On improving agricultural IoT management process for fault detection

Tien-Dung Nguyen a, Thach-Phat Chiem a, Trong-Hai Duong a, Duc-Hung Le b and Xuan-An Nguyen c

aSchool of Computer Science and Engineering, International University, Vietnam National University, Ho Chi Minh City, Vietnam; bDistributed System Group, TUWien, Wien, Austria; cHayabusa Research and Development Center, Ho Chi Minh City, Vietnam

ABSTRACT

In recent years, the emerging development of Internet of things (IoT) has created new opportunities for IoT service provider, device manufacturer and customer (end user). In agriculture, employing IoT can help reducing manpower as well as standardizing agricultural processes, thus increasing the benefit for customer who does agricultural business. Beside the opportunities, there are also issues while doing maintenance for an agricultural IoT system. One of them is how to automate the process to detect faults of IoT devices in a large deployment area. While previous studies focus on investigating algorithms as well as theoretical model to detect faults for specific IoT devices, automatically expanding of IoT management process for detecting faults of sensor/actuator has not been investigated yet. To achieve this, we should be able to deal with the relationship between IoT device manufacturer, service provider and customer. In this study, we conceptualize the knowledge of IoT devices, management process and fault detection requirement. To realize this model, we introduce a runtime system to expand the IoT management process for detecting sensor/actuator faults which is able to provide the above features. We do several experiments to show the effectiveness and usefulness of the proposed solution.

ARTICLE HISTORY

Received 1 November 2016
Accepted 28 February 2017

KEYWORDS

Internet of things; IoT management process; IoT fault detection; agricultural process improvement; process integration; process-based fault detection

Introduction

In the era of Internet of things (IoT), traditional agriculture, which requires a big amount of manpower, has been changing to adapt to the rise of these technologies. By doing so, the modern agriculture can take advantages of IoT devices and services to replace human in certain agricultural activities (Nukala et al., 2016). As a result, the management process of doing agriculture are standardized, manageable and improvable that making the reduce of overall operation cost and increase the quality of agricultural product. One major problem of applying IoT in agricultural management process is that the maintenance of IoT devices is difficult. A reason for that is the process is complex, therefore it is hard to...
find which devices have problems and what problems they have manually. The delay of solving the problems may lead to the delay of agricultural production and certainly losing benefit. Manually making changes to the current process for fault detection requires a lot of efforts (e.g. finding integration points in a process to add activities for fault detection and diagnosis). Moreover, the modified process can have mistaken and not able to be executed.

In order to reduce such problems, we investigate the common faults of IoT devices and their possible solutions. Then we define a model which stores knowledge of the IoT devices, an IoT management process and a fault detection requirement. The knowledge is provided by IoT device manufacturer, customer and IoT service provider. Based on this model, we propose an algorithm which receives the knowledge as inputs to modify the IoT management process. The modified one contains additional activities which supports detecting IoT device faults. We also propose a runtime system to evaluate the proposed algorithm. To this end, this paper has three main contributions:

- A model to store the knowledge of IoT devices, IoT management process and fault detection requirement.
- An algorithm to expand the IoT management process for fault detection.
- Ladybug runtime system for IoT service provider to offer fault detection service.

The rest of this paper is organized as follows: the second section presents motivation for our work. The third section presents common faults of IoT devices and possible solutions. The fourth section introduces the model to store knowledge of IoT devices, IoT management process and fault detection requirement. The fifth section presents an algorithm to modify the IoT management process to detect faults. The sixth section presents the Ladybug runtime system to provide fault detection service. The seventh section presents experiments. The eighth section discusses related work. Finally, we conclude the paper and discuss our future work in the ninth section.

**Motivation and approach**

**Scenario**

To motivate our research problems, we present a scenario in a context of Chinook wheat farming (Campbell & Read, 1968) as follow.

Using data of environmental factors such as light intensity, temperature and soil moisture which are optimized for farming Chinook wheat in a growth chamber, we raise a scenario by defining an IoT management process which can manage these factors as illustrated in Figure 1. The process manages three sensors and three actuators, as shown in Figure 2. The sensors are used to measure the environmental factors, while the actuators are used to make changes to the factors of the farming environment. For example, water sprayer can reduce air temperature, or light bulb can increase light intensity.

This process can be described as follows. At the beginning, the sensors are turned on (i.e. activities $a01$, $a05$ and $a08$) and checked whether their status change to
active or not (i.e. decisions $d_{01}$, $d_{05}$ and $d_{08}$). In case of inactive status, the process is stopped. In case of active status, sensor data are collected (i.e. activities $a_{02}$, $a_{03}$, $a_{06}$ and $a_{09}$) and checked whether their values are out of predefined-ranges or not (i.e. decisions $d_{02}$, $d_{04}$, $d_{06}$ and $d_{09}$). These predefined ranges are best values of the factors for wheat farming. In case the values are not in the predefined-ranges, the actuators are turned on in a specific amount of time to make these factors changing to the values in the predefined-ranges.

**Challenges**

As we can see from the above scenario, the IoT management process, which is a predefined process, does not contain activities for detecting faults of IoT sensors and actuators. There are two methods to add the activities for detecting faults into the predefined process.

The first method can be done by adding the validation activities at the beginning of IoT management process, which is when the system is powered on, and making them to repeat forever after an amount of time until the system is shut down. Although it is easy to modify the predefined process this way, it has several disadvantages. Because of executing these activities intensively, the IoT devices, which have micro-controller-units for low power consumption and small activities execution purposes, consume more energy and shorten their life expectancy.

The second method can be done by adding the validation activities when it is necessary (e.g. when turning on a sensor but its status remains inactive). Although this method can overcome the disadvantages of the first method, it requires significant efforts and time to find appropriate modified positions, as well as define connections between current activities and new added one. Moreover, the modified process is easier to have errors that makes it non-executable.

**Related work**

Many works have been proposed to solve the sensor fault detection and diagnosis problem. There are three main approaches in fault detection including sensor validation, integration fault detection modules into a decision support system (DSS) and process-based fault detection.

*Sensor validation* solves the problem by executing particular diagnosis applications which are developed by their manufacturers or third parties. Balaban et al. (2009)
Figure 2. IoT devices deployment.
proposed solutions to detect common faults of electromechanical actuators. Weimer et al. (2012) introduced a method for detecting faults of actuator in air conditioning. In our study, these applications can be developed as plugins for Ladybug to validate the sensor/actuator when it is necessary.

Integration fault detection modules into DSS solves the problem by collecting human knowledge, historical/online monitoring data and parameters to optimize fault detection by a DSS. Liu, Yang, Lv, and Wang (2013) defined a Statistics Sliding Windows to store sensor data, then regressed each window by Gaussian distribution, and used the regression to detect the fault value. Naidu, Zafiriou, and McAvoy (1990) compared the precision of neural network using back-propagation model and finite integral squared error for sensor failure detection. The result showed that neural network model was better than its competitor with the ability to capture nonlinear characteristics, the possibility for on-line training and speed. Another approach of sensor fault prediction is that discovering causal relationship between physical system using conditional probability analysis and a priori analysis, then using them to figure out failures (Wang, Vo, & Ni, 2015). Different from these approaches, we choose the process-based approach for detecting fault of an IoT agricultural system because the process is manageable, standardized and reusable. Moreover, the process also enables resource management at lower levels and separates the business logic from its execution.

Process-based fault detection solves the problem at service level. The IoT management process integrates activities for fault detection and diagnosis. Our work relates to business process improvement (Jalali & Leake, 2013; Meyer, Ruppen, & Magerkurth, 2013; Zhang, Zhou, & Gao, 2006). These works focus on improving a general business process based on different input criteria and process’ goals. However, these general approaches cannot be applied in agricultural IoT management process for fault detection due to lacking of domain knowledge of particular sensor and actuator and agricultural process. In our study, we make assumption that the domain knowledge is provided by device manufacturer and IoT service provider. To the best of our knowledge, a method using the concept of business process to study the auto modification of agricultural IoT management process has not been done yet.

Common faults of sensor/actuator

According to (Liu et al., 2013; Naidu et al., 1990; Wang et al., 2015) and the experiences from a Japanese robotic company – Hayabusa, we classified common faults of IoT objects (i.e. sensor or actuator) into three types:

- **Hardware failure**: IoT object does not respond to commands. The causes can be supply power is not stable or does not match manufacturer’s requirements, or IC legs are not fully connected, or the object is broken. This error can be detected by putting a voltage metre at the power source of the object to ensure the supply power in accordance with manufacturer’s requirements, then check the pins to find leaking welding legs.
- **Development library failure**: IoT object responds to electrical control, but does not respond to one or more embedded source code commands. The main reason is due to the connection does not follow the manufacturer standard or the I/O data are incorrect. A solution can be developing a new algorithm to retrieve data based on sensor datasheet.
• **Installation failure**: IoT object responds to electrical control, but does not respond to all embedded source code commands. This is caused by incorrect settings in accordance with the manufacturer’s instructions. A solution is to recheck installation instructions, and reset the object to factory setting if necessary.

**Knowledge for fault detection**

By embracing the concept of business management process, the IoT management process has abilities to monitor and control physical or virtual resources when operating IoT devices for cultivating (Meyer et al., 2013).

To be able to enhance farming efficiency, minimize errors, save resources and minimize power consumption. Although there exists process meta model for IoT (Domingos, Martins, Cândido, & Martinho, 2014; Meyer et al., 2013), they are generic meta models. Therefore, the process, which uses these models, becomes difficult for understanding and enhancing automatically in the context of agriculture. We propose a novel process meta model for agricultural IoT management process on top of web service business process execution language (WS-BPEL).¹

Figure 3 presents the model of IoT management process which consists of many IoT activities and decisions which implement extended activity of WS-BPEL. An IoT activity describes a controlling, monitoring or supporting actions for IoT objects, while a decision describes a set of conditions which results a Boolean value either YES or NO. Each activity can have several input and output links from other activities. Similarly, each decision can have several input links, but only one YES output link and one NO output link. These links

![Figure 3. IoT management process model.](image-url)
organize the IoT activities and decisions in sequential and parallel flows. An IoT activity can have several input/output parameters.

In the context of IoT in agriculture, we define seven types of IoT activities as follows:

- **Turn on**: used to turn on a sensor/actuator
- **Turn off**: used to turn off a sensor/actuator
- **Collect**: used to get a monitoring metric of a sensor or a parameter of an actuator
- **Validate**: used to execute a library provided by a manufacturer to validate possible errors of an IoT object class
- **Enrich**: used to find related information about an error code of a sensor/actuator, and who is responsible to handle this error
- **Notify fault**: used to send email about fault to a responsible person
- **Notify exception**: used to send conditions of IoT decision before validation activity and historical sensors/actuators data related to the conditions to manufacturer for further investigation

In order to automate the modification of the IoT management process, it is necessary to have a knowledge base which stores data related to IoT devices and requirements for fault detection. Figure 4 presents our proposed model (KDFault) for these purposes. Each IoT device can have several IoT objects. An object can have a type (i.e. sensor or actuator), a status (i.e. active or inactive), a list of parameters (e.g. a parameter of a light bulb is light intensity strength measured in percentage) and a validation library. This validation library is developed by the manufacturer and should be automatically updated via an IoT platform. By executing a library of an IoT object, we can understand what are possible errors, their causes and solutions. A fault detection requirement includes several fault detection conditions. A fault detection condition indicates which is related object, parameter, operator (i.e. =, <, >, <=, >=) and compared value. Listing 1 shows an example of fault detection requirement. This requirement includes a condition for a light intensity

---

![Figure 4. Knowledge model for fault detection pattern.](image-url)
sensor and a condition for a light bulb. This requirement means that if the value of light intensity sensor is less than 5000 and the status of the light bulb is active, there could be a fault related to the light intensity sensor or the light bulb and an in-depth validation should be taken place.

Algorithm to expand IoT management process

It is challenging to manually improve an IoT management process for farming to detect faults of IoT objects due to time constraint and human error. Algorithm 1 shows pseudo code to support to modify the predefined process by adding IoT activities/decisions and links at right positions automatically. The algorithm receives two inputs including a predefined process (preProcess) and a knowledge base for fault detection (KDFault), and return an expanded process. The algorithm can be divided into three parts:

- Adding IoT activities for validation (lines 2–8): the algorithm adds four activities including IoTActivityValidate, IoTActivityEnrich, IoTActivityNotifyFault and IoTActivityNotifyException, then generating input/output links between them.
- Fault validating after turn on/off IoT object (lines 9–21): the algorithm finds turn on/off activities, then get decisions that check the status of this object. If the status is inactive when executing turn on activity, or the status is active when executing turn off activity, there can be a problem and it is necessary for in-depth validation. To accomplish this, the algorithm adds a NO output link from the decision to the validation activity.
- Fault validating based on the fault detection requirement (lines 22–32): for each requirement, the algorithm creates a new decision which have all conditions in the requirement. This decision is used as signalling for early fault validation. The algorithm adds input/output links from the decision to the validation activity and the collect activities of the objects in the conditions.

Algorithm 1. Algorithm to expand the predefined-process for fault detection.
To offer the fault detection service for farming, we propose the Ladybug runtime system that implements the IoT process management, knowledge base for fault detection and the algorithm to expand the process for fault detection. Figure 5 shows the overall architecture of the Ladybug runtime system which consists of four components:

1. **Editor**: allows to edit the IoT management process, knowledge of IoT device as well as fault detection requirement.
2. **Core engine**: has core services including Orchestrator for managing the interactions between services, IoT process modifier implementing the proposed algorithm, and IoT process scheduler for executing IoT management process.
3. **IoT services**: this component has services provided by IoT activities. Device monitor collects streaming sensor data. IoT device controller controls the operations of actuators. Device validator allows to execute external library from manufacturer for detail error validation. Device enrichment allows to collect information related to the problem. Notification allows to send the message about the problem to a responsible person.
4. **Cloud storage**: this component has two storage services. Metadata database stores IoT management process and knowledge for fault detection. Distributed sensor database stores real-time data from sensors/actuators.

A predefined IoT management process will be uploaded to Ladybug system. Then the Orchestrator calls IoT process modifier to generate an extended management process.
which contains fault detection and diagnosis activities. After that, Orchestrator calls IoT process scheduler to execute the extended process. The execution of the process will check data, detect fault and control actuator automatically following the process flow.

The role of IoT services component is to provide services to collect sensor data; generate controlling actuator commands; validate, enrich and notify fault. Since IoT devices use local IP, Ladybug cannot communicate with them directly and have to wait for them to push sensing data to IoT device monitor and pull control commands from IoT device controller via RESTful web services.

We use two storage services. MySQL\textsuperscript{2} database is used for storing metadata, includes KDFault, IoT management process and fault detection requirement. Cassandra\textsuperscript{3} database is used for storing sensing data. The reason we choose MySQL for metadata because it does not need to modify frequently, small and required stable service. For Cassandra database, because the sensing data come in huge size with a very short time, we need to provide high bandwidth and distributed database to solve this problem.

**Evaluation**

In this section, we evaluate the proposed algorithm in the fifth section and the execution of the IoT management process using the case study in the second section.

**Modification of IoT management process for fault detection**

To evaluate the proposed algorithm, we use two inputs including the predefined IoT management process and the fault detection requirements, as shown in Figure 2 and Table 1, respectively. The output of the algorithm is an IoT management process which is able to detect faults specifying in the requirements.
The algorithm can be executed through three main steps to produce the output as an expanded process, as shown in Figure 6.

In the first step, the algorithm adds four activities into the process including IoTActivityValidate, IoTActivityEnrich, IoTActivityNotifyFault and IoTActivityNotifyException which have id $a_{11}, a_{12}, a_{13}$ and $a_{14}$. Then the algorithm adds input/output links between these activities.

In the second step, the algorithm finds turn on/off activities which are $a_{01}, a_{04}, a_{05}, a_{07}, a_{08}$ and $a_{10}$. Then, the algorithm adds a link from the decisions, which are used to check the status of the objects, to the IoTActivityValidate $a_{11}$.

In the third step, the algorithm uses the fault detection requirement in Table 1 to improve the IoT management process. For the requirement $req_1$, the algorithm gets

| reqId | condId | objectId | parameterKey          | operator  | comparedValue |
|-------|--------|----------|-----------------------|-----------|---------------|
| req1  | cond1  | i01      | light intensity metric | <         | 5000 lux      |
| req1  | cond2  | i02      | light bulb status      | =         | active        |
| req2  | cond3  | i03      | temperature standard deviation | >         | 3             |
| req3  | cond4  | i04      | soil moisture metric   | <         | 25%           |
| req3  | cond5  | i05      | water valve status     | =         | active        |

Figure 6. Expanded IoT management process for fault detection.
conditions in this requirement. These conditions, which involve to two metrics which are light intensity metric and light bulb status, are added into a new decision d11. The metric light bulb status and light intensity metric are collected after executing a04 and a02, respectively. Because these metrics are available after the decision d03 (i.e. both light intensity sensor and light bulb are turned on), the decision d11 is linked to a02 and a11. Similarly, for the requirement req3, the algorithm creates and adds the decision d13 into the process. This decision is linked to a09 and a11. For the requirement req2, the condition involves to the metric temperature standard deviation which is available after turning on the IoT object i03, and has not been collected in the process yet. Therefore, a new activity IoTActivityCollect a15 is added to the process to collect values of this metric after the decision d05. Then, the algorithm creates the decision d12 which has the condition cond3. This decision is linked to a15 and a11.

Execution of IoT management process

We deploy the Ladybug runtime system in Digital Ocean cloud. We create a synthesis data sent by the light intensity sensor, temperature sensor, soil moisture sensor, light bulb and water valve to simulate faults caused by the sensors/actuators. We develop a java app to send the data continuously to Ladybug with a frequency of 3s. We use Ladybug to execute the modified process resulted from the first experiment.

Figure 7(a) presents the monitoring values of two metrics light intensity and light bulb status. The horizontal axis presents time line in second within 9 minutes. The left vertical axis presents the value of light intensity sensor, and the right vertical axis presents the status of light bulb (i.e. 1 and 0 for active and inactive respectively). In the first 3 minutes, we see that the value of light intensity is higher than 17000 lux. We assume that the values of is day time is always true, so the decision d04 goes out at NO output link. At the fourth minute, the light intensity is simulated to decline to around 12000 lux gradually. As a result, the decision d04 goes out at YES output link making the light bulb to turn on. Therefore, the light intensity is kept at around 17000 lux until the sixth minute. From the seventh minute, we simulate the fault of the light intensity sensor which sends incorrect values of light intensity. As a result, the decision d11 goes out at YES output link for validating the sensor using the library provided by its manufacturer. The process detects a fault of light intensity sensor, enriches information of the fault and sends a message to a technician, as shown in Listing 2.

Figure 7. Monitoring sensors/actuators.
Figure 7(b) presents the monitoring values of the metric temperature standard deviation. The horizontal axis presents time line in s within 6 minutes. The left vertical axis presents the value of temperature standard deviation. In the first 3 minutes, we see that the value of temperature standard deviation is less than 0.1. It means that the change of values of temperature is minor and the temperature sensor is working well. At the fourth minute, the value of temperature standard deviation is simulated to increase to more than 0.3 suddenly. It means that the sensor may have some problems because it was sending inconsistency values of temperature within 3s intervals. The process detects a fault of temperature sensor, enriches information of the fault and sends a message to a technician.

Figure 7(c) presents the monitoring values of soil moisture and water valve status. The horizontal axis presents time line in s within 6 minutes. The left vertical axis presents the values of soil moisture sensor, and the right vertical axis presents the status of water valve (i.e. 1 and 0 for active and inactive respectively). In the first 3 minutes, we see that the value of soil moisture is above 65%. At the fourth minute, the values of soil moisture is simulated to decline gradually to less than 25%, which triggers the water valve to turn on. However, the process detects that the values of soil moisture sensor are not increased after turning on water valve. As a result, a fault of water valve is detected, enriched and notified to a technician.

In the experiments, we show that the proposed algorithm can be used to modify the IoT management process automatically for fault detection, and this process is executable using Ladybug.

Conclusions and future work

Automating the modification of the agricultural IoT management process for fault detection can save human resource and minimize human error. Moreover, the proposed solution helps reducing the time of determining sensor errors, controlling actuator automatically, updating faults database and handling exception faults.

Since service provider cannot communicate directly with sensors or actuators, sensor network is required to send sensor data and ask for controlling commands on the service provider frequently. It causes high network traffic for both users and service provider. Conceptually, the components can be deployed in separate virtual machines. We can have multiple instances of IoT Services for balancing loads by a load balancer – HAproxy. Another issue is that actuator waits for controlling command gather from API

Listing 2. A fault detection message of a light intensity sensor.

```json
{
    "error_description": "Sensor does not respond to commands",
    "error_type": "Hardware failure",
    "error_code": "1001",
    "solutions": [
        "Solution 1: Please check whether the link between your sensor and controller is connected or not. If not, please repair it and recheck the sensor status."
    ]
}
```
might cause some problems, for example, timeout, delay or wrong command due to the packet is modified during transmitting. Therefore, the actuator might not respond to the command immediately or respond wrongly. To handle this case, we suggest to use hardware redundancy strategy by integrate an additional sensor to confirm that status and captured data of the main sensor, which is used to capture data in real time, is correct. Moreover, the system requires user to provide their IoT management process, sensor data and their knowledge about suspicious sensor faults cases. These requests may rise security issues about privacy information, and intellectual property.

In future, we will improve the Ladybug system to an agricultural IoT platform. This platform will be integrated with other cloud-based platforms for better resources management and load balancing. Moreover, the proposed algorithm will be extended to deal with the IoT management process containing human activities.

Notes

1. http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html.
2. https://www.mysql.com.
3. http://cassandra.apache.org.
4. https://www.digitalocean.com.
5. https://wwwhaproxy.com.

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

This research is funded by International University – VNUHCM under grant number [T2016-04-IT].

Notes on contributors

Tien-Dung Nguyen is currently a researcher at the International University - Vietnam National University HCMC. His research interests include Data Mining, Internet of Things and Business Process Management. He has received his B.Sc. and M.Sc. degrees in Computer Science from the International University - Vietnam National University HCMC. He has been honored with the best student awards in both B.Sc. and M.Sc. programs. From 2013 to 2015, he had worked on the CELAR project, funded by the European Commission, at the Faculty of Informatics, Vienna University of Technology, Austria.

Thach-Phat Chiem is currently a full time iOS developer. He has received his bachelor’s degree in Computer Science from Ho Chi Minh City University of Technology, and master’s degree in Computer Science from the International University - Vietnam National University HCMC.

Trong-Hai Duong is currently a lecturer of the International University - VNU HCMC, Vietnam. He received M.Sc and Ph.D. degrees from the School of Computer and Information Engineering, Inha University, 2012. He received a Bachelor of Science with a certificate for the most excellent student at research from the Dept. of Informatics of Hue
University, Vietnam, 2004. He received the Golden Globe awards and Innovative Youth badge in 2016. He was one of 68 representative young scientists who had a meeting with the Prime Minister of Vietnam. He was one of 37 students of all Vietnam universities who won a scholarship certificate from the Ministry of Postal Service of Vietnam for the most excellent student of information technology, 2003. He has experience in Information Engineering, especially Artificial Intelligence. His research interest focuses on Semantics and Bigdata. He has more than 60 publications.

**Duc-Hung Le** completed his BSc and MSc at Hanoi University of Science and Technology. His previous research was about high performance and parallel computing. He started his new research as a research assistant at Distributed Systems Group from March 2013, which focuses on cloud computing. In particular, his research topic is about dynamic (re)configuration of virtual infrastructures and IoT cloud applications. Besides, he is pursuing the PhD program at Vienna PhD School of Informatics. Currently, he is working at TTTech Computertechnik AG as a DevOps Engineer.

**Xuan-An Nguyen** is currently a senior hardware engineer in Hayabusa Co., Ltd. He has been working in embedded system for four years. He has professional skills in both hardware and software of embedded systems. He has been developing many systems related to the Internet of Things and robots. He has received his Bachelor in Computer Engineering degree from the International University - Vietnam National University HCMC.

**ORCID**

Tien-Dung Nguyen [http://orcid.org/0000-0002-0531-5605](http://orcid.org/0000-0002-0531-5605)

Thach-Phat Chiem [http://orcid.org/0000-0002-9737-2198](http://orcid.org/0000-0002-9737-2198)

Duc-Hung Le [http://orcid.org/0000-0002-9237-4805](http://orcid.org/0000-0002-9237-4805)

Xuan-An Nguyen [http://orcid.org/0000-0002-7023-9354](http://orcid.org/0000-0002-7023-9354)

**References**

Balaban, E., Bansal, P., Stoelting, P., Saxena, A., Goebel, K. F., & Curran, S. (2009, March). A diagnostic approach for electro-mechanical actuators in aerospace systems. In 2009 IEEE Aerospace conference (pp. 1–13). IEEE.

Campbell, C. A., & Read, D. W. L. (1968). Influence of air temperature, light intensity and soil moisture on the growth, yield and some growth analysis characteristics of Chinook wheat grown in the growth chamber. *Canadian Journal of Plant Science, 48*(3), 299–311.

Domingos, D., Martins, F., Cândido, C., & Martinho, R. (2014). Internet of things aware WS-BPEL business processes context variables and expected exceptions. *Journal of Universal Computer Science, 20*(8), 1109–1129.

Jalali, V., & Leake, D. (2013). On deriving adaptation rule confidence from the rule generation process. In *International conference on case-based reasoning* (pp. 179–187). Berlin: Springer.

Liu, Y., Yang, Y., Lv, X., & Wang, L. (2013). A self-learning sensor fault detection framework for industry monitoring IoT. Mathematical problems in engineering.

Meyer, S., Ruppen, A., & Magerkurth, C. (2013). Internet of things-aware process modeling: Integrating IoT devices as business process resources. In *International conference on advanced information systems engineering* (pp. 84–98). Berlin: Springer.

Naidu, S. R., Zafiriou, E., & McAvoy, T. J. (1990). Use of neural networks for sensor failure detection in a control system. *IEEE Control Systems Magazine, 10*(3), 49–55.

Nukala, R., Panduru, K., Shields, A., Riordan, D., Doody, P., & Walsh, J. (2016). Internet of things: A review from ‘Farm to Fork’. In *Signals and Systems Conference (ISSC), 2016 27th Irish* (pp. 1–6). IEEE.
Wang, C., Vo, H. T., & Ni, P. (2015). An IoT application for fault diagnosis and prediction. In 2015 IEEE International Conference on Data Science and Data Intensive Systems (pp. 726–731). IEEE.

Weimer, J., Ahmadi, S. A., Araujo, J., Mele, F. M., Papale, D., Shames, I., & Johansson, K. H. (2012). Active actuator fault detection and diagnostics in hvac systems. In Proceedings of the fourth ACM workshop on embedded sensing systems for energy-efficiency in buildings (pp. 107–114). ACM.

Zhang, Y., Zhou, X., & Gao, Y. (2006). Optimizing the data intensive mediator-based web services composition. In Asia-Pacific web conference (pp. 708–713). Berlin: Springer.