International Journal of Scientific Research in Computer Science, Engineering and Information Technology

ISSN : 2456-3307 OPEN ACCESS

Available Online at : www.ijsrcseit.com
doi : https://doi.org/10.32628/IJSRCSEIT

Study of Various Known Bugs and Other Challenges Associated with IoT System Development

Sandhya Devi[1], **Dr. Dev Singh**[2]

[1]Assistant Professor, Dept. of Computer Science and Engineering, Allenhouse Institute of Technology, Rooma, Kanpur

[2]Professor, Dept. of Applied and Computational Mathematics, Allenhouse Institute of Technology, Rooma, Kanpur

ARTICLEINFO

Article History:

Accepted: 05 Sep 2023
Published: 16 Sep 2023

Publication Issue
Volume 9, Issue 5
September-October-2023
Page Number
125-139

ABSTRACT

Embedded systems and smart homes are only two examples of the fast expanding use cases for Internet of Things (IoT) infrastructure. There has been no in-depth research of the difficulties developers face while working on the Internet of Things despite its rising popularity and widespread use. We provide the first comprehensive assessment of the issues and difficulties encountered by IoT developers via a large-scale empirical inquiry. A total of 5,565 bug reports from 91 typical IoT project repositories were gathered, and from those, a random sample of 323 were classified according to failure types, underlying causes, and physical locations of problematic components. To learn more about IoT problems and the difficulties faced by IoT developers, we also conducted nine interviews with industry insiders. In the end, we polled 194 IoT developers for confirmation and further information. Based on our findings, we offer the first bug classification for IoT systems. We focus on the most prevalent types of bugs that affect IoT systems, along with their origins, relationships, and the difficulties and obstacles that programmers often encounter while fixing them.

Keywords – IoT, Software Engineering, Data Mining, and Real-World Evidence

I. INTRODUCTION

The concept of the Internet of Things (IoT) calls for a network of "smart objects" equipped with sensors and actuators to be connected to the web using standard protocols for exchanging data [1]. By 2020, there will be four times as many smart, connected gadgets as people [2], and by 2025, that number is expected to rise to 75.44 billion [3]. These "things" are capable of collecting data or carrying out commands when programmed and operated remotely. The specific aspects of the difficulties in designing IoT systems include programming physical devices with limited resources, dealing with different network protocols, and merging different organizations. IoT bugs are more difficult to understand than software defects because of the factors listed above.

The features of IoT repositories [4] and various problems of IoT systems [5–7] have been the subject of previous research. The existing literature on bug classification focuses on certain areas of the Internet of Things, such as flaws in smart aquaculture systems [8], operating system flaws in IoT devices [9], and deployment flaws in IoT systems [5].

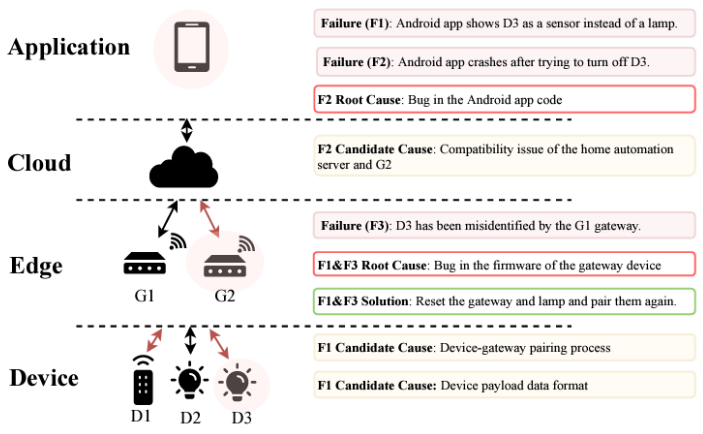
To the best of our knowledge, there is no similar research available for IoT, although more established software domains have benefitted from empirical and qualitative studies on their problems and developer issues [10]–[12].

This document surveys the problems that developers face while working on IoT systems, and it does so in a comprehensive and methodical way. We achieve this by collecting 5,565 bug reports from 91 typical IoT applications by crawling their GitHub repositories. We apply RCA to a sample of 323 bug reports, classifying them according to failure type, potential causes, and geographic origin. Based on our findings, we present the first comprehensive taxonomy of IoT system problems. We performed semi-structured interviews with nine IoT practitioners with real-world experience in various IoT layers to supplement the taxonomy and learn more about the difficulties faced by IoT developers. Finally, we confirmed our results with an online poll completed by 194 IoT developers.

II. CONTEXT AND INSPIRATIONS

The usual architecture of an IoT system is shown in Figure 1 [1, 7], [13], [14]. This layer is for the devices. Smart, programmable items whose embedded sensors and actuators interact with the physical environment are part of the device layer. Developers may create embedded code on top of the device OS with the help of lightweight embedded operating systems (like Contiki, RIOT, and TinyOS) [15]. Bare metal IoT devices, on the other hand, execute the embedded code directly in the hardware.

Device Layer: TinyOS with support for multiple programming languages, developers may build embedded code on top of the device OS [15]. These gateway devices have less resource limits and can collect, analyze, and route telemetry data locally at the network's edge. Bare metal IoT devices, on the other hand, execute the embedded code directly in the hardware.



Failure (F1): Android app shows D3 as a sensor instead of a lamp.

Failure (F2): Android app crashes after trying to turn off D3.

F2 Root Cause: Bug in the Android app code

F2 Candidate Cause: Compatibility issue of the home automation server and G2

Failure (F3): D3 has been misidentified by the G1 gateway.

F1&F3 Root Cause: Bug in the firmware of the gateway device

F1&F3 Solution: Reset the gateway and lamp and pair them again.

F1 Candidate Cause: Device-gateway pairing process

F1 Candidate Cause: Device payload data format

Figure 1: Architecture of an IoT system

Edge Layer: In this layer, gateway nodes with less resource constraints collect, analyze, and route telemetry data at the network's periphery.

Cloud Layer: Cloud servers in the Internet of Things collect and analyze all telemetry data and provide two-way communication between diverse IoT gadgets for remote management and monitoring. Users may specify the interoperability behaviors of the IoT system by writing automation logic between IoT devices using the rule engine found in IoT cloud servers [17].

Inspiring Illustration: Motivated by a real-world IoT problem, we investigate the difficulties faced by developers while working on IoT systems. The actions done by IoT developers (shown on the right side of Figure 1) to determine what caused the problem reported as PYTRADFRI/135 [18] are shown. This malfunction happened in a smart home setting, which necessitates the use of a gateway device to link various gadgets to a server handling home automation.

Developer conversations have shown that this flaw initially appears at the application layer. When the user installs a light bulb device (D3), the software thinks it's a sensor and attempts to turn it off (steps 1 and 2), crashing (step 2). The developers' first hypothesis (step 3) was that a mismatch between a gateway library and the home automation server was at the heart of the problem.

Step 4's study into the edge layer failure, however, uncovered the fact that D3 is incorrectly identified as a remote controller (F3) by the gateway. The possibility for irregularities in the subject device's payload data was also explored (step 5), since the gateway in this system depends heavily on a precise format of response data from devices to identify their kinds correctly.

Developers also tested pairing in a variety of scenarios (step 6) to account for factors like device battery life and distance from the gateway. After elimination of all other possible explanations, an external firmware flaw in the G2 gateway device was found as the cause (step 7), and the problem was resolved (step 8) by resetting both the device and the gateway and re-pairing them. But, amazingly, F2 survived. After keeping an eye on the.

III.METHODOLOGY

We want to learn more about the nature of software defects in IoT systems and the difficulties faced by developers. In order to achieve this goal, we answer the following study questions:

RQ1: What are the classes of bugs in IoT systems?

RQ2: What challenges do IoT developers face in practice?

To find the answers to these questions, we perform a two-stage empirical study. We examine 323 open-source IoT project bugs and code changes in the first phase. We use the results to provide the first comprehensive classification of IoT system flaws. We undertake a qualitative research in the second stage (RQ2) by (1) conducting semi-structured interviews with Internet of Things (IoT) developers to identify previously unidentified categories of bugs and issues, and (2) conducting a survey of IoT developers to confirm the results and glean further insights. Our whole set of quantitative and qualitative data [19] is now accessible.

A. Classification of Internet-of-Things Flaws

Receipt of reported bugs: Locating repositories that are typical of Internet of Things initiatives is the first order of business. We used the “GitHub topic feature” to locate repositories pertinent to the Internet of Things. The official GitHub documentation [20] explains that Topics are tags used to establish relationships between repositories based on shared interests. We conducted a search for subjects that had “internet-of-things” and “IoT” as relevant keywords, and then added “IoT-application”, “IoT-platform”, and “IoT-device” from the results to our list of desired areas of study. In January of 2020, we gathered 8,774 repositories based on these five themes. Repositories with fewer than 10 ratings were disregarded [21], leaving 1,356 for analysis. In order to filter for legitimate issues, we only included complaints that were marked as “closed” and had the words “bug,” “defect,” or “error” in the subject line. We also hand-analyzed repositories with more than five labelled problems or more than fifty closed issues based on data from the readme page, issued bug reports, and the website (where available) to ensure that we included only typical IoT repositories in our analysis. We didn't include UI, documentation, or old repositories since these aren't typical of Internet of Things solutions. To complete this study, we compiled a list of 91 open-source IoT repositories. We modified our search terms to include the labels used by five repositories (such as “problems”, “kind/bug”, and “type: bug”) that use custom labels. There were a total of 91 IoT repositories, from which we gathered 5,565 problem complaints.

All of the components of the IoT system's architecture shown in Figure 1 are represented in our topic IoT repository. Python (21%), Java (18%), JavaScript (17%), C (13%), and C++ (13%), are the most common programming languages found in the topic repositories.

Besides Java, C++, and Python, some developers prefer Go, Ruby, and C#. There is a wide range of star and fork counts across the chosen repositories. About 32% of the repositories we looked at had over 500 ratings in February 2020, while 40% had between 50 and 500, and 28% had between 10 and 50.

Labeling: Each report of a problem in our dataset was converted into a JSON object that details the failure, the reason for the failure, and the exact line or lines of code that were flawed. When a system exhibits behavior that goes against what should be expected of it, we call it a failure [22], [23]. We performed root-cause analysis (RCA) on each report of a problem, digging into the "why" behind the issue using the "five whys" approach [24]. This method recognizes that a system failure may have originated from a number of diverse sources. Using this method, we began with the error and probed for the underlying reason by asking "why" again and over. When anything goes wrong with your software, it's probably because of something your developers did wrong when creating your Internet of Things system. We utilized the structure described in Section II as a standard by which to identify and designate potential problem areas.

Participants: To find programmers with sufficient expertise making IoT systems, we employed purposive sampling [27]. We choose GitHub because it has such a large community of engineers working on a wide variety of projects, making it ideal for recruiting interviewees. The top three contributors to the most popular open-source IoT repositories' email addresses were the only ones we were able to get for our candidate interviewees.

We used email to contact potential participants and in-person interviews to gather information until we had enough information to repeat the research [26]. We based our decision to discontinue conducting interviews based on this well-established methodological guideline [28, 29], which is also employed in other qualitative research in software engineering [30, 31]. To account for differences in experimental outcomes among groups, we conducted interviews with persons from a wide range of development experiences and backgrounds [32].

Protocol: We used a semi-structured approach to interviews because we wanted to be receptive to fresh information and we lacked a definite framework for classifying bugs. Participants were first asked about their experience with and knowledge of IoT development. We may use this knowledge to ask more in-depth questions on the topic at hand during the technical portion of the interviews. The technical component of the test included both free-form and narrowly focused inquiries about several classes of IoT development flaws and obstacles. To prevent participants from being predisposed to our results, we began with open-ended questions and transitioned to more closed ones later on.

Analysis: To assure the quality of the theory developed, we used the grounded theory technique [33], since the fundamental goal of this research is to generate theories from the experiences of IoT practitioners rather than utilizing pre-conceived theories. By repeatedly comparing all the previously analyzed data with the emerging theories, we (i) collect qualitative data from the interviews, (ii) analyze the interview transcript line by line and assign labels (tags) to distinct units of meaning, and (iii) identify emerging categories and relate categories to their sub-categories. Each interview followed the same format. We were able to glean an average of 18 tags per interview. After each round of editing, the writers addressed any labeling issues that had arisen.

B. Survey Validation

To make sure our results can be extrapolated to the wider IoT developer community, we conducted a poll online.

Participants: We sent our web-based survey to programmers who have made at least three contributions to the compiled IoT repositories in section III-A, as well as to IoT development groups on social media sites like Linkedin and Facebook, and to online forums. Between July 19 and August 19, 2020, we conducted an online survey. Our artifact bundle [19] includes the survey in its entirety as it was sent out to participants.

IV. BUG CATEGORIES IN THE IOT

Here, we detail what we discovered about security flaws in the internet of things.

A. Bugs' Taxonomy

To create taxonomy of IoT system issues, we utilized all the tags acquired by RCA from the bug reports in our dataset. Our illustrative figure II shows how IoT flaws might have several manifestations at various levels and places. As a result, we made sure that our bug taxonomy could account for all of these features. Taking into account the numerous methods proposed by Usma et al. Since IoT bugs are complex and mostly uncharted, we followed Kwasnik et al.'s [34] method for building a taxonomy of them. Taking cues from their method, we started by defining the failures and places that would become central to our categorization. We then categorized all reported bugs into a hierarchical taxonomy according to these factors. Interview and survey responses were utilized to supplement and improve the taxonomy after the original version had been developed. Following each revision to the taxonomy, we went through all of the data and re-applied tags. Our taxonomy of IoT bugs is shown in Figure 3. Our taxonomy of bugs continues with a discussion of its primary divisions. We'll illustrate each kind of problem using real-world instances. All these illustrative bugs

Protocol: There are three parts to the survey. In the first portion, we gather general and IoT development experience from participants, as shown in Figure 2. In the second half, we ask questions on the difficulties of creating IoT systems in an effort to draw parallels between our research and the participants' own experiences.

Analysis: We received 194 legitimate replies to our poll, for a response rate of around 10%. There were 95 responses to the free-form questions. Following the same method outlined in section III-B, all survey responses are coded and evaluated.

IoT Gadget: Hardware and software flaws in IoT devices are the subject of this taxonomy category. In this subcategory, bugs manifest themselves in the hardware of an Internet of Things device. Wiring problems, incorrect pin status, and malfunctioning sensors and actuators are all examples of faults that may arise from improper hardware. For instance, the PEDALINOMINI/34 issue is associated with the gadget's inability to distinguish between single and double button pushes. Similar flaws sometimes stem from the device's limited storage space, battery life, or computing power.

Device firmware: Firmware bugs consist of three sub-categories. The first pertains to device firmware unexpected exception and hang issues. The second sub-category includes issues related to the configuration of the IoT device, which can be specified as an external instruction sent to the device for a specific purpose. This bug usually happens in the early stages of introducing an IoT device to the IoT network. Each device has to be configured properly in a way to be compatible with other hardware or software components and also be able to communicate with others on the network. Issues associated with configuring the device with WiFi credentials or with configuring the device with the correct firmware version are some common examples here. The third and most common sub-category is the firmware upgrade issue. There are various cases where poor practices for handling over-theair (OTA) updates of the device firmware, stale updates, or updating the device firmware with the wrong binary have caused failures of the IoT system. The WTHERMOSTATBECA/54 is an example of a device that requires WiFi credentials to be changed after each firmware update or else future firmware upgrades will be obsolete.

Compatibility: Compatibility issues are those that manifest themselves on just one device, communication protocol, or external component. Some devices may have compatibility issues because they display their telemetry data in a way that is not understood by other devices. Other frequent difficulties stem from incompatibilities between sensors and development boards, such as the DHT temperature sensor's incompatibility with the ESP32 microcontroller in MONGOOSE-OS/277. Problems with the compatibility of several protocols is another example. As an example, consider the MAINFLUX/1079 issue about the compatibility of the HTTP and MQTT protocols. Building protocol- or device-specific code is a typical error in IoT development that contributes to these problems. In DEVICE-OS/1938, for instance, the IoT platform depends on event components to indicate what protocols each event is meant for, allowing for separate functions to be executed for each protocol. However, in order to get around the restrictions of third-party devices, developers are occasionally forced to resort to this error-prone method. For instance, [2] highlighted an example where developers were obliged to design bespoke logic to communicate between the Raspberry Pi and certain kinds of sensors due to the mismatch between the two platforms. Depending on the kind of sensor, developers have to choose between the Raspberry Pi's default implementation and a bespoke one.

Connection to the Internet of Things: Communication flaws between IoT devices or between IoT devices and external entities. There are two main families of bugs to consider here:

Problems with a device's Internet connection may stem from difficulties with the network via which the device is trying to connect. If the device is unable to locate a working network, such as a Wi-Fi hotspot, Internet connectivity will be lost. In addition to the network discovery, not handling a network reset or unstable and unreliable networks are also common issues that can lead to failures, as mentioned by P9: "When the device location is changed to another room or another building, the device has to be reconfigured for the new access point." However, even a good network, Internet of Things devices don't always succeed in establishing a reliable connection to the gateway or distant cloud servers. IoT developers also face challenges related to reconnecting devices, updating connections, and preventing failures in one component from spreading to others as a result of a connectivity issue.

Bugs in this category also reveal themselves in the form of sudden disconnection or connection closure. Connectivity issues are the most significant and difficult, according to two interviewers [6, 9]. Our ability to interface with IoT devices is our platform's weak point, as P9 puts it.

Issues with the transmission of data and messages inside the IoT system fall under this category. Communications between devices in the Internet of Things (IoT) often take the form of instructions delivered to devices in the cloud or telemetry data received from devices at the edge, cloud, or apps. Occasionally, flaws prevent these communications from reaching their intended recipients. The timing of messages is a source of trouble for certain users. Some of the identified issues include the delivery time and sequence of messages.

Moreover, certain flaws are associated with the content of the communications themselves. Failures may sometimes be traced back to issues with the payload's size or structure. Message truncation and overwriting are other common causes of payload integrity breaches.

Hybrid cloud and edge computing services: Edge-layer services provided by distant cloud servers or gateway devices are susceptible to this group of problems.

Management of Devices: All Internet of Things gadgets should communicate their status to a central server or hub, where they may also receive orders from users. Failures in device management (DM) may be attributed to a variety of factors. The first category of DM problems occurs during initialization of the IoT device in either the cloud or the edge systems. When the cloud or edge components do not correctly identify the IoT device, this is an example of a device initialization (DI) problem that may lead to further problems in the targeted IoT system. In addition, the IoT device wouldn't be able to access distant services if it couldn't prove its identity to the cloud or edge. Bugs with device registration and provisioning may manifest itself in a number of different ways, such as with duplicate device certificates, problems with automatically provided devices, or an inability to get data from the provisioning service. Binding, association, and pairing issues with IoT devices are yet another category of DI defects. When the relationship between a sensor device and a physical item is not handled correctly, flaws may be introduced into the IoT system simply by grouping devices (such as devices in one room). One of our interviewees [5] gave an example in which two switches controlled the same bulb, but only one of the switches really worked owing to problems with labeling devices with multiple instances. The second group of DM concerns involves checking up on the health of IoT gadgets. The connection status of a device may be checked to see whether it is online; this is also called a heartbeat check. This category of defects includes issues like incorrect device heartbeat rate, incorrectly displaying a lost connection as active, or failing to inform other components when the device is offline. The state of the device, such as the color and brightness of a light bulb, may not be retrieved, the status may be shown erroneously, or the device's status may not be updated.

Progress in General Terms: Many errors encountered while programming are cataloged here. Unexpected crashes or performance issues in the IoT project are also typical, as are difficulties with installation, compilation, and construction. Authentication and authorisation flaws are also considered part of generic development flaws. The generation, signing, or upkeep of certificates that devices must submit for utilizing cloud or edge services (AZURE-IOT-SDK-C/657) is one IoT-specific authorisation challenge. Interface-related, usability, and external problems are other types of development flaws.

B. Typical Insect Traits

Reasons why: General programming errors account for 48% of all faults, followed by device management problems at 29% and communication problems at 19%. In terms of root causes, semantic programming errors (SEM) are the most common sources of defects, followed by generic software programming faults (SWP) such syntax difficulties. Mistakes in control flow, functionality logic, or return values are examples of semantic errors made by IoT developers. Recent studies [37] also examine logical flaws in automation applications and other semantic errors that are connected to the automation logic of the IoT system. Dependency errors (DEP) are another common source of problems, and they occur when programmers utilize outdated versions of necessary libraries, tools, devices, or protocols in their code. Timing errors (TM) are a common source of problems in hardware, network, and message delivery. Time-related reasons include, but are not limited to, incorrect time-out values for connection closures, improper handling of asynchronous behaviors, and improper management of time-outs and rates of operations. Hardware programming failures (HWP) are those that occur more often in hardware-specific code, such as interrupt handling. Another common source of IoT flaws is improper handling of exceptional circumstances (EC). Mistakes in handling edge situations (big or out of range data), failures, and modifications to specifications or third-party components are all examples. Finally, concurrency faults (CON), configuration faults (CNF), and memory faults (MEM) account for the remaining reasons.

Relationships between various types of bugs: Based on our research, we found that some types of bugs tend to cluster together. We utilized Lift [38], a statistical measure developed by Han and Kamber that determines the likelihood of two categories occurring together, to analyze the relationships between bug types. A lift number more than 1 indicates a positive connection between the two bug categories, whereas a lift value less than 1 indicates a negative association.

How often and how badly bugs occur: We polled users and asked them how often they encounter each problem (sub)category, how severely it affects the IoT system, and how long it takes to remedy. The outcomes are shown in Table III. All of the categories in our taxonomy reflect genuine defects in IoT systems, since at least 82% of IoT developers have encountered them. More than 97% of IoT developers have encountered connectivity difficulties, making it the most common and serious fault category.

After connection concerns, device-related bugs are the second most serious kind of defect there is. Automation problems are the least serious flaws, yet more than 91% of IoT engineers have encountered them at some point, according to the report. According to the experiences of IoT developers, compatibility problems are the least common kind of defect. About 95% of IoT developers have encountered device initiation difficulties, making them the most common and serious device management defects.

Bugs affecting the device's connection state have also become increasingly common. However, issues with the current settings of a device were seen as more serious by survey participants. Bugs that arise from the limitations of IoT devices are the most common kind of device-related problem. The worst device-related defects are those that occur due to exceptions in the firmware. As a middleware developer put it in [5], “IoT device vendors do not provide a mock of their devices, and we have to do reverse engineering on the actual hardware devices rather than working with the simulated ones.” Additionally, [5, 8, 9] all agreed that the current simulation solutions in IoT are not mature enough and are only valid for limited scenarios, such as testing high-level controllers or small unit tests.

The report highlights the difficulties of having a wide variety of IoT devices, developing complicated custom logic for realistic IoT device mocking and simulation, and establishing test environments with IoT devices.

Identifying the precise location of the problem: Fault localisation is hampered by a lack of visibility into the inner workings of IoT systems, as reported by eight interviews, nine survey comments, and half of the survey's respondents. A major challenge in tracking executions of various external components in IoT systems is that, as described in [7], “there is no environment that logs everything.” Open-source software was highlighted by [3] as a means to provide comprehensive logging.

Improved taxonomy: Our data set for Internet of Things flaws includes 79 interview tags and 18 survey comments. Experienced interviewers uncovered new problems, such as those with device binding, performance, and third-party compatibility, after the fact. No more taxonomic detail was revealed in the survey responses. However, the contextual data provided by the retrieved tags allowed us to better describe each problem category (from both the interviews and the survey).

V. RESULTS OF RQ2

Here, we provide our results on the difficulties encountered by IoT developers. Difficulties in Testing and Bug-Fixing depending on use of the actual gadget. IoT developers depend on access to devices to test and debug their IoT system, through operations like manual reset or monitoring device output [2, 3, 7], as reported by 7 interviewers and many GitHub issues.

When devices are located in inconvenient or inaccessible areas, remote debugging becomes even more important. Four of the individuals surveyed agreed that realistic simulation solutions are necessary for improved IoT testing and debugging. Fault localisation may also be affected by the presence of concealed failures.

This is seen in GitHub issues (DEVICEOS/1926, ZWAVE2MQTT/141, VSCP/207), and is illustrated by

[7], which states, "It's hard to recognize on the app that the temperature the device is reporting now is for several minutes ago." [2, 4] also provided examples of failures that manifest themselves only after the device has worked for a certain amount of time (five minutes for [2], several hours for [4]). This problem increases the unpredictability of IoT failures and may mask mistakes made by developers. The absence of tools and developer assistance is another roadblock on the path to fault localization. For instance, [3] uses Wireshark to monitor communications and do bit-level inspections of device messages. Since there is no feedback of faults or corruptions from devices, [2], a developer of a hardware platform, stated, "Since there is no feedback of errors or corruptions from devices, we've added some LEDs to them to track if something is working in the device level or not."

Flaws in the Internet of Things that keep reproducing: We gathered four tags from interviews and three survey responses about the difficulty of replicating IoT problems by following various GitHub conversations (DITTO/414, TESLA-API/68). Some flaws only occur with a particular device configuration or with certain conditions of the IoT system, in addition to the above described variables which harden bug replication, such as restricted access to devices or concealed failures. Without the same context, IoT developers will not be able to replicate these issues. One survey response, for instance, said

Verifying and fixing special-case issues: Problems arise when trying to test for extreme conditions, such as when there are too many devices or when it's really cold outside. Several GitHub threads (DEVICE-OS/1926, TEMPERATURE-MACHINE/13) and comments from interviews and survey takers all point to this difficulty.

For instance, [4] mentioned that "We should put effort to write proper tests against concurrency issues since we should be able to handle 140, 000 HTTP requests per second because our IoT system is deployed in different cities." This issue has been encountered by the largest percentage of respondents (83%), making it the most experienced testing challenge.

Lack of experience in testing. Sixty-four percent of respondents said developers are the primary testers in their IoT project, as seen in Figure 5. 'We do not have a QA crew,' P6, creator of a nearly 7,000-star IoT project, said. Often, software developers lack the expertise to test the hardware side, thus it's up to the developers to undertake the testing, either manually or by designing automated tests. The lack of understanding about methods and techniques of hardware testing was cited by P9, the creator of an IoT platform with 1.5K stars, as the main bottleneck of their IoT platform.

Heterogeneity: Some IoT developers have noted that their platform is limited to specific protocols rather than devices in order to achieve interoperability [2, 5, 8]. For example, [3] stated he has to develop a distinct adapter for talking with each particular device. He added, "There is no guarantee that something that works with brand A also works with brand B."

All interviewees mention the difficulty of third-party breaking changes (23 tags), and 63% of survey participants agree with this assessment; furthermore, several comments in the survey (eight tags) discuss this issue. Three interviewees stated that third-parties make breaking changes without prior notice.

Challenges posed by the fundamental diversity of IoT technologies are the most repeated challenges in both interviews (30 tags) and survey comments (25 tags), and this is agreed upon by 60% of survey respondents. Multiple interviewees and survey commenters noted that IoT development necessitates a wide range of development skills, including hardware programming and familiarity with dealing with network protocols.

Programmers seldom go through this training: "developers tend to use protocols which they are familar with, but sometimes better solutions exist and developers do not know/use them." [2, 3, and 2] survey comments mentioned that user requirements and users' backgrounds and skills can be very disparate, making it challenging to develop a generalized IoT system that can support all possible use cases. For example, [2] mentioned they had to i) understand the low-quality documentation of some device manufacturers and ii) interpret complex response payloads from some devices.

A. Various Difficulties

Six of the 14 participants who mentioned security-related challenges cited it as the most important challenge. Furthermore, 66% of IoT developers find security a complicated task.

Similarly, nearly 60% of IoT developers believe that device constraints make security tasks difficult. Another theme from our data concerns the difficulty of end-to-end security, from the IoT device to the cloud. Some [8, 9] believe that the security of the local communication between the device and IoT gateway is usually underestimated while it can be highly insecure. Six IoT developers mentioned "getting critical updates installed on already sold devices" or "firmware updates in large deployments" as difficulties associated with releasing updates for IoT devices, with half of interviewees agreeing that this is an inevitable challenge [5, 8].

Device limitations in different layers have also been mentioned by our interviewees [2, 3, 6, 8], and 63% of participants agreed that device constraints make IoT development more difficult. Most IoT developers struggle to design and implement software to consume less processing power and energy.

Lack of device-level monitoring tool support: Investigating the log data of IoT devices is a common debugging task for IoT developers. This task becomes even more important as the device status issues are among the most frequent bug categories. This bug category has appeared in around half of the bug reports in our dataset, and most IoT developers reported that they need to log communications or internal executions of the device as part of the debugging process for these bugs [1, 2, 3, 4, 7]. There is no universal tool that receives log data from all types of devices, and developers often have to manually employ naive approaches to monitor device status and communications, such as serial print for each device separately [2, 7] or using general-purpose tools like Wireshark [3, 7]. As several IoT developers discussed their limitations, existing logging solutions to track devices are considered inefficient. One IoT developer best mentioned it: "even if some devices provide log libraries and tools, they should be manually aggregated or traced from each component separately to track an issue."

The Internet of Things environment is chaotic and always evolving: The quick decline in usefulness of hardware is now one of the biggest obstacles to the expansion of the Internet of Things. Several IoT specialists and blog postings [47], [48] highlight a rapid increase in the rate at which IoT devices become unsupported and hidden from public view. IoT device upgrades often render previously-released devices obsolete and cause havoc for existing IoT developer implementations. Developers have challenges when trying to keep their device- or protocol-specific code up-to-date inside the dynamic IoT ecosystem. Not only must IoT developers be able to purchase all versions of devices to keep up with these changes, but they must also devote a significant portion of their development work to moving from one version or ecosystem to the other. Some nations in 2019 have imposed laws on the minimum period IoT providers may send updates after the device is acquired [49], since this problem affects both IoT users and developers. Some alternatives, such as contract-based testing, were also proposed by respondents in interviews [5] to guarantee ongoing interoperability with external systems. Since they are all dependent on preexisting contracts and rules, none of these approaches can provide a permanent and all-encompassing fix.

B. Validity Threats

Internal consistency: Researchers' bias in categorizing qualitative data is an intrinsic danger to the validity of our study, as it is to the validity of most qualitative investigations. We reduced this possibility by having all article authors participate in the tagging process and by addressing any labeling differences across all bug reports, interview transcripts, survey comments, and other types of qualitative data. Through the use of triangulation, we were able to rule out the influence of interviewer bias on our findings by having two researchers independently tag all relevant information in interview transcripts.

Context-free proof: The potential for the generalizability of the examined IoT repositories poses an external danger to the reliability of our research. To lessen the impact of this problem, we analyzed 91 repositories representative of all IoT system levels. Our study's validity is further threatened by the fact that the interview and survey participants may not be typical of IoT developers. However, we were able to reduce this possibility by selecting interview and survey respondents from a wide range of backgrounds, experiences, industries, and firms relevant to the Internet of Things. In addition, 194 IoT developers with a wide range of expertise and experience have participated in our poll. You may get the bug dataset, as well as interview and survey questions we used in our research, on our website [19].

VI. SUBJECT MATTER

Internet of Things flaws and difficulties: Even while certain forms of IoT system problems have been recognized in the past [5, 8, 9], no prior research has attempted to systematically classify all sorts of true IoT system faults. Recent research from 2020 [4] looked at the ways in which developers contribute to IoT repositories to draw conclusions about the unique characteristics of these open-source resources. However, the study's findings on the features of IoT development do not take into account issues and the experiences of IoT developers.

There is a growing amount of research on the problems and defects in design that lead to security breaches in IoT systems [50], [51]. Security flaws in smart home ecosystem devices' firmware [52]–[54], communication protocols [55]–[57], smart applications, and the safety of their interactions [37], [58], [59], and interactions between various IoT system components [17] have all been investigated. There are taxonomies for characterizing the features of IoT systems in terms of privacy and security [60, 61]. These articles have a different emphasis, on security needs and threats, and they don't provide their taxonomy development technique.

The difficulties of evaluating Internet of Things systems have been the subject of many research [39, 46, 62, 63]. Model-based testing [62], IoT mutation operators and test event generators [64, 65], and testing tools [39] are only a few of the methods presented for IoT testing. There have also been suggestions made for tools and procedures to help IoT developers create IoT systems [66–68].

Different approaches to the difficulties of creating IoT systems have been presented [5, 7, 13]. Previous research looked at the difficulties of newbie IoT developers to determine which aspects of development posed the most difficulty [6] and created a tool to aid novice developers. However, there has been no attempt to systematically examine the difficulties encountered by IoT developers via in-depth interviews and surveys of IoT professionals.

Problems with bug mining and programming: Many research have used mining of software repositories or issue trackers to classify defects in Machine Learning systems [10], [69]-[71], but no such work has focused on mining IoT repositories. Prior studies in Blockchain systems [11], Big Data computing platforms [72], web applications [73], and service compositions [74] have all used this method to classify problem types. Furthermore, the difficulties encountered by developers have been studied in many settings, including mobile app development [12] and Blockchain development [75].

VII. CONCLUSIONS

In this study, we presented the first comprehensive taxonomy of Internet of Things (IoT) system bugs. Using a qualitative analysis, we also identified a series of categories of problems encountered by these systems. Our results may inform the development of new methods and tools for IoT development by illuminating the challenges that developers in the wild face. Our research reveals the most common and severe IoT issues, their correlations, and their core causes, which may help developers avoid or quickly spot these problems as they work on IoT systems.

VIII. REFERENCES

[1] Demeter, Minerva, and D. As stated by Rotondi in the IEEE Internet Initiative volume, “Towards a definition of the internet of things (IoT),” 1, no. 1, pp. 1-86, 2015.

[2] M. Hung, “Leading the IoT, Gartner Insights on How to Lead in a Connected World,” Gartner Research, 2017, pp. 1–29.

[3] F. Data from Schwandt, “Internet of Things (IoT) Connected Devices Installed Base Worldwide from 2015 to 2025 (in billions),” Statista, 2016.

[4] F. Corno, Louise De Russis, and John R. P. Sáenz, “How is open source software development unique in mainstream IoT endeavors?' IEEE Access, vol. 8, pp. 28 337-28 348, 2020.

[5] T. It was found by W. Hnat, V. Srinivasan, J. Lu, T. I. Sookoor, R. Dawson, J. Stankovic, and K. Whitehouse, “The hitchhiker's guide to successful residential sensing deployments,” Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems, 2011, pp. 232-245.

[6] F. Corno, Louise De Russis, and John R. “On the challenges novice programmers experience in developing IoT systems: A survey,” P. Sáenz, Journal of Systems and Software, vol. 157, p. 110389, 2019.

[7] LR Stojkoska & K. V. Trivodaliev, “A review of internet of things for smart home: Challenges and solutions,” Journal of Cleaner Production vol. 140, pp. 1454-1464, 2017.

[8] Y. Chen, Zhen-Zhen, Hui-Yu, and Jian-Ping. Xu, “Application of fault tree analysis and fuzzy neural networks to fault diagnosis in the internet of things (IoT) for aquaculture,” Sensors, volume, page. 17, no. 1, p. 153, 2017.

[9] H. Together, Liang, Zhao, Wang, and H. Liu, “Understanding and detecting performance and security bugs in IoT oses,” 17th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD), 2016. IEEE, 2016. pp. 413–418.

[10] G. The authors (Jahangirova, N. Humbatova, G. Bavota, V. Riccio, A. Stocco, etc)

[11] P. This article cites the 2019 arXiv publication by Tonella, “Taxonomy of real faults in deep learning systems,” as its primary citation.

[12] Z. Xia, and L. Cai, “Bug characteristics in blockchain systems: a large-scale empirical study,” in IEEE/ACM MSR 2017: The 14th International Conference on Mining Software Repositories. In 2017 edition of IEEE, pages 413–424.

[13] M. By E. Joorabchi, A. Mesbah, and P. According to Kruchten, “Real challenges in mobile app development,” was published in the 2013 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement. 15-24 in IEEE's 2013 publication.

[14] According to “The three software stacks required for IoT architectures,” written by I. W. Group et al. in 2016, all IoT architectures must include all three.

[15] Javed, Muhammad K. Afzal, Muhammad Sharif, and B.-S. Kim, “A Comparative Review of the Support for Internet of Things (IoT) Operating Systems, Networking Technologies, Applications, and Challenges,” IEEE Communications Surveys & Tutorials, vol. 20, no. 3, pp. 2062-2100, 2018.

[16] H. The authors Tschofenig, Arkko, and D. According to McPherson, “Architectural Considerations in Smart Object Networking,” Internet Engineering Task Force, Fremont, CA, USA, 2014.

[17] W. A group of researchers led by Zhou found that Y. See: Zhang, “Discovering and Understanding the

Security Hazards in the Interactions between IoT Devices, Mobile Apps, and Clouds on Smart Home Platforms," in Proceedings of the 28th USENIX Security Symposium (USENIX Security), 2019, pp. 1133-1150.

[18] 2018; https://github.com/ggravlingen/pytradfri/issues/135 "Lamps not identified as lamps with f/w 1.3.14," Github.

[19] Both Makhshari and A. August 2020, Mesbah, https://github.com/IoTSEstudy/ IoTbugschallenges, IoT Bugs and Development Challenges Artifact Package.

[20] Topics are a great way to organize your repository, https://help.github.com/en/github/administering-a-repository/.

[21] classifying-your-repository-with-topics.

[22] H. In a similar vein to Borges, M. What's the value of a star on GitHub?," T. Valente. a study of repository-starring habits on GitHub, "Journal of Systems and Software," vol. 146, pp. 112-129, 2018.

[23] L. X. Wang, Y. Zhou, and C. Liu; Z. Li; C. Tan; C. Liu; X. "Bug characteristics in open source software," by Zhai. Empirical software engineering, vol. 19, no. 6, pp. 1665-1705, 2014.

[24] For this study, Avizienis, J.-C. Laprie, B. Randell, and C. In "Basic Concepts and Taxonomy of Dependable and Secure Computing," by Landwehr (IEEE Transactions on Dependable and Secure Computing, Vol. 1, no. 1, pp. 11-33, 2004.

[25] O. Knowledge solutions, Serrat's "The five whys technique," 2010. Pages 307 and 310, Springer, 2017.

[26] "Qualitative methods in empirical studies of software engineering," by B. Seaman, IEEE Transactions on software engineering, vol. 25, no. 4, pp. 557-572, 1999.

[27] P. Fusch, I., and L. Are we there yet?," R. Ness. qualitative study reaches a point of data saturation, The Qualitative Report, Vol. 20, no. 9, p. 1408, 2015.

[28] M. Purposive sampling was first described by D. C. Tongco in his article "Purposive sampling as a tool for informant selection,"

[29] Research on ethnobotany and its practical uses, volume. 5, pp. 147-158, 2007.

[30] J. Morse, M. 2015. "Data were saturated..."

[31] The three of you, L. Johnson: "At what point do we have enough interviews? data saturation and variation in a field experiment," Field Methods, volume, number. 18, no. 1, pp. 59-82, 2006.

[32] L. F. Figueira Filho Singer and M.-A. Storey, "Software engineering at the speed of light: how developers stay current using Twitter," 36th International Conference on Software Engineering, 2014, pp. 211-221.

[33] M. Aniche, Christian Treude, Ian Steinmacher, Ian Wiese, Giuseppe Pinto, Michael A. Storey, and Michael J. 2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE), A. Gerosa, "How contemporary news aggregators help development communities shape and share knowledge." 499–510 (IEEE, 2018).

[34] J. All three authors (Henrich, S. J. Heine, and A. The most peculiar people on Earth?" — Norenzayan.Science of the Brain and Behavior, Vol. 33, no. 2-3, pp. 61-83, 2010.

[35] Coley and R. O'Connor, "Using grounded theory to understand software process improvement: A study of Irish Software Product Companies," in Information and Software Technology, vol. 49, no. 6, pp. 654-667, 2007.

[36] M. The authors (Usman, R. Britto, J. Börstler, and E. As Mendes et al. detail in "Taxonomies in Software Engineering: A Systematic Mapping Study and a Revised Taxonomy Development Method," published in Information and Software Technology, volume 59, issue 1, pages 61-80. 85, pp. 43-59, 2017.

[37] H. Kwasnik, "The role of classification in knowledge representation and discovery," GSLIS. A school in Illinois. .., 1999.

[38] vyshwanara, "Potential Time Lag Issues Due to Raspberry Pi's Missing Hardware Clock," 2018. [Online]. Lack of a hardware clock in the Raspberry Pi might cause scheduling problems; for more information, see https://blog.pisignage.com/.

[39] M. Together, Alhanahnah, C. Stevens, and H. Bagheri, "Scalable analysis of interaction threats in

IoT systems," in Proceedings of the 29th Annual ACM SIGSOFT International Symposium on Software Testing and Analysis, 2020, pages 272-285.

[40] M. Data Mining: Concepts and Techniques, by J. Pei Kamber et al. San Francisco: Morgan Kaufmann Publishers, 2001, volume. 2.

[41] J. Dias, Couto, Paiva, and H. S. Ferreira, "A brief overview of existing tools for testing the internet-of-things," in 2018 IEEE International Conference on Software Testing, Verification, and Validation Workshops (ICSTW). Page numbers: IEEE, 2018.

[42] P. Pontes, Lima, and Pontes, J. For example, see P. Faria, "Test patterns for IoT," in the 2018 proceedings of the 9th ACM SIGSOFT International Workshop on Automating TEST Case Design, Selection, and Evaluation. Pages 63–66.

[43] IOTIFITY [Online]; "Advanced IoT system simulation engine and test automation for enterprise IoT apps." To access, visit https://iotify.io/.

[44] Authors: V. Looga, Z. Ou, Y. Deng, and A. Ylä-Jääski, "Mammoth: A Massivescale Emulation Platform for the Internet of Things," 2012 IEEE 2nd International Conference on Cloud Computing and Intelligence Systems, vol. 3. Reference: IEEE, 2012. pp. 1235–1239.

[45] M. "Arduinounit," by Murdoch. 2013. [Online]. Get it at: https://github.com/. com/mmurdoch/arduinounit

[46] Kravets, "Platformio: An Open Source Ecosystem for IoT Development," PlatformIO.[On-line]. Platformio: https://platformio.com/. org. [Referenced on September 25, 2019] 2018.

[47] "Fit IoT-lab: A large scale open experimental IoT testbed," by R. Pissard-Gibollet, F. Saint-Marcel, G. Schreiner, J. Vandaele, et al., in 2015 IEEE 2nd World Forum on Internet of Things (WF-IoT). Specifically: IEEE, 2015. p.

[48] It was Voas, Kuhn, and P. "Testing IoT systems," by Laplante, at 2018 IEEE Symposium on Service-Oriented System Engineering (SOSE). Pages 48–52 in IEEE's 2018 publication.

[49] Song, V. "The Neverending Death of Smart Home Gadgets," March 2020. [Online]. For more information on the inevitable demise of smart home devices, see https://gizmodo.com/ the-never-ending-death-of-smart-home-gadgets-1842456125.

[50] P. Paul, P. Jabangwe, P. Nguyen-Duc, and P. XP Workshops, 2017, pp. 11-1, Abrahamsson, "Security challenges in IoT development: a software engineering perspective."

[51] Y. Xiao, Yujia Jia, Cheng Xiao, Yu Jia, Jia Yu, and Wei Xue. Author: Lv, "Edge computing security: State of the art and challenges," Proceedings of the IEEE, vol. 107, no. 8, pp. 1608-1631, 2019.

[52] G. A. Hernandez, D. Buentello, Y. According to Jin in "Smart Nest Thermostat: A Smart Spy in Your Home," published in Black Hat USA, issue 2015, 2014.

[53] Z. X. Ling, K. Wu, C. Gao, Y. Xu, J. Luo, and X. Reference: Fu, "Security vulnerabilities of internet of things: A case study of the smart plug system," IEEE Internet of Things Journal, vol. 4, no. 6, pp. 1899-1909, 2017.

[54] S. With the help of M. Siddiqi, R. Notra, V. Sivaraman, H. H. Gharakheili, and R. Boreli, "An Experimental Study of Security and Privacy Risks with Emerging Household Appliances," 2014 IEEE Conference on Communications and Network Security. IEE, 2014, pages 79–84.

[55] Asher Shamir, Alexander Weingarten, Daniel Ronen, and C. O'Flynn, "IoT Goes Nuclear: Creating a Zigbee Chain Reaction," IEEE Symposium on Security and Privacy (SP), 2017. Page numbers: IEEE, 2017.

[56] R. Researchers Goyal, N. Dragoni, and A. Spognardi, "Mind the tracker you wear: a security analysis of wearable health trackers," in Proceedings of the 2016 ACM Symposium on Applied Computing, pp. 131-136.

[57] B. Both Fouladi and S. "Honey, i'm home!" Ghanoun exclaimed.Zwave home automation system hacking," Black Hat USA 2013.

[58] Z. We thank B. Celik, G. Tan, and P. Author: D. McDaniel Citation: "IoTguard: Dynamic enforcement of security and safety policy in commodity IoT." in NDSS, 2019.

[59] Z. For this article, we consulted the expertise of B. Celik, P. McDaniel, and G. Annual Technical Conference (USENIX ATC), 2018, Tan, "Soteria: Automated IoT safety and security analysis," pages 147-158.

[60] A. Alqassem and D. Reference: Svetinovic, "A taxonomy of security and privacy requirements for the internet of things (IoT)," 2014 IEEE International Conference on Industrial Engineering and Engineering Management. Reference: IEEE, 2014. p. 1244-1248.

[61] S. Zhang, Z. Li, Y. Zhang, Q. Deng, S. Ray, and Y. Journal of Hardware and Systems Security, vol. Jin, "Internet-of-Things Security and Vulnerabilities: Taxonomy, Challenges, and Practice," 2, no. 2, pp. 97-110, 2018.

[62] Ahmad, F. Bouquet, E. Fourneret, F. Le Gall, and B. Reference: Legeard, "Model-based testing as a service for IoT platforms," in International Symposium on Leveraging Applications of Formal Methods. 727-742 in Springer (2016).

[63] P. R. Rosenkranz, M. Wählisch, E. Baccelli, and L. Ortmann, "A distributed test system architecture for open-source IoT software," in 2015's Workshop on IoT issues in Mobile and Industrial Systems' proceedings, pp. 43-48.

[64] A. Garca-Domnguez, G. Gutiérrez-Madroal, and I. Software: Practice and Experience, Vol. Medina-Bulo, "Evolutionary Mutation Testing for IoT with Recorded and Generated Events," p. 49, no. 4, pp. 640- 672, 2019.

[65] According to research by Gutiérrez-Madroal, I. Medina-Bulo, and J. "IoT-teg: Test event generator system," by J. Domnguez-Jiménez, appeared in Journal of Systems and Software volume 2017. 137, pp. 784-803, 2018.

[66] B. The authors Morin, N. Harrand, and F. IEEE Software, vol. Fleurey, "Model-based software engineering to tame the IoT jungle," 2016. 34, no. 1, pp. 30-36, 2017.

[67] Le Pallec, Radu Mateescu, Laurent Noirie, and Gurumaa Krishna. "IoT composer: Composition and deployment of IoT applications," by Salaün, will appear in the Companion Proceedings of the 2019 IEEE/ACM International Conference on Software Engineering (ICSE-Companion). 19–22 in IEEE's 2019 publication.

[68] Corno, Louise De Russis, and John R. P. Sáenz, "Towards computational notebooks for IoT development," in Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems, 2019, pp. 1–6.

[69] R. An article by Zhang, W. Xiao, H. Zhang, Y. Liu, H. Lin, and M. "An empirical study on deep learning jobs program failures," by Yang, to appear in the proceedings of the 42nd IEEE/ACM International Conference on Software Engineering (ICSE) in 2020. pages 1159–1170 in IEEE's 2020 publication.

[70] The authors (Zhou, J.-G. Lou, H. Zhang, H. Lin, H. Lin, and T. 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering, "An Empirical Study on Quality Issues of Production Big Data Platform," Qin. 2. 2017 IEEE, pages 17–26.

[71] F. Researchers S. Ocariza, K. Bajaj, K. Pattabiraman, and A. According to Mesbah's research, "A study of causes and consequences of client-side javascript bugs," published in IEEE Transactions on Software Engineering, volume 2, issue 1, is a good example. 43, no. 2, pp. 128-144, 2016.

[72] K. Those authors (Chan, Bishop, Steyn, Baresi, and Chan) are M. Presented at the International Conference on Service-Oriented Computing with the title "A fault taxonomy for web service composition," by Guinea. Pp. 363.–375. Springer, 2007.

[73] W. We thank X.-B. D. Le, X. Xia, Y. Feng, Z. Chen, and B. Zou for their contributions to this work. Xu, "Smart contract development: Challenges and opportunities," IEEE Transactions on Software Engineering.

Cite this article as :

Sandhya Devi, Dr. Dev Singh, "Study of Various Known Bugs and Other Challenges Associated with IoT System Development ", International Journal of Scientific

Research in Computer Science, Engineering and Information Technology (IJSRCSEIT), ISSN : 2456-3307, Volume 9, Issue 5, pp.125-139, September-October-2023.