ISSN 2089-8673 (Print) | ISSN 2548-4265 (Online)
Volume 12, Issue 2, July 2023

APPLYING USE CASE 2.0 APPROACH TO THE DEVELOPMENT OF IOT-BASED RAINFALL MONITORING SYSTEM

Mohammad Fajar[1], Ferian Bagus Chandra[2], Hamdan Arfandy[3]
[1,2]Teknik Informatika, STMIK Kharisma Makassar
[3]Teknik Informatika, Universitas Islam Makassar

email: fajar@kharisma.ac.id, talkactive@gmail.com, hamdanarfandy@gmail.com

Abstract
The use of Internet of Things (IoT) technology for monitoring and controlling environmental conditions or objects is quite popular. However, most of the development of the IoT systems including rainfall monitoring systems, mainly focuses on the implementation perspective, rather than discussing the development approaches or design techniques. The use of suitable development approaches will increase maintainability aspect of the IoT system in the future. Therefore, the aim of this study is to implement and evaluate the use case 2.0 approach in modeling and designing an IoT-based monitoring rainfall system. Data collection was performed through evaluation using object-oriented metrics to measure encapsulation, polymorphism, and reusability properties of the designed system. In modeling the IoT system, collected requirements specifications are organized into user stories. The user stories are then mapped into UML use case diagrams. Each of the use case should be sliced into thinner pieces, taking into account of the basic and alternative flows of the user stories. Moreover, the use case slices are designed, implemented, and evaluated independently. The results of modeling and designing a rainfall monitoring system using the use-case 2.0 are then implemented on the NodeMCU platform and Android-based application. Evaluation results show that the implementation of use-case Reading, Viewing and Searching for Rainfall Data can be run successfully on the target platform. The measurement uses object-oriented metrics on the designed IoT system indicating that the use case slices have an impact on the ease of system modification level.

Keywords : Use-Case 2.0, IoT, IoT Modeling, Rainfall Monitoring System, Object Oriented Metrics

Received: 14-05-2023 | **Revised:** 27-07-2023 | **Accepted:** 28-07-2023
DOI: https://doi.org/10.23887/janapati.v12i2.61529

INTRODUCTION

Currently, the Internet of Things (IoT) has become one of the most studied topics in the field of information and communication technology. IoT is a system that connects various devices such as sensors, actuators, and computers through the internet to perform specific tasks for users. It serves as a bridge between physical objects in the real world, such as kitchen appliances or vehicles, and virtual objects in the digital world, allowing them to communicate [1].

Many of the IoT systems or devices have been studied and developed by researchers in various fields. As example, in the field of agriculture, research [2], [3], [4], and [5] have developed IoT-based smart farming systems to monitor and control environmental conditions such as air temperature, soil moisture, and light. In the field of health, as studied by [6] and [7], IoT systems are utilized to monitor the conditions of the diabetic patients. The IoT technology was also applied by [8], for remote control system household electric appliances. The majority of the studies focuses mainly on the implementation issue of the IoT system, without describing and discussing specifically of how to use development approaches or methodologies. It is necessary to apply a certain development approach or a design methodology in order to handle system complexity and to increase maintainability, as well as to ensure the quality of the IoT system [9], [10].

Several studies related to the approaches, methodologies, and development tools for the IoT systems have been reviewed in the literature. Such as the use of Agile methodology for IoT application development and increasing its business value [11], the implementation of System Modeling Language for IoT application engineering [12], and its development based on service-oriented architecture (SOA) using the modeling language SoaML4IoT [13]. In the same way, study in [14] shows of how to employ UML to model IoT systems for monitoring and predicting power consumption. However, using the modelling language tools only is not enough, without specific methods or techniques to guide the modeling process of the system, including which diagrams need to be applied. Therefore, this research aims to implement and evaluate the use-case 2.0 approach to modeling and designing an IoT system. As a case-study, we develop a rainfall monitoring system that consist of mobile apps and IoT devices. The Use-case 2.0 is a extended version of the use-case driven approach based on the idea of user interaction with user stories and agile methodology, and it is an easy-to-use approach [15]. In this research, modeling is an activity that is carried out before implementation phase, by creating a more abstract model or form that can be further analyzed, so that system developers have a clearer understanding of what is being developed [9], [16]. The contribution of this work is to present of how to model and design of IoT systems, specifically rainfall monitoring system that increase maintainability using use case 2.0 approach.

METHOD

The stages in this research follow the system development life cycle scenario that begins with system requirements and analysis, design, implementation, and evaluation. The use-case 2.0 approach is applied as a guide in each of these stages, specifically in the requirements and analysis phase (modeling). In the evaluation step, the implementation code of selected use-cases that have been defined should be tested. Moreover, the quality aspects of the system are evaluated using object-oriented metrics to measure encapsulation, polymorphism, and reusability properties of two systems that is with and without use case 2.0 approach. The results of the evaluation are then used as the basis for drawing a conclusion of this research.

A. Implementation Scenario of Use-Case 2.0

In this paper, the scenario used in implementing the Use-Case 2.0 approach refers to the study [15], as follows:

1. Identify the use-case story of each use-case.
2. Slice each use-case into small pieces (use case slice).
3. Prepare the use-case slices.
4. Analyze the use-case slices.
5. Implement the use-case slices.
6. Test the use-case slices.

B. Evaluation Using Object Oriented Metrics

The level of modification is an important aspect in system development. As we know, information systems or software will always need to be modified or changed to fit the users' needs [17]. System components that are designed independently will make it easier for developers to modify the system's functionality without affecting other parts. In the evaluation process, system components with low dependencies (loose coupling) are important parameters for developing quality information systems or software. This is because the cost incurred from maintaining or modifying the system is seen as the largest cost of the overall system development [18]. In information technology field, one of the important tools for measuring code or software quality is object-oriented metrics [19].

The research conducted by [20] used object-oriented metrics to measure the level of modification of a model-driven development methodology, including the use-case driven approach. The measurement was done on the level of encapsulation, polymorphism, and reusability of a system.

Encapsulation

Study [21] proposed a formula to calculate the object-oriented metrics, where encapsulation can be measured using Method Hiding Factors (MHF) and Attribute Hiding Factors (AHF). In this paper, the MHF and AFH parameter defined formally is as follows:

$$MHF = \frac{\sum_{i}^{TC}\sum_{m=1}^{m_d(ci)}(1-V(M_{mi}))}{\sum_{i-1}^{TC}M_d(c_i)} \quad (1)$$

Where $M_d(c_i)$ is the number of methods declared in a class, and

$$AHF = \frac{\sum_{j=1}^{TC} is_{visible}(M_{mi},C_j)}{TC-1} \quad (2)$$

Where TC is the total number of classes, and

$$is_{visible}(M_{mi},C_j) = \begin{cases} 1 & if j \neq i \wedge C_j may call M_{mi} \\ 0 & otherwise \end{cases} \quad (3)$$

i is the iterator for calculations from class 1 to class TC of the system, if the system has class TC. M is the iterator for computation from the first method of class i of the system to the method $M_d(c_i)$ of class i if class i has an $M_d(c_i)$

method defined in that class. $V(M_{mi})$ is the number of classes that call method to m from class to i (besides class to i itself), divided by the total class of the system that has been reduced by 1.

Polymorphism
The degree of polymorhpism can be measured using the number of method overriden by a subclass (NMO) and polymorphism factor (PF) with equation 3 and 4.

$$NMO = Totalnumberof\ overridenmethods \in asubclass \qquad (4)$$

$$PF = \frac{\sum_{i=1}^{TC} M_0(C_i)}{\sum_{i=1}^{TC} [M_n(C_i) \times DC(C_i)]} \qquad (5)$$

Where $M_d(C_i) = M_n(C_i) + M_0(C_i)$

and

$M_n(C_i) = thenumberofnewmethods,$
$M_o(C_i) = thenumberofoverridingmethods,$
$DC(C_i) = thedescendantcount$

Reusability
Reusability aspects of the system can be computed using reuse ratio (RR) and specialized ratio (SR) with equation 5 and 6 as follows.

$$RR = \frac{\sum SC}{TC} \qquad (6)$$

$$SR = \frac{\sum IC}{TC} \qquad (7)$$

Where

$\sum SC = totalnumberofsuper \vee baseclasses,$
$\sum IC = totalnumberof\ \ descendantclasses$
$TC = totalnumberofclassesused \in system$

As the results of the research conducted by [20] and [21], high values of MHF and AHF parameters indicate that the system has a high level of encapsulation. Similarly, if the values of PF and NMO parameters are high, it shows that the system has a high level of polymorphism. A high value of the RR parameter expresses that the system has many reusable classes, while a high value of the SR parameter states that the system has many specialized classes. Therefore, the RR and SR are important parameters to determine the impact of a modeling and system design technique in terms of modification aspect.

RESULT AND DISCUSSION

A. System Requirements and Analysis

The first step to employ the use case 2.0 approach is to define the system requirement and analyse it. The following are the specifications of the functional requirements:
1. The system allows users to view rainfall data.
2. The system allows users to search rainfall data.
3. The system allows users to select notification periods through configuration (30 minutes, hour).
4. The system allows users to receive notifications for the latest rainfall data.
5. The system allows users to export rainfall data in csv format.
6. The system allows IoT devices to collect rainfall data.
7. The system allows IoT devices to store rainfall data into a database.

For the non-functional system specifications, they are defined as follows:
1. Data is stored using Firestore in the Firebase.
2. Minimum requirement for Android is Android 5.0 Lollipop.

Furthermore, each point in the functional specification needs to be transformed into user stories to determine the actors and the use cases. To simplify the determination of the actors and the use cases, this paper employs user stories with the pattern of role-feature-reason [22], as follows:

1. As a user, I want to view rainfall data so that I know the rainfall conditions.
2. As a user, I want to search for rainfall data so that I can view the data I need.
3. As a user, I want to choose a notification period so that I can receive the latest data notifications regularly.
4. As a user, I want to receive notifications so that I can view rainfall data at a certain time without opening the application.
5. As a user, I want to export rainfall data so that I can use the data in other applications.
6. As an IoT device, I want to collect rainfall data so that I can monitor the rainfall conditions.
7. As an IoT device, I want to store rainfall data so that the collected data can be saved into the database.

After the user story has been described, we are able to define use cases of the system and their actors, and then use them to visualize in a use case diagram for providing an overal big picture of the rainfall monitoring system. Figure 1 shows the proposed use case diagram of the rainfall monitoring sytem.

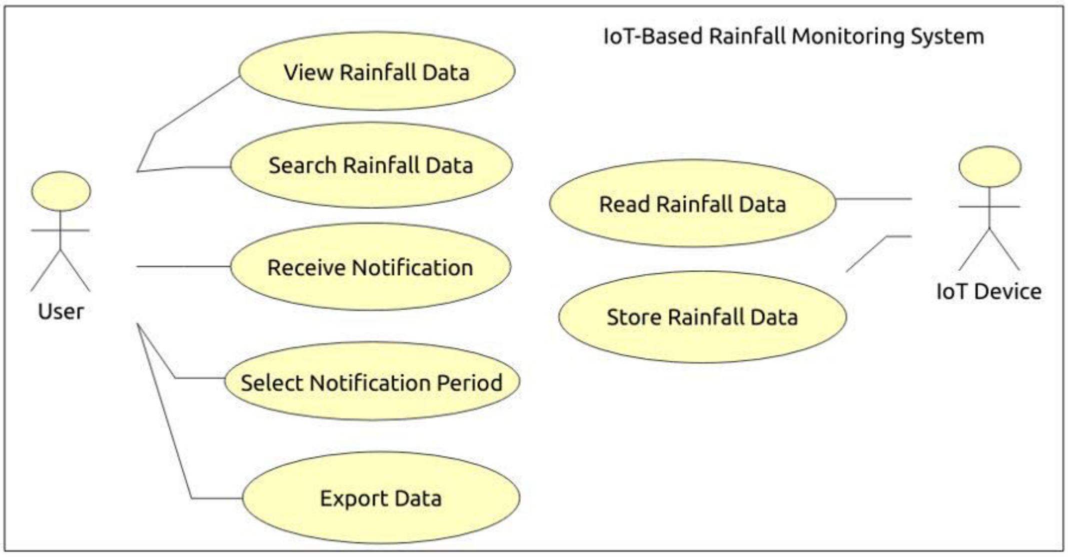


Figure 1. The Use-Case Diagram for Rainfall Monitoring System

Table 1. Slicing Use Cases

Use-case	Use-case Story	Use-case Slice
View Rainfall Data	Viewing the Latest Data (Basic Flow)	Succeed to view the latest data
	Data cannot be displayed (Exception)	Fail to view the latest data
Search Rainfall Data	Display all data (Basic Flow)	Display all data
	Display data based on input filter (Alternate)	Display data according to parameters
	There are no data (Exception)	Search empty result
Read Rainfall Data	Read rainfall data (Basic Flow)	Succeed to read rainfall data
	Fail to read rainfall data (Alternate Flow)	Fail to read rainfall data
Store Rainfall Data	Store rainfall data (Basic Flow)	Succeed to store rainfall data
	Fail to store rainfall data (Alternate Flow)	Fail to store rainfall data

The next activity is the core of the use-case 2.0 approach where a system must be built in slices. Each use case presented in Figure 1 should be broken down into use-case slices (thinner slices) by considering its basic flow, alternate, and exception flow. The analysis of the use-case slices for four use case is demonstrated in Table 1.

A detailed use-case development diagram based on the Table 1 can be visualized by illustrating each use-case slice as an extension of its original use case. As shown in Figure 2, the use-case slices for the "View Data" and "Read Rainfall Data" use cases are depicted.

ISSN 2089-8673 (Print) | ISSN 2548-4265 (Online)
Volume 12, Issue 2, July 2023

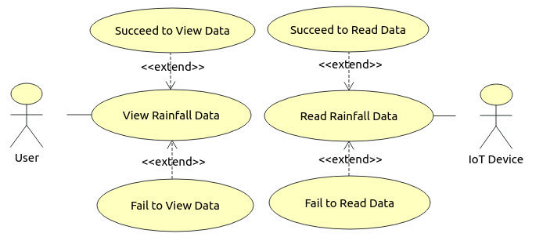


Figure 2. The Use-Case Slices for View and and Read Rainfall Data

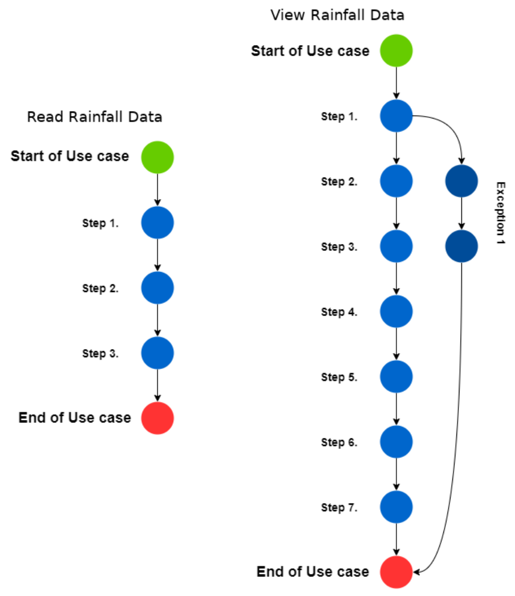


Figure 3. The Use-case Stories for Read and View Rainfall Data

After the use-case slices are identified and defined in the backlog or priority board development, the next step is to prepare these use-case slices for further analysis by creating detailed steps. For example, we prepare the use-case slice of Succeed and Fail to View Data, which are extensions of the View Rainfall Data use case where the user is defined actor.

The steps of the Succeed to View Data use-case slice are as follows:

1. User opens the home page.
2. Application connects to the internet.
3. Application searches for the latest data.
4. Data is found.
5. Data is received by the application.
6. Application processes the data.
7. Rainfall information is displayed to the user.

The steps in the "Failed to View Latest Data" use-case slice are as follows:

1. The user opens the home page.
2. The application is not connected to the internet.
3. The application displays an error message to the user.

As for the use case involving the IoT device actor, for example, the "Successfully Read Rainfall Data" use case, the analysis of its steps is as follows:

1. Initialize the IoT device.
2. Read data from the sensor for a specific period.
3. Calculate the average rainfall data.

The result of the analysis of the steps for the "Read Data" and "View Data" use-case slices is modeled visually in Figure 3. In the diagram, for the "View Rainfall Data" use case, there are seven steps, and after step 1, an exceptional condition can occur if the application fails to connect to the internet.

After preparing the use-case stories, each of the use-case slice should be analyzed one by one, then designed and implemented them into the system.

The next step, an activity/swimlane diagram can be created to model the activities performed by each object in the system, referring to the previously analyzed steps. In the example use case "Viewing Latest Data," there are three objects with certain roles in the system: User, IoT Device, and the Application. The User performs the activity of opening the application's home page, while the Application performs activities such as checking the internet connection, requesting the latest data from the IoT Device, and displaying the latest rainfall data to the User. The IoT Device, on the other hand, performs the activity of collecting rainfall data from the sensor and providing them to the Application. Figure 4 presents the "Viewing Latest Data" activity and the involved objects through a swimlane diagram.

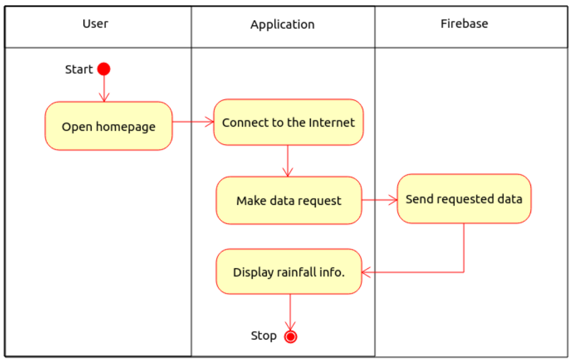


Figure 4. The Swimlane for View Data

To analyze and understand each of the use-case slice in terms of action sequences, a Sequence Diagram can be used. In the "Successful Viewing of Latest Data" use case, the action sequence starts with the User opening the home page, followed by the Application checking/connecting to the internet, making a request for the latest data, and processing them to display to the User. Figure 5 represents the sequence of data exchange for the "Successful Viewing Data" use case.

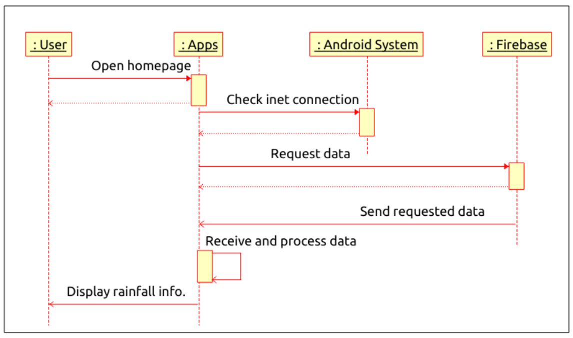


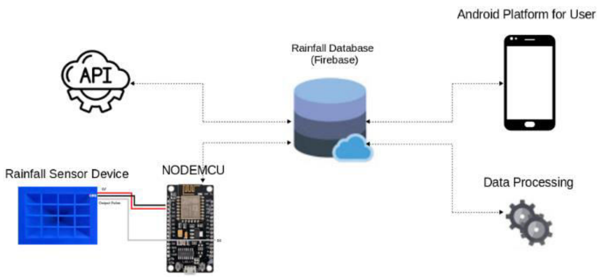
Figure 5. The Sequence diagram for View Rainfall Information

The final step in the analysis phase is to create a conceptual diagram to identify the classes that may be used in each use-case slice. For the "Successful Viewing Data" use case, it is known to consist of the Home Page class, which is a subclass of the Activity class. The class is also connected to other classes, namely the Rainfall class and the ConnectivityManager class. Activity and ConnectivityManager are two classes in the Android platform that are required based on the target platform in the definition of non-functional system requirements.

B. System Design and Implementation

In the design phase, we design system architecture, user interface (UI) for android platform apps, and the structure of the system by showing its classes, attributes, operates and relationship among classes.

The developed rainfall monitoring system consists of IoT hardware components, specifically the NODEMCU platform, and a rain sensor selected based on the considerations of the ease of use and cost-effectiveness. It also involves a mobile device based on the Android platform. The software components include the logic codes for processing rainfall data on the NODEMCU side using the C++/Arduino programming language, while the logic program on the Android platform using the Kotlin programming language. Firebase platform is used as the data storage for the designed IoT system. In the design phase, the selection of Firebase and the Android platform is based on the defined non-functional specifications of the analyses phase. Furthermore, to visualize the various components of the designed system and how they connected, we design architecture system which is presented in Figure 6. In the IoT device design, the output pin of the rain sensor generates a digital pulse signal using transistor-transistor logic (TTL). This signal can be read by the microcontroller as rainfall in inches or millimeters (mm). The sensor's output pin is connected to pin D8 on the NodeMCU. The VCC pin of the rain sensor is connected to the VU (Voltage USB) pin since the pin outputs a voltage of 5V, which is required for the sensor



to operate properly.

Figure 6. The System Architecture

ISSN 2089-8673 (Print) | ISSN 2548-4265 (Online)
Volume 12, Issue 2, July 2023

The UI design is tailored to the Android platform, which is the target implementation platform. Figure 7 presents an example of the UI design for the 'Successful Data Viewing' use case and the 'Search Specific Data by Time' use case. When rainfall data is successfully loaded, the application displays the retrieval date and time of the data, as well as the rainfall data for a specific time interval.

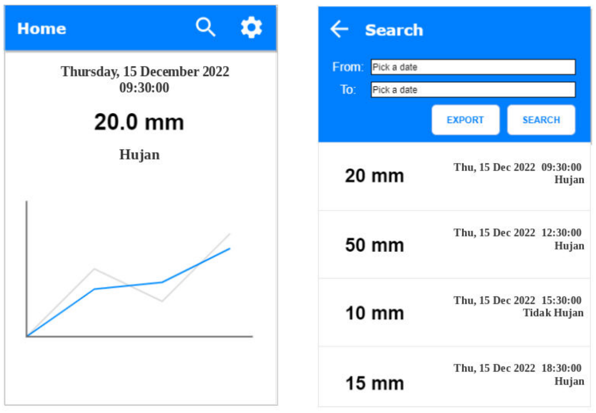

Home
Thursday, 15 December 2022
09:30:00
20.0 mm
Hujan
Search
From: Pick a date
To: Pick a date
EXPORT
SEARCH
20 mm
Thu, 15 Dec 2022 09:30:00
Hujan
50 mm
Thu, 15 Dec 2022 12:30:00
Hujan
10 mm
Thu, 15 Dec 2022 15:30:00
Tidak Hujan
15 mm
Thu, 15 Dec 2022 18:30:00
Hujan

Figure 7. UI Design for Display Rainfall Information.

In order to view the static structure of the application and to facilitate mapping activity from design to the implementation code, the detailed design is organized in to a class diagram. The conceptual class diagram in the analysis phase can be developed by defining in detail each class, function, attribute, and its type. Figure 8 illustrates the example of a section of the class diagram designed for the 'Successful Data Viewing' use case

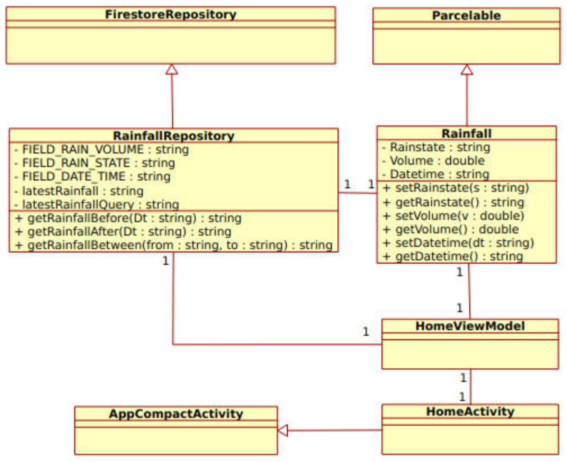
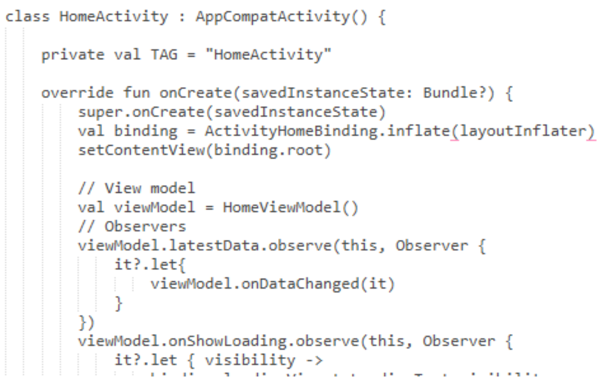

FirestoreRepository
Parcelable
RainfallRepository
- FIELD_RAIN_VOLUME : string
- FIELD_RAIN_STATE : string
- FIELD_DATE_TIME : string
- latestRainfall : string
- latestRainfallQuery : string
+ getRainfallBefore(Dt : string) : string
+ getRainfallAfter(Dt : string) : string
+ getRainfallBetween(from : string, to : string) : string
Rainfall
- Rainstate : string
- Volume : double
- Datetime : string
+ setRainstate(s : string)
+ getRainstate() : string
+ setVolume(v : double)
+ getVolume() : double
+ setDatetime(dt : string)
+ getDatetime() : string
HomeViewModel
AppCompactActivity
HomeActivity
1
1
1
1
1
1
1
1

Figure 8. The Class Diagram for View Latest Data

In the class diagram for 'Successful Data Viewing,' for example, the 'Rainfall' class has private (-) attributes with types String, Double, and string/TimeStamp, as well as the necessary methods for each attribute (set and get). Command methods, indicated by the prefix 'set' (e.g., setRainstate ()), are defined as procedures that accept String data, while query methods, indicated by the prefix 'get' (e.g., getRainstate()), are defined as functions that return String/Double/Timestamp data. Additionally, the diagram includes some parent class information, which represents Android libraries, based on the functions used by the subclasses.

In the implementation phase, when the use-case slices have been analyzed and ready to be implemented into the target platform. Then, each of use-case slice is applied one by one into the system until all the use-cases are implemented. For the Android platform, the implementation is done using Kotlin programming language. Figure 9 presents a snippet code mapped from the HomeActivity class design.

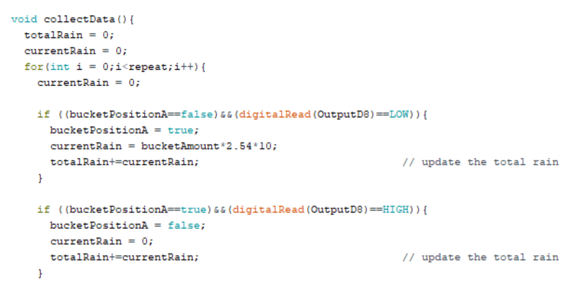


```
class HomeActivity : AppCompatActivity() {

    private val TAG = "HomeActivity"

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        val binding = ActivityHomeBinding.inflate(layoutInflater)
        setContentView(binding.root)

        // View model
        val viewModel = HomeViewModel()
        // Observers
        viewModel.latestData.observe(this, Observer {
            it?.let{
                viewModel.onDataChanged(it)
            }
        })
        viewModel.onShowLoading.observe(this, Observer {
            it?.let { visibility ->
```

Figure 9. Implementation Code in Kotlin for Viewing Data

IoT device in this designed system based on a low cost NODEMCU platform, then the software implementation for the platform is mapped into the C++/Arduino programming language. Figure 10 shows a snippet of code mapped from the Successful Data Reading class design for the NODEMCU/Arduino platform.

ISSN 2089-8673 (Print) | ISSN 2548-4265 (Online)
Volume 12, Issue 2, July 2023

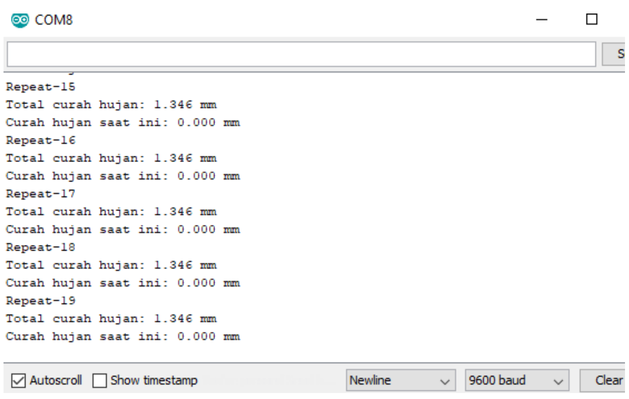


```
void collectData(){
  totalRain = 0;
  currentRain = 0;
  for(int i = 0;i<repeat;i++){
    currentRain = 0;

    if ((bucketPositionA==false)&&(digitalRead(OutputD8)==LOW)){
      bucketPositionA = true;
      currentRain = bucketAmount*2.54*10;
      totalRain+=currentRain;                              // update the total rain
    }

    if ((bucketPositionA==true)&&(digitalRead(OutputD8)==HIGH)){
      bucketPositionA = false;
      currentRain = 0;
      totalRain+=currentRain;                              // update the total rain
    }
```

Figure 10. Implementation Code in C++/Arduino Language for Reading Data

C. Evaluation

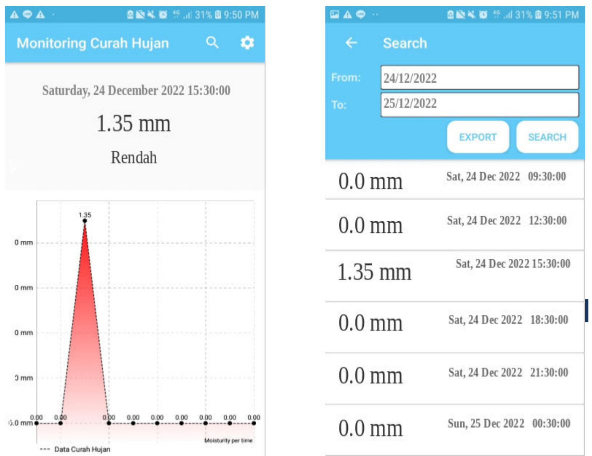
In this paper, we perform two evaluation scenarios. First was conducted to check if the use-case slices have been implemented correctly by executing the selected number of use-cases in a controlled environment. The first test involves the Successful Sensor Data Reading use-case, which involves the IoT device. Figure 11 presents the data successfully read by the IoT device through the rain sensor and displayed on the serial monitor. The successfully read data is then sent by the IoT device to the Firebase database on the internet for storage purposes.



```
Repeat-15
Total curah hujan: 1.346 mm
Curah hujan saat ini: 0.000 mm
Repeat-16
Total curah hujan: 1.346 mm
Curah hujan saat ini: 0.000 mm
Repeat-17
Total curah hujan: 1.346 mm
Curah hujan saat ini: 0.000 mm
Repeat-18
Total curah hujan: 1.346 mm
Curah hujan saat ini: 0.000 mm
Repeat-19
Total curah hujan: 1.346 mm
Curah hujan saat ini: 0.000 mm
```

Figure 11. Reading Data By IoT Device

Furthermore the testing for the Successful Data Viewing use-case. In this evaluation, a Samsung J7 Prime device with Android version 8.1.0 is used. The application reads data from the Firebase database and displays them on the mobile device. Figure 12 shows the execution result of the rainfall monitoring application on the target platform.



Saturday, 24 December 2022 15:30:00
1.35 mm
Rendah

From: 24/12/2022
To: 25/12/2022

0.0 mm Sat, 24 Dec 2022 09:30:00
0.0 mm Sat, 24 Dec 2022 12:30:00
1.35 mm Sat, 24 Dec 2022 15:30:00
0.0 mm Sat, 24 Dec 2022 18:30:00
0.0 mm Sat, 24 Dec 2022 21:30:00
0.0 mm Sun, 25 Dec 2022 00:30:00

Figure 12. Rainfall monitoring apps on Android platform

The second evaluation was conducted to compare the object-oriented metric properties between the system using the use-case 2.0 approach and the system without the approach (ad-hoc). The total number of classes, methods, visible and non-visible attributes, overridden methods, newly created methods, and the number of subclasses in the application codes are counted and measured. Table III presents the comparison results of the metric calculations with three parameters such as encapsulation, polymorphism, and reusability.

Table 2. OO Metrics Evaluation System with and without Use Case

System Label	Rainfall Monitoring System	
	With use case 2.0	Without use case (ad-hoc)
MHF	0.125	0.1515151515
AHF	0.5760869565	0.7777777778
PF	0.5208333333	1
NMO	24	19
RR	0.3513513514	0.1891891892
SR	0.5945945946	0.2702702703

As shown in Table 2, it can be observed that, specifically for the reusability parameter, the Reuse Ratio (RR) and Specialization Ratio (SR), which are important indicators of the Use-Case 2.0 approach, have significantly higher values compared to the ad-hoc method. The monitoring system for rainfall, modeled using the Use-Case 2.0 approach, has an RR value of 0.35 and an SR value of 0.59, while the ad-hoc method has an RR value of 0.18 and an SR value of 0.27. This indicates that the Use-Case 2.0 approach enables the system components to be more specific or high cohesive and independent or loose coupling, thus achieving

ISSN 2089-8673 (Print) | ISSN 2548-4265 (Online)
Volume 12, Issue 2, July 2023

the goal of breaking down the system into smaller parts (use-case slices) that are easier to develop and implement independently.

CONCLUSION

Based on the evaluation of the implemented code for the Use Cases of Reading, Viewing, and Searching for Rainfall Data, can be executed on the target platform. Furthermore, the measurement of the quality aspect using object-oriented metrics in the designed system, specifically the important parameters for the Use-Case 2.0 approach, has values of RR=0.35 and SR=0.59, while the development of the system without the approach has values of RR=0.18 and SR=0.27. This indicates that the implementation of the Use-Case 2.0 approach, which divides the system into small and independent parts namely use-case slices, can have an impact on the ease of modifying the IoT system.

REFERENCES

[1] J. Qiu, Z. Tian, C. Du, Q. Zuo, S. Su, and B. Fang, "A survey on access control in the age of internet of things," *IEEE Internet Things J.*, vol. 7, no. 6, pp. 4682–4696, 2020.

[2] M. A. Zamora-Izquierdo, J. Santa, J. A. Martínez, V. Martínez, and A. F. Skarmeta, "Smart farming IoT platform based on edge and cloud computing," *Biosyst. Eng.*, vol. 177, pp. 4–17, 2019.

[3] J. Muangprathub, N. Boonnam, S. Kajornkasirat, N. Lekbangpong, A. Wanichsombat, and P. Nillaor, "IoT and agriculture data analysis for smart farm," *Comput. Electron. Agric.*, vol. 156, no. June 2018, pp. 467–474, 2019.

[4] K. S. Uray Ristian, Ikhwan Ruslianto, "Sistem Monitoring Smart Greenhouse pada Lahan Terbatas Berbasis Internet of Things (IoT)," *JEPIN (Jurnal Edukasi dan Penelit. Inform.*, vol. 8, no. 1, pp. 87–94, 2022.

[5] P. K. Tripathy, A. K. Tripathy, A. Agarwal, and S. P. Mohanty, "MyGreen: An IoT-Enabled Smart Greenhouse for Sustainable Agriculture," *IEEE Consum. Electron. Mag.*, vol. 10, no. 4, pp. 57–62, 2021.

[6] T. Nguyen Gia *et al.*, "Energy efficient fog-assisted IoT system for monitoring diabetic patients with cardiovascular disease," *Futur. Gener. Comput. Syst.*, vol. 93, pp. 198–211, 2019.

[7] A. Rghioui, A. Naja, J. L. Mauri, and A. Oumnad, "An IoT Based diabetic patient Monitoring System Using Machine Learning and Node MCU," *J. Phys. Conf. Ser.*, vol. 1743, no. 1, 2021.

[8] K. U. Ariawan, "Penerapan IoT untuk Sistem Kendali Jarak Jauh Peralatan Listrik Rumah Tangga Berbasis RASPBERRY PI," *J. Nas. Pendidik. Tek. Inform.*, vol. 9, no. 3, p. 292, 2020.

[9] I. Sommerville, *Software Engineering, 9th Edition*. Addison-Wesley, 2011.

[10] G. Guerrero-Ulloa, C. Rodríguez-Domínguez, and M. J. Hornos, "Agile Methodologies Applied to the Development of Internet of Things (IoT)-Based Systems: A Review," *Sensors*, vol. 23, no. 2. 2023.

[11] D. Pandit, S. Chowdary, P. S. R. Patnaik, B. Shaharkar, and A. Surde, "Agile Methodology for IoT Application Development and Business Improvisation," in *Smart Trends in Computing and Communications*, 2022, pp. 601–608.

[12] B. Costa, P. F. Pires, and F. C. Delicato, "Modeling IoT Applications with SysML4IoT," *Proc. - 42nd Euromicro Conf. Softw. Eng. Adv. Appl. SEAA 2016*, pp. 157–164, 2016.

[13] B. Costa, P. F. Pires, and F. C. Delicato, "Modeling SOA-Based IoT Applications with SoaML4IoT," *IEEE 5th World Forum Internet Things, WF-IoT 2019 - Conf. Proc.*, pp. 496–501, 2019.

[14] M. T. B. Geller and A. A. de M. Meneses, "Modelling IoT Systems with UML: A Case Study for Monitoring and Predicting Power Consumption," *Am. J. Eng. Appl. Sci.*, vol. 14, no. 1, pp. 81–93, 2021.

[15] I. Jacobson, I. Spence, and B. Kerr, "The hub of software development," *Commun. Acm*, vol. 59, no. 61, pp. 94–123, 2016.

[16] H. Gomma, *Software Modeling & Design*. 2011.

[17] B. Mehboob, C. Y. Chong, S. P. Lee, and J. M. Y. Lim, "Reusability affecting factors and software metrics for reusability: A systematic literature review," *Softw. - Pract. Exp.*, vol. 51, no. 6, pp. 1416–1458, 2021.

[18] U. Kaur and G. Singh, "A Review on Software Maintenance Issues and How to Reduce Maintenance Efforts," *Int. J. Comput. Appl.*, vol. 118, no. 1, pp. 6–11, 2015.

[19] N. Padhy, S. Satapathy, and R. P. Singh, "State-of-the-art object-oriented metrics

and its reusability: A decade review," *Smart Innov. Syst. Technol.*, vol. 77, no. January, pp. 431–441, 2018.

[20] N. Nwe and E. Thu, "Measuring modifiability in model driven development using object oriented metrics," *Adv. Sci. Technol. Eng. Syst.*, vol. 3, no. 1, pp. 244–251, 2018.

[21] R. Harrison, S. J. Counsell, and R. V. Nithi, "An evaluation of the MOOD set of object-oriented software metrics," *IEEE Trans. Softw. Eng.*, vol. 24, no. 6, pp. 491–496, 1998.

[22] I. K. Raharjana, D. Siahaan, and C. Fatichah, "User Story Extraction from Online News for Software Requirements Elicitation: A Conceptual Model," *JCSSE 2019 - 16th Int. Jt. Conf. Comput. Sci. Softw. Eng. Knowl. Evol. Towar. Singul. Man-Machine Intell.*, pp. 342–347, 2019.