.

[13] S. Trilles, A. González-Pérez, and J. Huerta, "An iot platform based on microservices and serverless paradigms for smart farming purposes," *Sensors*, vol. 20, no. 8, p. 2418, 2020.

[14] R. Montella, M. Ruggieri, and S. Kosta, "A fast, secure, reliable, and resilient data transfer framework for pervasive iot applications," in *IEEE INFOCOM 2018-IEEE conference on computer communications workshops (INFOCOM WKSHPS)*. IEEE, 2018, pp. 710–715.

[15] G. Kousiouris, S. Ambroziak, B. Zarzycki, D. Costantino, S. Tsarsitalidis, V. Katevas, A. Mamelli, and T. Stamati, "A pattern-based function and workflow visual environment for faas development across the continuum," ser. ICPE '23 Companion. New York, NY, USA: Association for Computing Machinery, 2023, p. 165–172. [Online]. Available: https://doi.org/10.1145/3578245.3584934

[16] "Physics pattern flows for cloud/edge and openwhisk," https://flows.nodered.org/collection/HXSkA2JJLcGA, 2022.

[17] G. Fatouros, Y. Poulakis, A. Polyviou, S. Tsarsitalidis, G. Makridis, J. Soldatos, G. Kousiouris, M. Filippakis, and D. Kyriazis, "Knowledge graphs and interoperability techniques for hybrid-cloud deployment of faas applications," in *2022 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*. IEEE, 2022, pp. 91–96.

[18] O. Friha, M. A. Ferrag, L. Shu, L. Maglaras, and X. Wang, "Internet of things for the future of smart agriculture: A comprehensive survey of emerging technologies," *IEEE/CAA Journal of Automatica Sinica*, vol. 8, no. 4, pp. 718–752, 2021.

[19] M. Saqib, T. A. Almohamad, and R. M. Mehmood, "A low-cost information monitoring system for smart farming applications," *Sensors*, vol. 20, no. 8, p. 2367, 2020.

[20] A. Koubaa, A. Aldawood, B. Saeed, A. Hadid, M. Ahmed, A. Saad, H. Alkhouja, A. Ammar, and M. Alkanhal, "Smart palm: An iot framework for red palm weevil early detection," *Agronomy*, vol. 10, no. 7, p. 987, 2020.

[21] G. Adamides, N. Kalatzis, A. Stylianou, N. Marianos, F. Chatzipapadopoulos, M. Giannakopoulou, G. Papadavid, V. Vassiliou, and D. Neocleous, "Smart farming techniques for climate change adaptation in cyprus," *Atmosphere*, vol. 11, no. 6, p. 557, 2020.

[22] V. P. Kour and S. Arora, "Recent developments of the internet of things in agriculture: a survey," *Ieee Access*, vol. 8, pp. 129 924–129 957, 2020.

[23] V. K. Quy, N. V. Hau, D. V. Anh, N. M. Quy, N. T. Ban, S. Lanza, G. Randazzo, and A. Muzirafuti, "Iot-enabled smart agriculture: architecture, applications, and challenges," *Applied Sciences*, vol. 12, no. 7, p. 3396, 2022.

[24] K. Paul, S. S. Chatterjee, P. Pai, A. Varshney, S. Juikar, V. Prasad, B. Bhadra, and S. Dasgupta, "Viable smart sensors and their application in data driven agriculture," *Computers and Electronics in Agriculture*, vol. 198, p. 107096, 2022.

[25] S. Palacios, D. Zabrocki, B. Bhargava, and V. Aggarwal, "Auditable serverless computing for farm management," in *Proceedings of the International Workshop on Big Data in Emergent Distributed Environments*, 2021, pp. 1–6.

[26] F. M. Ribeiro, R. Prati, R. Bianchi, and C. Kamienski, "A nearest neighbors based data filter for fog computing in iot smart agriculture," in *2020 IEEE International Workshop on Metrology for Agriculture and Forestry (MetroAgriFor)*. IEEE, 2020, pp. 63–67.

[27] Open Fog Consortium Working Group, "Openfog reference architecture for fog computing," OpenFog Consortium, White Paper, Feb 2017.

[28] M. S. Farooq, S. Riaz, A. Abid, K. Abid, and M. A. Naeem, "A survey on the role of iot in agriculture for the implementation of smart farming," *Ieee Access*, vol. 7, pp. 156 237–156 271, 2019.

[29] H. A. Al-Kashoash, H. Kharrufa, Y. Al-Nidawi, and A. H. Kemp, "Congestion control in wireless sensor and 6lowpan networks: toward the internet of things," *Wireless Networks*, vol. 25, pp. 4493–4522, 2019.

[30] G. Mulligan, "The 6lowpan architecture," in *Proceedings of the 4th workshop on Embedded networked sensors*, 2007, pp. 78–82.

[31] S. Cesco, P. Sambo, M. Borin, B. Basso, G. Orzes, and F. Mazzetto, "Smart agriculture and digital twins: Applications and challenges in a vision of sustainability," *European Journal of Agronomy*, vol. 146, p. 126809, 2023.

[32] OpenJS Foundation. Node-red. [Online]. Available: https://nodered.org/

[33] M. Blackstock and R. Lea, "Toward a distributed data flow platform for the web of things (distributed node-red)," in *Proceedings of the 5th International Workshop on Web of Things*, 2014, pp. 34–39.

[34] K. Ferencz and J. Domokos, "Using node-red platform in an industrial environment," *XXXV. Jubileumi Kandó Konferencia, Budapest*, pp. 52–63, 2019.

[35] G. Kousiouris, S. Tsarsitalidis, E. Psomakelis, S. Koloniaris, C. Bardaki, K. Tserpes, M. Nikolaidou, and D. Anagnostopoulos, "A microservice-based framework for integrating iot management platforms, semantic and ai services for supply chain management," *ICT Express*, vol. 5, no. 2, pp. 141–145, 2019.

[36] I. Baldini, P. Castro, P. Cheng, S. Fink, V. Ishakian, N. Mitchell, V. Muthusamy, R. Rabbah, and P. Suter, "Cloud-native, event-based programming for mobile applications," in *Proceedings of the International Conference on Mobile Software Engineering and Systems*, 2016, pp. 287–288.

[37] "Split join parallelized execution pattern subflow," https://flows.nodered.org/flow/7a5acfc999b1ad47bb32b5d37419c777/in/HXSkA2JJLcGA, 2022.

[38] "Node-red function interface pattern for openwhisk," https://flows.nodered.org/flow/d010a0a5af458e093342420349d63773/in/HXSkA2JJLcGA, 2022.

[39] "Openwhisk sliding window monitor subflow," https://flows.nodered.org/flow/a86475720659b3ed9eb5024052d94b1d/in/HXSkA2JJLcGA, 2022.

[40] G. Kousiouris and A. Pnevmatikakis, "Performance experiences from running an e-health inference process as faas across diverse clusters," in *Companion of the 2023 ACM/SPEC International Conference on Performance Engineering*, ser. ICPE '23 Companion. New York, NY, USA: Association for Computing Machinery, 2023, p. 289–295. [Online]. Available: https://doi.org/10.1145/3578245.3585023

[41] "Split join parallelized execution pattern subflow," https://flows.nodered.org/flow/3c1fbcd16dd405a12911049f121f3601/in/HXSkA2JJLcGA, 2022.

[42] G. Kousiouris, "A self-adaptive batch request aggregation pattern for improving resource management, response time and costs in microservice and serverless environments," in *2021 IEEE International Performance, Computing, and Communications Conference (IPCCC)*, 2021, pp. 1–10.

[43] "Request aggregator pattern subflow implementation," 2022. [Online]. Available: https://flows.nodered.org/flow/a97ec190bbf75cec594092895d39c01d/in/HXSkA2JJLcGA