IoT Bugs and Development Challenges

Amir Makhshari
University of British Columbia
Vancouver, BC, Canada
amirosein@ece.ubc.ca

Ali Mesbah
University of British Columbia
Vancouver, BC, Canada
amesbah@ece.ubc.ca

Abstract—**IoT systems are rapidly adopted in various domains, from embedded systems to smart homes. Despite their growing adoption and popularity, there has been no thorough study to understand IoT development challenges from the practitioners' point of view. We provide the first systematic study of bugs and challenges that IoT developers face in practice, through a large-scale empirical investigation. We collected 5,565 bug reports from 91 representative IoT project repositories and categorized a random sample of 323 based on the observed failures, root causes, and the locations of the faulty components. In addition, we conducted nine interviews with IoT experts to uncover more details about IoT bugs and to gain insight into IoT developers' challenges. Lastly, we surveyed 194 IoT developers to validate our findings and gain further insights. We propose the first bug taxonomy for IoT systems based on our results. We highlight frequent bug categories and their root causes, correlations between them, and common pitfalls and challenges that IoT developers face. We recommend future directions for IoT areas that require research and development attention.**

Index Terms—**Internet of Things, Software Engineering, Mining Software Repositories, Empirical Study**

I. Introduction

Internet of Things (IoT) envisions a self-configuring, adaptive, and complex network that interconnects smart objects, embedded with sensors or actuators, to the internet through the use of communication protocols [1]. By 2020, Gartner estimates that smart inter-connected devices will outnumber humans 4-to-1 [2] and it is estimated that by 2025, there will be over 75.44 billion smart things worldwide [3]. These smart "*things*" can be programmed and remotely controlled to collect their data or to control their actions. Programming physical devices with constraint resources, dealing with diverse network protocols, and the integration of diverse entities in IoT systems, add unique characteristics to the challenges of developing such systems. Driven by the above considerations, the concept of bugs in IoT is more complicated than traditional software.

Previously, some studies have investigated the characteristics of IoT repositories [4], and discussed some challenges of IoT systems [5]–[7]. Existing research on bug categorization is limited to specific sub-domains of IoT such as bugs in smart aquaculture systems [8], bugs in IoT device operating systems [9], or bugs uncovered during the deployment of IoT systems [5].

While more mature software domains have benefited from empirical and qualitative studies on their bugs and developer challenges [10]–[12], no such study is available for IoT to the best of our knowledge.

In this paper, we provide a generalized and systematic overview of bugs and developer challenges in IoT systems. In order to do so, we mine IoT GitHub repositories and collect (5,565) bug reports from 91 representative IoT projects. By applying Root Cause Analysis (RCA), we categorize a subset of 323 bug reports considering the observed failures, root causes, and locations of the bugs. We propose the first taxonomy of bugs in IoT systems, which is constructed by analyzing these real-world IoT bugs. To complement the taxonomy and study the challenges of IoT developers, we conducted semi-structured interviews with nine IoT practitioners that have hands-on experience in different layers of IoT. Lastly, we validated our findings through an online survey that was completed by 194 IoT developers.

The contributions of this paper are:

- An empirical study to understand IoT failures and their root causes in practice
- The first IoT bug taxonomy
- An overview of state-of-the-practice challenges faced by IoT developers

Our findings show that general development issues, device management issues, and messaging issues are the most frequent bug categories. Furthermore, the most frequent root causes of bugs are software programming faults, semantic faults, and dependency faults. In addition, we highlight the challenges of IoT developers in various areas such as testing, debugging, and dealing with heterogeneity and security in IoT.

II. Background and Motivation

Figure 1 shows a typical architecture of an IoT system [1], [7], [13], [14].

Device layer. The device layer at the bottom includes smart programmable things interacting with the physical world through their embedded sensors and actuators. Some IoT devices employ light embedded operating systems (e.g. contiki, RIOT, TinyOS) that have support for various programming languages allowing developers to write embedded code on top of the device OS [15], while bare metal IoT devices run the embedded code directly on their hardware processor.

Edge layer. This layer contains gateway devices with fewer resource constraints with the ability to handle telemetry data collection, processing, and routing locally on the edge.

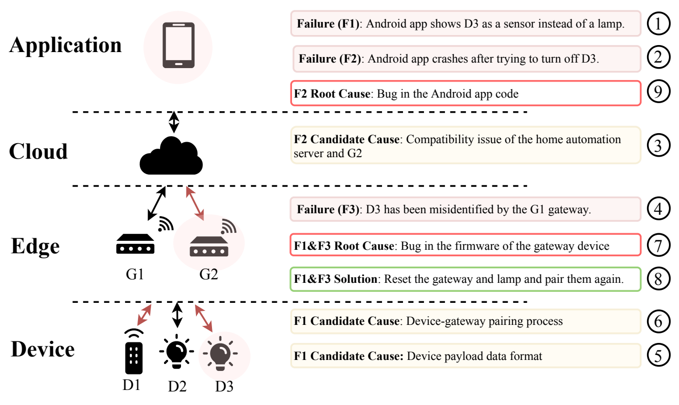

Application
Failure (F1): Android app shows D3 as a sensor instead of a lamp.
Failure (F2): Android app crashes after trying to turn off D3.
F2 Root Cause: Bug in the Android app code
Cloud
F2 Candidate Cause: Compatibility issue of the home automation server and G2
Edge
Failure (F3): D3 has been misidentified 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.
G1
G2
Device
F1 Candidate Cause: Device-gateway pairing process
F1 Candidate Cause: Device payload data format
D1
D2
D3
1
2
9
3
4
7
8
6
5

Figure 1: A typical layered architecture of IoT systems and an example of an IoT bug

Gateway devices can handle the device-device and device-cloud interoperability by interpreting diverse communication protocols such as MQTT, CoAP, and HTTP [16].

Cloud layer. Remote IoT cloud servers accumulate and process all telemetry data, and communicate with heterogeneous IoT devices to control and monitor them remotely. IoT cloud servers rule engine lets users write automation logic between IoT devices to define interoperability behaviours of the IoT system [17].

Motivating example. We use a real IoT bug to motivate our study of bugs and developer challenges in IoT systems. Figure 1 shows (on the right side) the steps that IoT developers have taken to find the root cause of the bug report PYTRADFRI/135 [18]. This bug occurred in a smart home environment where different devices should be connected to a home automation server with the help of a gateway device.

Based on developers' discussions, the first manifestation of this bug happens at the application layer. A light bulb device (D3) is mistakenly recognized as a sensor device (F_1) as the user adds it, and also the Android app crashes (F_2) once it tries to turn the light bulb off (step 1 and 2). Initially, the developers suspected the root cause of the bug to be the incompatibility of a gateway library with the home automation server (step 3).

However, further investigation of the failure in the edge layer (step 4), revealed that the gateway misidentifies D3 as a remote controller (F_3). Since the gateway in this system relies mainly on a specific format of response data from devices to identify their types properly, potential inconsistencies of the payload data from the subject device were also investigated (step 5).

Also, since the device battery and distance to the gateway could affect the pairing process, developers tried pairing in different circumstances (step 6). After none of the candidate causes showed as the potential reason for the bug, the root cause was identified as an external bug in the firmware of the G2 gateway device (step 7) and resetting both the device and the gateway and pairing them again solved the issue (step 8). But surprisingly F_2 continued to exist. After monitoring the data that the app receives via Wireshark, the root cause of F_2 was identified as a fault in the Android app code (step 9).

As we can see in this example, distribution of failures and candidate causes in different layers, dealing with behaviours of diverse devices such as comparing their payload, and depending on naive debugging practices such as monitoring network traffic, make it extremely hard for developers to deal with IoT bugs. Despite these compelling challenges, previous studies [5]–[9] do not take into consideration real-world experiences of IoT developers. In this paper, we aim to provide a systematic and generalized understanding of bugs and challenges in IoT by mainly considering IoT developers' experiences.

III. Methodology

Our goal is to characterize software bugs in IoT systems and understand the challenges developers face in practice. To this end, we address the following research questions in this work:

- `RQ1`: What are the classes of bugs in IoT systems?
- `RQ2`: What challenges do IoT developers face in practice?

In order to answer these questions, we conduct an empirical investigation consisting of two phases. In the first phase (RQ1), we analyze 323 issues and pull requests (PR) from open-source IoT projects. We use the findings to form the first taxonomy of bugs in IoT systems. In the second phase (RQ2), we conduct a qualitative study through (1) semi-structured interviews with IoT developers to discover new categories of bugs and challenges, (2) a survey of IoT developers to validate the findings and gain new insights. All our quantitative and qualitative data is available online [19]

A. IoT Bug Categorization

Collecting bug reports. The initial step is to find repositories that are representative of IoT projects. We employed the "GitHub topic feature" to find IoT-related repositories. According to GitHub's official website [20], Topics are labels that create subject-based connections between GitHub repositories.

We searched among topics with related keywords such as "internet-of-things" and "IoT" and we added the top three topics "IoT-application", "IoT-platform", and "IoT-device" from the results to our list of targeted topics. Initially, we collected 8,774 repositories using these five topics in January 2020. We excluded repositories with less than 10 stars [21], resulting in 1,356 repositories to consider.

To only consider valid bugs, we looked for issued bug reports with only "closed" status and with "bug"', "defect", or "error" labels. Moreover, to select only representative IoT repositories for our study, we manually analyzed repositories that have more than five labeled issues or more than 50 closed issues based on the information in their readme page, issued bug reports, and their website (when available). We then excluded projects that were not representative of IoT systems such as user interface, documentation, or outdated repositories. Our final project list contained 91 open source IoT repositories to analyze. For five repositories that use custom labels (e.g., "problems", "kind/bug", "type : bug"), we manually added their labels to our search keywords. In the end, we collected *5,565 bug reports* from the 91 IoT repositories.

Our subject IoT repositories together cover all the layers of the IoT systems' architecture depicted in Figure 1. The most popular programming languages among the subject repositories are Python (21%), Java (18%), JavaScript (17%), C (13%), and C++ (13%). A few of them use other programming languages such as Go, Ruby, and C#. The selected repositories are also diverse in terms of the number of stars and forks they have. In February of 2020, 32% of our subject GitHub repositories had more than 500 stars, 40% between 50 to 500 stars, and 28% between 10 to 50 stars.

Labeling. For each bug report in our dataset, we created a JSON object containing failure(s), cause(s) of the failure, and location(s) of the faulty code. Failure refers to any observable unexpected behavior of the system that is against the correct functionality of that system [22], [23]. To explore the causes of the failure, we did RCA on each bug report using the *five whys technique* [24]. Based on this technique, multiple causes can contribute together or at different levels to a visible failure in the system. Following this approach, we started from the failure and repeatedly asked "why" until we reached the root cause of the problem. In the case of software failures, root causes are often developers' faults in the design or implementation of the IoT system. To label the location(s) of faults, we used the architecture defined in Section II as our reference.

Bug reports were manually labeled by the author(s) individually following the open coding procedure [25]. We followed an iterative process for labeling where in each iteration, we randomly sampled new instances from the collected bug reports and labeled them. After each labeling iteration, all potential conflicts in labels between the author(s) were resolved. We continued this process until bug categories reached a state of saturation where no new category appeared [26].

We also flagged and discarded issues and PRs that could not represent a bug or a bug-fix such as enhancements or how-to-use questions. We examined the entire discussion among developers, as well as the fix commit data (e.g. the commit message and the code diff) to label each bug report. At the end, we labeled *323 bug reports*.

Table I: Interview Participants

ID	Role	IoT Systems Type	Projects Domain	Dev Exp (yr)	IoT Dev Exp (yr)
P1	Software and hardware lead	Full-stack	Smart home	13	7
P2	Hardware lead	Hardware	Education	15	5
P3	Software dev	Full-stack	Smart home	5	4
P4	Software dev	Middleware	Smart city	3	3
P5	Software lead	Full-stack	Smart home	20	17
P6	Software dev	Cloud	Not domain-specific	10	3
P7	Software and hardware dev	Full-stack	Smart home	20	3
P8	Software lead	Full-stack	Smart home	11	4
P9	Software lead	Cloud	Not domain-specific	20	12

B. *Interviews*

While manual analysis of bug reports and developers' discussions provided useful insights into the characteristics of IoT development, there was still a possibility that our analysis was restricted to only code-level issues and we might miss high-level problems of IoT developers. To mitigate this issue, we conducted semi-structured interviews with IoT developers to reveal new bug categories and collect their development challenges to complement our results.

Participants. We used purposive sampling [27] to recruit developers with adequate experience in developing IoT systems. To collect interview participants, we employed GitHub as it provides a diverse pool of developers and their contributions to different projects. We obtained valid email addresses of only the top three contributors to popular open-source IoT repositories as our candidate interviewers.

We contacted candidates through emails and conducted interviews until we reached data saturation, where we had sufficient data to replicate the study and further data collection is unnecessary [26]. We relied on this widely-applied methodological principle to decide when to stop interviewing [28], [29] as it is also used in other qualitative studies in software engineering [30], [31]. We interviewed people with different development backgrounds and experiences before deciding about the data saturation to consider variability in experimental results across different populations [32].

Table I presents all nine interview participant's experience and field of expertise in IoT development. For a high-level picture of their general development background, the lowest value is three years and the highest value is 20 years (avg=13, sdv=6.4). In terms of IoT development experience, the lowest value is three years and the highest value is 17 years (avg=6.4, sdv=4.6). Participants' IoT development experience covers all sections of IoT systems spanning from hardware to middleware, cloud, and end-applications. In addition, their projects cover a variety of domains such as smart home and Industrial IoT (IIoT).

Protocol. Since our goal for conducting interviews was to be open to new data, and we did not have the definitive structure of bug categories, we conducted interviews following a semi-structured approach. Interviews started with some questions about participants' IoT development background and their field of expertise in IoT. This information could help us improvising insightful questions during the technical section of the interviews. The technical section had a combination of both open-ended and specific questions about different categories of bugs and challenges in IoT development. Our strategy was to start with open-ended questions to avoid biasing participants toward our findings, and then we gradually shifted to more structured and predefined questions during the interview process. With participants' consent, we recorded the audio and video of all the interviews for later analysis. All the interviews were conducted remotely through Zoom. Interviews took around 43 minutes on average ranging from 31–70 minutes. We used Descript, an automated speech detection tool, to generate transcribes of the interviews and we did manual corrections afterward in case of any mistakes in the automatically generated transcripts.

Analysis. As the primary objective of this study is to generate theories from the experiences of IoT practitioners instead of using pre-conceived theories, we followed the grounded theory methodology [33] to ensure the quality of the generated theory. Our analysis steps consist of iteratively (i) collecting qualitative data from the interviews (ii) analyzing the interview transcript line by line and assigning labels (tags) to distinct units of meanings, and (iii) identifying emerging categories and relating categories to their subcategories while continuously comparing all the previously analyzed data with the emerging theories. These steps were repeated for each interview. On average, we extracted 18 tags per interview. Potential conflicts in the labels were resolved after each iteration by the authors.

C. Validation Survey

In order to make sure that our findings are generalizable, comprehensive, and representative, we involved more IoT developers through an online survey.

Participants. We sent our online survey to developers that have made at least three contributions to the collected IoT repositories in subsection III-A and to IoT developer groups in social media platforms such as Linkedin and Facebook, and online forums. Our survey was online between 19 July and 19 August 2020. The survey, as distributed to participants, is available in our artifact package [19].

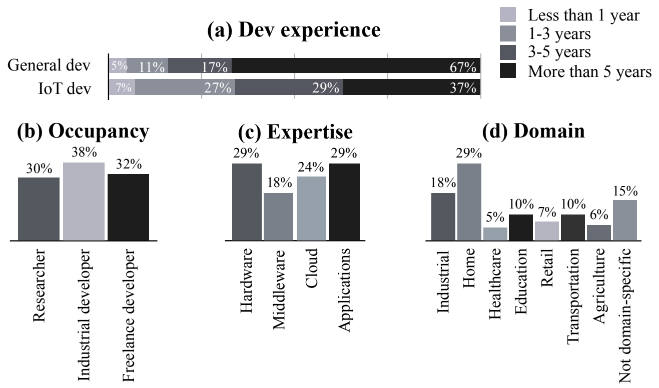


Figure 2: Development backgrounds of the survey participants

Protocol. The survey has three sections. In the first section, we collect participants' background in IoT, and in general development, which is depicted in Figure 2. The second section concerns the challenges of developing IoT systems with questions aimed at correlating our findings with the participants' own experiences.

The third section is about the frequency and severity of bug categories based on participants' previous experience in IoT development. At the end of each section, there are open-ended questions to allow them to share their comments about our results and mention new categories.

Analysis. Our survey was completed by 194 respondents, with a response rate of around 10% for valid responses. We received 95 comments through the open-ended questions sections. All of the survey respondents' comments are coded and analyzed following the same procedure discussed in subsection III-B.

IV. Findings: IoT Bug Categories (RQ1)

In this section, we describe our findings regarding IoT bugs.

A. Taxonomy of Bugs

We used all the tags collected by RCA of the bug reports in our dataset, to build a taxonomy of bugs in IoT systems. As our motivating example illustrates Figure II, IoT bugs can be multi-faceted and manifest at different layers and locations. Therefore, we designed our bug taxonomy to accommodate all these bug characteristics. Considering various approaches suggested by Usma et. al. [34] for taxonomy construction, we followed the approach suggested by Kwasnik [35] as IoT bugs are multi-faceted and relatively a new and unexplored concept. Following their approach, we first defined facets of our classification as all failures and locations of failures. Then, we analyzed all bug reports based on these facets and built a hierarchical taxonomy that accommodates all the dimensions.

After we built the initial version of the taxonomy, we used the data from the interviews and the survey to complement and enhance the taxonomy. We reviewed all previously tagged data and re-tagged them after each alteration of the taxonomy. Figure 3 depicts our IoT bug taxonomy. Next, we describe the major bug categories in our taxonomy. We will use specific bugs as examples for each category. All these bug examples are available in our dataset, which is available online [19].

IoT Device. This category of taxonomy covers bugs that are related to IoT device hardware and firmware.

Device hardware: Bugs in this subcategory are related to the physical aspects of IoT devices. Examples include bugs related to wiring issues, device pin status issues, or issues with physical sensors and actuators of the device. For example, PEDALINOMINI/34 is related to the device not differentiating between single and double presses of a hardware button. Other common bugs in this category are those that are linked to the device's limitations in memory, power consumption, or processing capacity. One such instance provided by P_1 as he described a scenario where a device on low battery generated incorrect data to the cloud. There are various similar cases in our collected bug reports where the low battery of the device or the removal of the power source of the device causes failures. In addition, heavy calculations and processing on the device (HOMIE-ESP8266/575, interviewee P_8) or running out of memory (ZWAVE2MQTT/141, interviewee P_7) are other examples of bugs related to this category. Another example of known hardware issues of IoT devices is the timing issues of Raspberry Pi devices [36], which is also mentioned by an interviewee (P_8).

Device firmware: Firmware bugs consist of three subcategories. 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 type of 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-the-air (OTA) updates of the device firmware, stale updates, or updating the device firmware with the wrong binary have caused failures of the IoT system. In some cases, the stale update issues are related to device configuration issues as in WTHERMOSTATBECA/54, where the device needs to be re-configured with WiFi credentials after each firmware update, otherwise, future firmware updates would be stale.

Compatibility. When a bug occurs only on a specific type of device, communication protocol, or third-party component, it falls under the compatibility category. For instance, a common device incompatibility issue happens when certain devices represent their telemetry data in different formats, leading to the other components not being able to process their data. Other common bugs are linked to compatibility issues of certain combinations of sensors and development boards, e.g., incompatibility of the DHT temperature sensor with the ESP32 microcontroller in MONGOOSE-OS/277. Issues with the interoperability of different protocols is another case. One example is MAINFLUX/1079, which is related to the interoperability between the HTTP and MQTT protocols. A common bad practice in IoT development related to these issues is developing protocol-specific or device-specific code. For instance, in DEVICE-OS/1938, the IoT platform relies on event components to report what protocols each event is intended for, in order to be able to run different functions for each protocol individually. However, sometimes developers have no other choice but to follow this error-prone approach, just to bypass the limitations of third-party devices. For instance, (P_2) mentioned a case where the incompatibility of the Raspberry Pi and some types of sensors had forced their developers to implement custom logic for the communication of these devices. Developers had to switch between Raspberry Pi's default implementation and their own custom implementation based on the sensor type, leading to many issues.

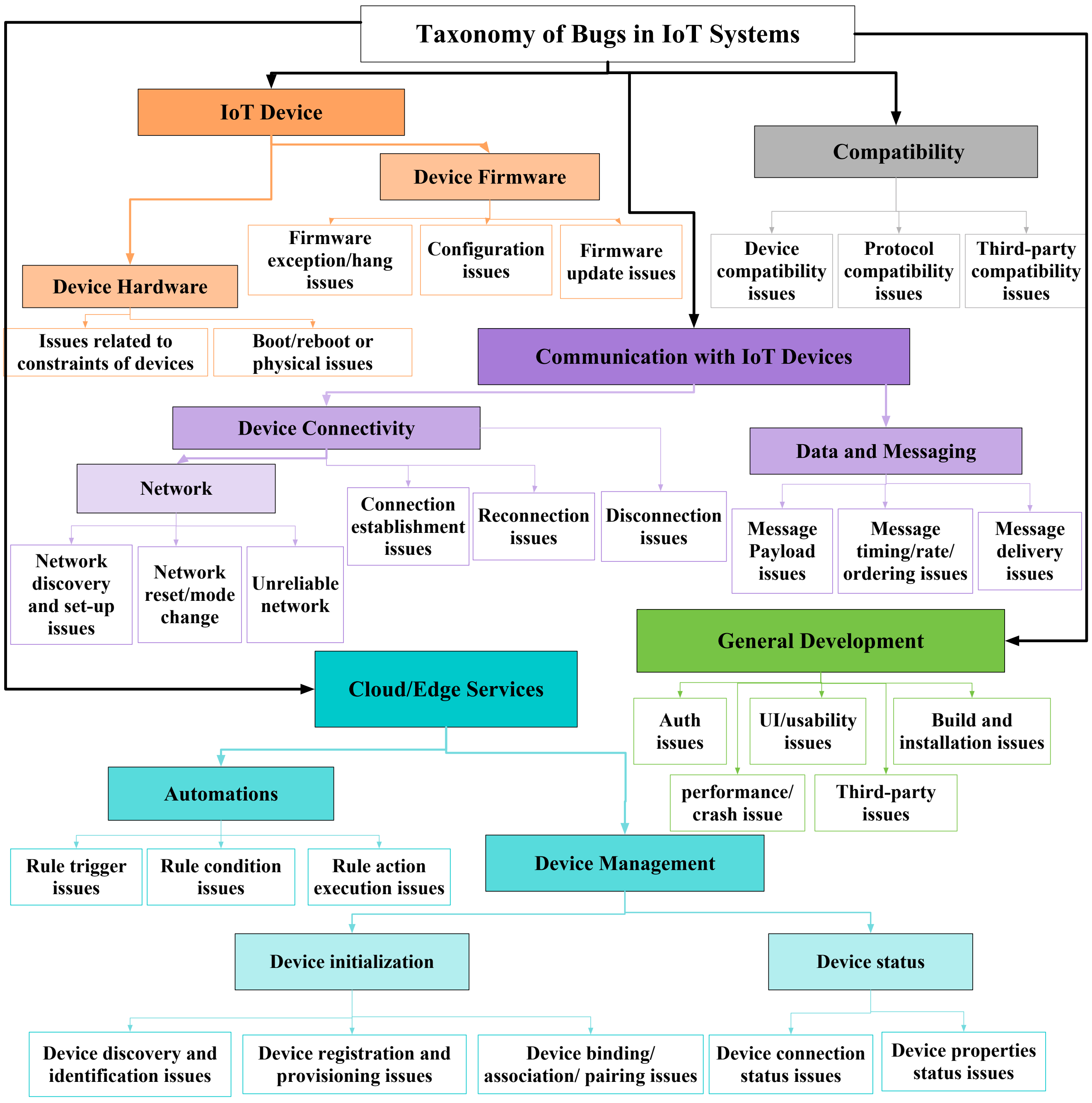
Communication with IoT devices. Bugs that are related to the communication of IoT devices with each other or with other entities fall under this category. Generally, there are two types of bugs in this category:

Device Connectivity: Some of the connectivity issues are related to the network that the device relies on for connecting to the internet. One example is when the device cannot discover a valid and available network such as a local access point and therefore loses access to the internet. As it is also mentioned by P_9 *"When the device location is changed to another room or another building, the device has to be reconfigured for the new access point."* In addition to the network discovery, not handling a network reset, or unstable and unreliable networks are other common issues that can lead to failures. However, Sometimes IoT devices fail to establish a valid connection to the gateway or remote cloud servers despite a valid network status. Failure in reconnecting, connection refreshing, and ensuring such connectivity failures do not cause propagated failures in other components are other pitfalls that IoT developers often deal with. Additionally, unexpected disconnection or connection closure issues are other manifestations of bugs in this category. Two interviewees (P_6 and P_9) believe that connectivity bugs are the most serious and challenging bugs. As P_9 states *"the weakest part of our IoT platform is to communicate with IoT devices"*

Data and Messaging: This category includes bugs that are related to data and message sending in the IoT system. Typically, messages are either commands that are sent to IoT devices via the cloud, or they are telemetry data that are received from IoT devices in the edge, cloud, or applications. Some bugs cause failures in delivering these messages from the sender to the receiver. Some other messaging bugs are related to the timing of messages. For instance, various reported bugs are related to the rate and order of the messages. Additionally, some bugs are linked to the payload that is being delivered through the messages. In some cases, payload size or format are the causes of failures. There are also cases where there are violations of payload integrity by messages being truncated or overwritten.

Cloud/Edge Services. This category includes bugs that are related to the services delivered by the remote cloud servers or gateway devices in the edge layer.

Device Management: To monitor and control each IoT device remotely, devices should be connected to a cloud server or a hub device, and report their status while listening to user commands. Device management (DM) issues include problems that cause failures in this process. The first class of DM issues happens in the stage of initializing the IoT device in the cloud or edge systems. One type of device initialization (DI) issue is when the IoT device is not properly or uniquely identified by the cloud or edge components and therefore causes further failures in the subject IoT system. Besides, if the IoT device fails to provide a recognizable identity and valid permissions to the cloud or edge, it would not be allowed to use remote services. Some examples of device registration and provisioning bugs are duplicate device certificates, issues with auto-provisioned devices, or failure in retrieving data from the provisioning service. Another class of DI bugs is problems with binding, association, and pairing of IoT devices. There are several cases where bugs are introduced to the IoT system just because devices are grouped together (such as devices in one room), due to not properly handling the association of a sensor device with a physical object. There is an example mentioned by one of our interviewees (P_5) where two switches were associated with one lamp and only one switch was working due to issues with addressing multi-instance devices with labels. The second class of DM issues is related to problems with monitoring the status of IoT devices. One type of device status is the connectivity status,



Taxonomy of Bugs in IoT Systems
IoT Device
Device Firmware
Firmware exception/hang issues
Configuration issues
Firmware update issues
Device Hardware
Issues related to constraints of devices
Boot/reboot or physical issues
Compatibility
Device compatibility issues
Protocol compatibility issues
Third-party compatibility issues
Communication with IoT Devices
Device Connectivity
Network
Network discovery and set-up issues
Network reset/mode change
Unreliable network
Connection establishment issues
Reconnection issues
Disconnection issues
Data and Messaging
Message Payload issues
Message timing/rate/ ordering issues
Message delivery issues
General Development
Auth issues
UI/usability issues
Build and installation issues
performance/ crash issue
Third-party issues
Cloud/Edge Services
Automations
Rule trigger issues
Rule condition issues
Rule action execution issues
Device Management
Device initialization
Device discovery and identification issues
Device registration and provisioning issues
Device binding/ association/ pairing issues
Device status
Device connection status issues
Device properties status issues

Figure 3: Taxonomy of IoT bugs.

to check whether the device is online, which is also known as the heartbeat check. Some examples of bugs in this type are wrong device heartbeat rate, showing a lost connection as live and vice-versa, or not notifying other components when the device goes offline. Failure in retrieving the device status such as color and brightness for a light bulb, showing the status incorrectly, or failure in updating the device status are some examples of these types of issues.

Automation: This bug category is related to automation services that IoT cloud or edge platforms provide and it is classified into the trigger, condition, and execution issues. Rule trigger defines a condition under which a rule is initiated. Trigger failures usually cause a rule not to become triggered when it should be (SMARTHOME/5578) or vice-versa (HOME-ASSISTANT-CONFIG/2). Rule condition is the statement that should be checked when the rule is triggered. Examples of the rule condition issues are problems in retrieving the device state to check the rule condition since the condition usually relies on device latest status (TESLA-API/43). Issues in the execution of the rule action are the last and the most prominent automation issues. Some examples of these issues are crash after rule action execution, issues in handling asynchronous behavior and threads in rules, and having problems with the output of the rule being unpredictable or nondeterministic.

General Development. This category captures common development bugs. Some common issues are problems with installing, compiling, or building a project as well as unexpected crashes or performance issues in the IoT project. The general development category also includes bugs in the authentication or authorization process. One of the IoT-specific authorization issues are problems with generating, signing, or maintaining the certificates that devices have to present for using cloud or edge services (AZURE-IOT-SDK-C/657). Other sub-categories of general development bugs are UI-related, usability, or external issues.

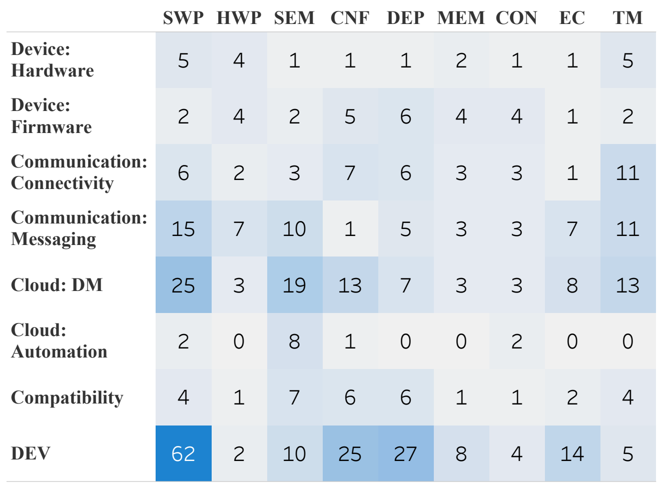
B. Characteristics of Bugs

Root causes. Figure 4 shows the distribution of bug categories and root causes. The most frequent categories of bugs are general development issues (48%), device management issues (29%), and messaging issues (19%). In terms of root causes, after general software programming faults (SWP) such as syntax issues, semantic programming faults (SEM)

are the most dominant root causes of the bugs. Some semantic mistakes that IoT developers make are wrong control flow, functionality logic, or return values. However, some of the semantic faults are related to the automation logic of the IoT system, such as logical faults in automation apps, which are also discussed in recent studies [37].

The next frequent root cause is dependency faults (DEP), where developers use wrong versions of the software or firmware libraries, tools, devices, or protocols.

One of the most important root causes often leading to hardware, connectivity, and messaging issues are timing faults (TM). Improper handling of time-outs or rate of operations,



	SWP	HWP	SEM	CNF	DEP	MEM	CON	EC	TM
Device: Hardware	5	4	1	1	1	2	1	1	5
Device: Firmware	2	4	2	5	6	4	4	1	2
Communication: Connectivity	6	2	3	7	6	3	3	1	11
Communication: Messaging	15	7	10	1	5	3	3	7	11
Cloud: DM	25	3	19	13	7	3	3	8	13
Cloud: Automation	2	0	8	1	0	0	2	0	0
Compatibility	4	1	7	6	6	1	1	2	4
DEV	62	2	10	25	27	8	4	14	5

Figure 4: Distribution of bug categories (vertical) and root causes (horizontal)

wrong time-out values for connection closures, or not handling asynchronous behaviors are among timing-related root causes.

Some faults that are more specific to hardware programming such as interrupt handling, are assigned to hardware programming faults (HWP). Faults in handling exceptional cases (EC) are another root cause for IoT bugs that include mistakes in handling corner cases (large or out of range data), not handling errors properly, or not handling changes of the requirements or changes in third-party components. Finally, the remaining root causes are related to memory faults (MEM), concurrency faults (CON), and configuration faults (CNF).

Correlations among bug categories. During our analysis, we observed some frequent patterns of certain bug categories appearing together more often. To study the correlations between bug categories, we used Lift [38], a statistical metric introduced by Han and Kamber that computes the probability of two categories appearing together. For each pair of bug category, a lift value of more than 1 shows a positive correlation, and a lift value below 1 reveals a negative correlation.

Table II shows the lift values of correlated bug category pairs. The top correlated bug categories are [hardware, firmware], [hardware, connectivity], and [firmware, connectivity]. This correlation analysis, besides helping IoT developers in debugging, gives insight into how intertwined IoT bugs can be in practice.

Frequency and severity of bugs. We asked our survey participants whether and how frequently they face each bug (sub)category and what are their perceived severity based on the impact on the IoT system and fixing-time. Table III shows the results. All the bug categories in our taxonomy have been faced by at least 82% of IoT developers, which shows the bug categories are representative of the real-world bugs in IoT systems. Connectivity issues are the most frequent and severe bug category with more than 97% of IoT developers have faced it at least once.

Table II: Bug Categories with Positive Correlation

Bug Category	Bug Category	Lift Value
Device: Hardware	Device: Firmware	3.86
Device: Hardware	Communication: Connectivity	2.57
Device: Firmware	Communication: Connectivity	2.25
Compatibility	Cloud: Device Management	1.84
Compatibility	Communication: Messaging	1.44
Communication: Messaging	Device: Firmware	1.42
Cloud: Device Management	Automation	1.38
Cloud: Device Management	Device: Hardware	1.28
Communication: Connectivity	Communication: Messaging	1.22
Communication: Connectivity	Compatibility	1.14

Device-related issues are the least experienced bug categories, but they are the most severe bugs after connectivity issues. According to the survey respondents, automation issues are the least severe bugs, however, more than 91% of IoT developers have faced them at least once. The compatibility issues are the least frequent bugs according to IoT developers' experiences.

Regarding sub-categories, device initialization issues are the most frequent and severe device management bugs since about 95% of IoT developers having dealt with them.

Also concerning the device status issues, bugs related to the status of connection have shown more frequency while bugs related to the status of device properties are more severe based on survey respondents. Among device-related issues, bugs that are related to the constraints of IoT devices have the most frequency. The device firmware exception issues are the most severe device-related bugs. Concerning general development issues, installation issues are the most frequent and severe bugs.

Taxonomy augmentation. Regarding IoT bugs, we collected 79 tags from interviews and 18 tags from the survey comments. Device binding issues, performance issues, and third-party compatibility issues were not discovered before the interviews and were added by interviewers' experience. The survey comments did not reveal any new information to be added to the taxonomy. However, the extracted tags, from both the interviews and survey, helped us to characterize each bug category by providing contextual data.

V. RQ2 Findings

In this section, we provide our findings regarding the challenges faced by IoT developers.

A. Testing and Debugging Challenges

Relying on access to the real device. According to seven interviewees, several GitHub issues (DEVICE-OS/1871, MY-CONTROLLER/485, TESLA-API/43), and 74% of survey participants, IoT developers rely on access to devices to test and debug their IoT system, by tasks such as manual reset or device output monitoring ($P_{2,3,7}$).

In some scenarios, devices are out of reach or in hard-to-access locations thus making remote debugging more essential. Four interviewees believed that practical simulation solutions are required for better IoT testing and debugging.

Table III: Survey Results: Bug Taxonomy

Bug Category	Have faced	Frequency			Is Severe
		Frequently	Sometimes	Rarely	
Device: Hardware	82.50%	30.30%	39.06%	30.64%	43.61%
Device: Firmware	85.75%	24.20%	44.48%	31.32%	42.78%
Communication: Connectivity	97.22%	50.29%	38.29%	11.42%	62.78%
Communication: Messaging	92.22%	33.74%	36.14%	30.12%	40.56%
Cloud: Device Management	90.93%	25.06%	46.43%	28.51%	40.75%
Cloud: Automation	91.67%	20.00%	44.24%	35.76%	32.22%
Compatibility	86.11%	23.88%	40.00%	36.12%	36.11%
Dev	87.89%	24.53%	39.44%	36.03%	38.89%

As P_5, a middleware developer stated *"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."* Also as $P_{5,8,9}$ stated, current simulation solutions in IoT are not mature enough and they are only valid for limited scenarios, such as testing high-level controllers or small unit tests, rather than being suitable for all levels of testing such as system testing.

Some relevant challenges mentioned in the survey are *affording to have all types of IoT devices*, *complex custom logic for effective IoT device mocking and simulation*, and *setting up test environments with IoT devices*.

Fault localization. According to eight interviewees, nine survey comments, and also half of the survey participants, fault localization is a barrier due to lack of transparency in the operations of IoT systems. As P_7 mentioned *"there is no environment that logs everything."* One contributing factor to this is the difficulty of tracing executions of numerous external components in IoT systems. P_3 mentioned using open-source solutions just to be able to log everything.

Another factor that impacts fault localization is the existence of hidden failures.

As an example, P_7 mentioned *"It's hard to recognize on the app that the temperature the device is reporting now is for several minutes ago."* Besides, $P_{2,4}$ mentioned examples of failures that only show up after the device has worked for a specific amount of time (five minutes for P_2, several hours for P_4), which is also observed in GitHub issues (DEVICE-OS/1926, ZWAVE2MQTT/141, VSCP/207). This issue makes IoT failures more unpredictable and may hide developers' faults. Another issue toward fault localization is the lack of tools and developers' support. For instance, P_3 inspects device messages in bit-level by monitoring communications using Wireshark. P_2, as a hardware platform developer, said *"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."*

Reproducing IoT bugs. In addition to observing several GitHub discussions (DITTO/414, TESLA-API/68), we collected four tags from interviews, and three survey comments regarding the challenge of reproducing IoT bugs. Besides the already mentioned factors which harden bug reproduction, such as limited access to devices or hidden failures, some bugs only happen with a specific device setting or with certain environments of the IoT system. IoT developers cannot reproduce these bugs unless they have exactly the same setting or environment. For instance, a survey comment indicated

"It's hard to reproduce some memory-related bugs in X86 devices when they have ASLR enabled." Also, we observed other examples such as TEMPERATURE-MACHINE/13: *"I can reproduce the bug with the help of ice packs taken at three different temperatures from my freezer."*

Combinatorial explosion. In addition to all the evolving components in traditional software, such as libraries and operating systems, there are more changing factors in IoT systems. Hardware devices produced by various manufacturers with different standards, device integration middlewares, and communication protocols are some examples of these extra changing factors in IoT. With all these components releasing new versions at a specific rate, a combinatorial explosion problem is likely to happen when developers want to cover all possible combinations with test cases. One relevant statement is *"I do not have all kinds of bulbs, remotes, and sensors, so I could be completely wrong!"* in a GitHub discussion (PY-TRADFRI/135). We could collect eight tags come from four interviewees and two tags from survey comments regarding the challenge of combinatorial testing. As one example, P_9 said *"We have to test with 10 or 15 different devices each time"*. Also, 80 percent of survey respondents agree with the combinatorial explosion as a testing challenge for IoT developers, and P_8 mentioned it as the most severe testing challenge.

Testing and debugging edge-cases. Covering large-scale scenarios (e.g. too many devices) and exceptional cases (e.g. temperatures below zero) add to the test coverage obstacles. This challenge is mentioned by four interviewees, three survey participants, and observed in several GitHub discussions (DEVICE-OS/1926, TEMPERATURE-MACHINE/13). As one example, P_4 said *"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."* Additionally, this challenge is the most experienced testing challenge (83% of respondents).

Immature testing culture. Figure 5 shows an over-reliant on IoT developers for testing as 64% of participants mentioned developers are the main testers in their IoT project.

As P_6, a developer of a popular IoT project with near 7K stars, stated *"We do not have a QA team. it's up to developers to do testing, either manually or writing automated tests."* Often, software developers do not have the skills to test the hardware side. P_9, a software developer of an IoT platform with 1.5K stars, told that the bottle-neck of their IoT platform is testing the hardware side since they do not have sufficient knowledge for tools and practices of hardware testing.

As Figure 5 shows, IoT testing highly depends on manual tasks as only 5% of participants reported testing completely automatic. Also, during interviews, four interviewees mentioned manual approaches for IoT testing. According to the survey respondents, the most adopted IoT testing approach is hybrid strategies. An example of such an approach is described by one IoT developer *"Services which don't interact with devices directly are tested automatically, but checking the entire platform with devices requires manual testing."*

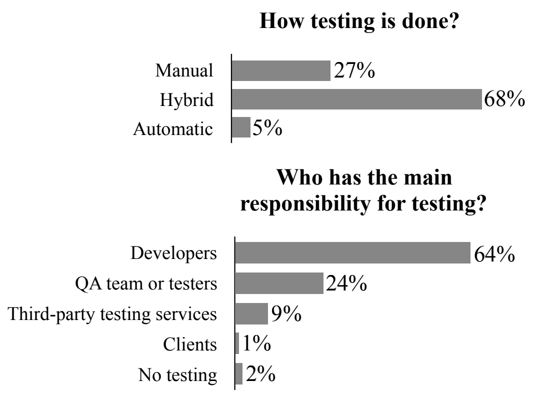


Figure 5: Survey responses about testing in IoT projects.

B. Heterogeneity

Device and protocol fragmentation. Some of the IoT developers reported developing separately for each device or protocol in order to fulfill interoperability ($P_{2\text{-}5}$, P_8). For instance, P_3 stated he has to develop a distinct adapter for talking with each particular device. He mentioned *"There is no guarantee that something that works with brand A also works with brand B."* On the other hand, P_6 noted that their platform is restricted to certain protocols instead of devices. Other developers mentioned fragmentation challenges by pointing out *fragmentation on the same platform* and *time to implement new technologies.* In addition, the majority of the interviewees (seven out of nine), 11 survey comments, and near half of the survey respondents find integration with a new IoT device or communication protocol challenging.

Third-party breaking changes. Third-party changes challenge is mentioned by all interviewees (23 tags) and is agreed by 63% of survey participants and also there are several comments in the survey about it (eight tags). Three interviewees stated that third-parties make breaking changes without prior notice. Also, $P_{5,8}$ mentioned examples where the third-party system stopped supporting a device or a service which caused breakage in their IoT system. Four interviewees ($P_{2,4,5,8}$) explicitly mentioned that it's hard to keep pace with all the rapid changes from various third-parties such as device manufacturers.

Diversity of technologies, backgrounds, and requirements. Challenges posed by the fundamental diversity of IoT technologies are the most repeated challenges among both interviews (30 tags) and survey comments (25 tags). Also, it is agreed by 60% of survey respondents. Several participants mentioned that IoT development requires diverse development skills such as hardware programming and knowledge in dealing with network protocols.

Commonly, developers do not go through this learning curve: *"developers tend to use protocols which they are famil-*

iar with but sometimes better solutions exist and developers do not know/use them."

$P_{2,3,7,8}$ and several survey comments mentioned that it is hard to understand low-quality documentation of certain device manufacturers and interpret complex response payloads from particular devices. $P_{2,3}$ and two survey comments also mentioned that user requirements, as well as users' backgrounds and skills, can be very disparate and it's challenging to develop a generalized IoT system that can support all possible use cases. For instance, P_2 mentioned that they had to include more pins on their hardware and add support for obscured sensors to cover all user requirements. Other challenges are *large search-space for selecting compatible devices or libraries* ($P_{5,7,8}$), and *dealing with diverse regulations and standards* ($P_{5,8}$, and three survey comments).

C. Other challenges

Security. As the survey results suggest, more than half of the IoT developers are not confident about the security of the third-party components, such as operating systems and libraries, used in their IoT system. Also, from a total of 14 participants who mentioned security-related challenges, six of them posed it as the most important challenge. Moreover, 66% of IoT developers find security a complicated task. Our interview participants mentioned security issues rooted in the device firmware ($P_{1,3,4}$), network protocols ($P_{7,8}$), and automation rules (P_6). One of the main challenges, also mentioned by P_7, is generating and storing access tokens within IoT devices that have processing and storage limitations. Similarly, near 60% of IoT developers think that device constraints make security tasks challenging. Another emerged theme from our data is related to the challenge of end-to-end security, from the IoT device to the cloud. Some ($P_{8,9}$) believe that the security of the local communication between the device and IoT gateway is usually underestimated while it can be highly insecure. As P_9 argued, *"to make the development of the IoT system faster, developers don't consider the security of the local network"*. Other challenges mentioned are *the complexity of the certification process*, *supporting different use cases while following security protocols*, and *existence of various attack surfaces*.

Releasing updates for IoT devices. Half of the interviewees believe that releasing software upgrades or security patches for already shipped devices ($P_{5,8}$) is inevitably challenging. Six IoT developers in the survey made comments such as *"getting critical updates installed on already sold devices"* or *"firmware updates in large deployments"* regarding update challenges.

Programming for constrained devices. 63% of the participants agreed that device constraints make IoT development harder. Most IoT developers struggle to design and implement software in a way to consume less processing power and energy. Device limitations in different layers have also been mentioned by our interviewees ($P_{2,3,6,8}$).

Handling failures. An interesting theme that 62% of our participants agreed with is the challenge of handling failures in IoT systems, in a way to avoid losing data and making the system unavailable. As $P_{4,5}$ and five IoT developers from the survey described, developers have to design the system to be tolerable to failures and data losses. *Handling a backlog of sensor data in gateways or constraint devices in case of disconnections* (P_6, ZWAVE2MQTT/141), and *reducing mean-time-to-repair (MTTR) on already shipped devices* are some of the mentioned reliability challenges.

VI. Discussion

IoT testing solutions are not adopted in practice. Various IoT testing tools and methods [39] [40] have been proposed in the literature, such as device simulators [41] and emulators [42], IoT unit testing frameworks [43], [44], and IoT testbeds [45]. However, none of them seems to be adopted by IoT developers as only 9% of them mentioned using third-party services as their main testing approach. Besides, although IoT test automation frameworks exist [41], IoT testing is still carried out in a manual and ad-hoc manner, as 95% of the IoT developers in our study perform manual testing practices. Also, as it is mentioned by $P_{5,8}$, device simulation does not support simulating all types of devices. One possible future direction is having device simulators and emulators specifically crafted for each IoT device individually to virtualize their characteristics and bypass the need for the presence of the actual hardware device during testing. Also, as the importance of combinatorial testing in the context of IoT has been discussed previously in the literature [46], more focus is needed on combinatorial testing tools that consider the heterogeneous nature of IoT devices and protocols.

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 ($P_{1,2,3,4,7}$). 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 ($P_{2,7}$) or using general-purpose tools like Wireshark ($P_{3,7}$). Existing logging solutions to track devices are believed to be inefficient as their limitations were discussed by several IoT developers. One IoT developer best mentioned it as *"even if some devices provide log libraries and tools, they should be manually aggregated or traced from each component separately to track an issue."*

Fragmented and ever-changing ecosystem of IoT. One of the most serious challenges of IoT development nowadays is the rapid obsolescence of hardware devices. As several IoT experts and blog posts [47], [48] describe it, the pace that IoT devices get obscured and stop being supported by the providers is increasing. New updates for IoT devices often make the older devices unusable while they also break IoT developers' implementations. Within this ever-changing ecosystem of IoT, developers have to struggle with maintaining their device-specific or protocol-specific code. IoT developers, not only have to afford all versions of devices to keep up with these changes but also they have to allocate much of their development effort into migrating from one version or ecosystem to the other. As this issue targets both IoT consumers and developers, in 2019, some countries put regulations on the minimum time that IoT providers can release updates after the device is bought [49]. Furthermore, some solutions such as contract-based testing were suggested by interviewees (P_5), to ensure continuous compatibility with third-party systems. However, none of these methods can be a long-term and universal solution as they are still dependent on the contracts and regulations in place.

A. Threats to Validity

Internal validity. One internal threat to the validity of our study, similar to most qualitative studies, is researchers' bias in coding qualitative data. We mitigated this risk by having all authors of the paper involved in the tagging process and discuss any discrepancies in tags for all pieces of qualitative data from various sources (bug reports, interview transcripts, survey comments). For interviews specifically, we eliminated any personal preference on our results by triangulation; each piece of meaningful data from interview transcripts was tagged by one researcher who conducted the interview and one who was not present in the interview session.

External validity. One external threat to the validity of our study is the generalization of studied IoT repositories. We minimized this issue by studying a large number of repositories (91 repositories) selected from all layers of IoT systems. Another risk to the validation of our study is the interview and survey participants not being representative of all IoT developers. However, we minimized this risk by recruiting interview and survey participants with different IoT-related field of expertise, years of experience, companies, and domains. In addition, our survey is filled out by 194 IoT developers with a diverse distribution of skills and experiences.

All our study material, including the bug dataset and interview and survey questions, is available online [19].

VII. Related Work

Bugs and challenges of IoT systems. Although a few previous studies have acknowledged some categories of bugs in IoT systems [5], [8], [9], no study is concerned about categorizing all types of real bugs in IoT systems using a systematic approach. In a recent 2020 study [4], certain peculiarities of open-source IoT repositories were analyzed via examining how developers contribute to IoT repositories. However, this study does not consider bugs and experiences of IoT developers to reach conclusions about the characteristics of IoT development.

A growing body of literature has investigated issues and design flaws that cause safety and security violations in IoT systems as well as security challenges in IoT [50], [51]. More specifically, in smart home ecosystem, security bugs related to the device firmware [52]–[54], communication protocols [55]–[57], smart apps, and safety of their interactions [37], [58], [59], as well as interactions of different components of IoT systems [17] have been studied. There exist taxonomies for describing characteristics of IoT systems with respect to security and privacy concerns [60] [61]. However, these papers do not present their taxonomy construction process, and they are focused on a different goal, namely security requirements and attacks.

Also several studies have investigated challenges of testing IoT systems [39], [46], [62], [63]. Various solutions for IoT testing have been proposed based on e.g., model-based testing [62], IoT mutation operators and test event generators [64], [65], and testing tools [39]. Also, tools and methodologies have been proposed to aid IoT developers in developing of IoT systems [66]–[68].

The challenges of developing IoT systems have been discussed from different perspectives [5], [7], [13]. A previous study investigated the challenges of novice IoT developers to see what development tasks are more challenging for them [6] and developed a tool to help the novice developers [68]. However, no study has tried to study IoT developers' challenges systematically by interviewing and surveying IoT practitioners in the field.

Bug mining and developers' challenges. Although mining IoT repositories has not received any attention in the literature, a growing number of studies have employed mining of software repositories or issue trackers to characterize bugs in Machine Learning systems [10], [69]–[71]. Some prior researches have followed this approach to identify bug categories in Blockchain systems [11], Big Data computing platforms [72], web applications [73] and service compositions [74]. In addition, developers' challenges have been investigated in different contexts such as mobile app development [12], and Blockchain development [75].

VIII. Conclusions

In this paper, we proposed the first bug taxonomy for IoT systems. We also presented a series of categories of challenges in these systems with a qualitative study. Our findings can help both researchers and practitioners in understanding real-world pain-points of IoT development in the wild and designing new techniques and tools. Our findings shed light on the most frequent and severe IoT bugs, their correlations, and their root causes, and thereby allowing these faults to be avoided or detected early in the development of IoT systems.

References

[1] R. Minerva, A. Biru, and D. Rotondi, “Towards a definition of the internet of things (IoT),” *IEEE Internet Initiative*, vol. 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*, pp. 1–29, 2017.

[3] F. Schwandt, "Internet of things (IoT) connected devices installed base worldwide from 2015 to 2025 (in billions)," *Statista*, 2016.

[4] F. Corno, L. De Russis, and J. P. Sáenz, "How is open source software development different in popular IoT projects?" *IEEE Access*, vol. 8, pp. 28 337–28 348, 2020.

[5] T. 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," in *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems*, 2011, pp. 232–245.

[6] F. Corno, L. De Russis, and J. P. Sáenz, "On the challenges novice programmers experience in developing IoT systems: A survey," *Journal of Systems and Software*, vol. 157, p. 110389, 2019.

[7] B. L. R. Stojkoska and 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, Z. Zhen, H. Yu, and J. Xu, "Application of fault tree analysis and fuzzy neural networks to fault diagnosis in the internet of things (IoT) for aquaculture," *Sensors*, vol. 17, no. 1, p. 153, 2017.

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

[10] G. Jahangirova, N. Humbatova, G. Bavota, V. Riccio, A. Stocco, and P. Tonella, "Taxonomy of real faults in deep learning systems," *arXiv preprint arXiv:1910.11015*, 2019.

[11] Z. Wan, D. Lo, X. Xia, and L. Cai, "Bug characteristics in blockchain systems: a large-scale empirical study," in *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*. IEEE, 2017, pp. 413–424.

[12] M. E. Joorabchi, A. Mesbah, and P. Kruchten, "Real challenges in mobile app development," in *2013 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*. IEEE, 2013, pp. 15–24.

[13] A. Čolaković and M. Hadžialić, "Internet of things (IoT): A review of enabling technologies, challenges, and open research issues," *Computer Networks*, vol. 144, pp. 17–39, 2018.

[14] E. I. W. Group *et al.*, "The three software stacks required for IoT architectures," 2016.

[15] F. Javed, M. K. Afzal, M. Sharif, and B.-S. Kim, "Internet of things (IoT) operating systems support, networking technologies, applications, and challenges: A comparative review," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 3, pp. 2062–2100, 2018.

[16] H. Tschofenig, J. Arkko, and D. McPherson, "Architectural considerations in smart object networking, internet engineering task force, rfc-7452," *Internet Engineering Task Force, Fremont, CA, USA*, 2014.

[17] W. Zhou, Y. Jia, Y. Yao, L. Zhu, L. Guan, Y. Mao, P. Liu, and Y. Zhang, "Discovering and understanding the security hazards in the interactions between IoT devices, mobile apps, and clouds on smart home platforms," in *28th USENIX Security Symposium (USENIX Security)*, 2019, pp. 1133–1150.

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

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

[20] *Classifying your repository with topics*, https://help.github.com/en/github/administering-a-repository/classifying-your-repository-with-topics.

[21] H. Borges and M. T. Valente, "What's in a github star? understanding repository starring practices in a social coding platform," *Journal of Systems and Software*, vol. 146, pp. 112–129, 2018.

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

[23] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE transactions on dependable and secure computing*, vol. 1, no. 1, pp. 11–33, 2004.

[24] O. Serrat, "The five whys technique," in *Knowledge solutions*. Springer, 2017, pp. 307–310.

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

[26] P. I. Fusch and L. R. Ness, "Are we there yet? data saturation in qualitative research," *The qualitative report*, vol. 20, no. 9, p. 1408, 2015.

[27] M. D. C. Tongco, "Purposive sampling as a tool for informant selection," *Ethnobotany Research and applications*, vol. 5, pp. 147–158, 2007.

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

[29] G. Guest, A. Bunce, and L. Johnson, "How many interviews are enough? an experiment with data saturation and variability," *Field methods*, vol. 18, no. 1, pp. 59–82, 2006.

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

[31] M. Aniche, C. Treude, I. Steinmacher, I. Wiese, G. Pinto, M.-A. Storey, and M. A. Gerosa, "How modern news aggregators help development communities shape and share knowledge," in *2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE)*. IEEE, 2018, pp. 499–510.

[32] J. Henrich, S. J. Heine, and A. Norenzayan, "The weirdest people in the world?" *Behavioral and brain sciences*, vol. 33, no. 2-3, pp. 61–83, 2010.

[33] G. Coleman and R. O'Connor, "Using grounded theory to understand software process improvement: A study of irish software product companies," *Information and Software Technology*, vol. 49, no. 6, pp. 654–667, 2007.

[34] M. Usman, R. Britto, J. Börstler, and E. Mendes, "Taxonomies in software engineering: A systematic mapping study and a revised taxonomy development method," *Information and Software Technology*, vol. 85, pp. 43–59, 2017.

[35] B. H. Kwasnik, "The role of classification in knowledge representation and discovery," *Graduate School of Library and Information Science. University of Illinois . . .*, 1999.

[36] vyshwanara, "Lack of hardware clock in raspberry pi - possible time lag issues," 2018. [Online]. Available: https://blog.pisignage.com/lack-of-hardware-clock-in-raspberry-pi-scheduling-issues/

[37] M. Alhanahnah, C. Stevens, and H. Bagheri, "Scalable analysis of interaction threats in IoT systems," in *Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis*, 2020, pp. 272–285.

[38] M. Kamber, J. Pei *et al.*, *Data mining: Concepts and techniques*. Morgan Kaufmann Publishers San Francisco, 2001, vol. 2.

[39] J. P. Dias, F. Couto, A. C. 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)*. IEEE, 2018, pp. 104–109.

[40] P. M. Pontes, B. Lima, and J. P. Faria, "Test patterns for IoT," in *Proceedings of the 9th ACM SIGSOFT International Workshop on Automating TEST Case Design, Selection, and Evaluation*, 2018, pp. 63–66.

[41] IOTIFITY, "Advanced IoT system simulation engine and test automation for enterprise IoT apps." [Online]. Available: https://iotify.io/

[42] V. Looga, Z. Ou, Y. Deng, and A. Ylä-Jääski, "Mammoth: A massive-scale emulation platform for internet of things," in *2012 IEEE 2nd International Conference on Cloud Computing and Intelligence Systems*, vol. 3. IEEE, 2012, pp. 1235–1239.

[43] M. Murdoch, "Arduinounit," 2013. [Online]. Available: https://github.com/mmurdoch/arduinounit

[44] I. Kravets, "Platformio: An open source ecosystem for IoT development," *PlatformIO.[En ligne]. Disponible sur: https://platformio.org.[Consulté le: 25-sept-2019]*, 2018.

[45] C. Adjih, E. Baccelli, E. Fleury, G. Harter, N. Mitton, T. Noel, R. Pissard-Gibollet, F. Saint-Marcel, G. Schreiner, J. Vandaele *et al.*, "Fit IoT-lab: A large scale open experimental IoT testbed," in *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*. IEEE, 2015, pp. 459–464.

[46] J. Voas, R. Kuhn, and P. Laplante, "Testing IoT systems," in *2018 IEEE Symposium on Service-Oriented System Engineering (SOSE)*. IEEE, 2018, pp. 48–52.

[47] V. Song, "The never ending death of smart home gadgets," Mar 2020. [Online]. Available: https://gizmodo.com/the-never-ending-death-of-smart-home-gadgets-1842456125

[48] P. Group, "Why can't my dishwasher program my tv? the problems with smart homes," February 2020. [Online]. Available: https://www.iotforall.com/smart-home-problems/

[49] C. Towers-Clark, "Uk to introduce new law for IoT device security," May 2019. [Online]. Available: https://www.forbes.com/sites/charlestowersclark/2019/05/02/uk-to-introduce-new-law-for-iot-device-security/#72f4d763579d

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

[51] Y. Xiao, Y. Jia, C. Liu, X. Cheng, J. Yu, and W. Lv, "Edge computing security: State of the art and challenges," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1608–1631, 2019.

[52] G. Hernandez, O. Arias, D. Buentello, and Y. Jin, "Smart nest thermostat: A smart spy in your home," *Black Hat USA*, no. 2015, 2014.

[53] Z. Ling, J. Luo, Y. Xu, C. Gao, K. Wu, and X. 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. Notra, M. Siddiqi, H. H. Gharakheili, V. Sivaraman, and R. Boreli, "An experimental study of security and privacy risks with emerging household appliances," in *2014 IEEE conference on communications and network security*. IEEE, 2014, pp. 79–84.

[55] E. Ronen, A. Shamir, A.-O. Weingarten, and C. O'Flynn, "IoT goes nuclear: Creating a zigbee chain reaction," in *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017, pp. 195–212.

[56] R. Goyal, N. Dragoni, and A. Spognardi, "Mind the tracker you wear: a security analysis of wearable health trackers," in *Proceedings of the 31st Annual ACM Symposium on Applied Computing*, 2016, pp. 131–136.

[57] B. Fouladi and S. Ghanoun, "Honey, i'm home!!, hacking zwave home automation systems," *Black Hat USA*, 2013.

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

[59] Z. B. Celik, P. McDaniel, and G. Tan, "Soteria: Automated IoT safety and security analysis," in *Annual Technical Conference (USENIX ATC)*, 2018, pp. 147–158.

[60] I. Alqassem and D. Svetinovic, "A taxonomy of security and privacy requirements for the internet of things (IoT)," in *2014 IEEE International Conference on Industrial Engineering and Engineering Management*. IEEE, 2014, pp. 1244–1248.

[61] K. Chen, S. Zhang, Z. Li, Y. Zhang, Q. Deng, S. Ray, and Y. Jin, "Internet-of-things security and vulnerabilities: Taxonomy, challenges, and practice," *Journal of Hardware and Systems Security*, vol. 2, no. 2, pp. 97–110, 2018.

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

[63] P. Rosenkranz, M. Wählisch, E. Baccelli, and L. Ortmann, "A distributed test system architecture for open-source IoT software," in *Proceedings of the 2015 Workshop on IoT challenges in Mobile and Industrial Systems*, 2015, pp. 43–48.

[64] L. Gutiérrez-Madroñal, A. García-Domínguez, and I. Medina-Bulo, "Evolutionary mutation testing for IoT with recorded and generated events," *Software: Practice and Experience*, vol. 49, no. 4, pp. 640–672, 2019.

[65] L. Gutiérrez-Madroñal, I. Medina-Bulo, and J. J. Domínguez-Jiménez, "IoT–teg: Test event generator system," *Journal of Systems and Software*, vol. 137, pp. 784–803, 2018.

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

[67] A. Krishna, M. Le Pallec, R. Mateescu, L. Noirie, and G. Salaün, "IoT composer: Composition and deployment of IoT applications," in *2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*. IEEE, 2019, pp. 19–22.

[68] F. Corno, L. De Russis, and J. 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. Zhang, W. Xiao, H. Zhang, Y. Liu, H. Lin, and M. Yang, "An empirical study on program failures of deep learning jobs," in *2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*. IEEE, 2020, pp. 1159–1170.

[70] M. J. Islam, G. Nguyen, R. Pan, and H. Rajan, "A comprehensive study on deep learning bug characteristics," in *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2019, pp. 510–520.

[71] Y. Zhang, Y. Chen, S.-C. Cheung, Y. Xiong, and L. Zhang, "An empirical study on tensorflow program bugs," in *Proceedings of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis*, 2018, pp. 129–140.

[72] H. Zhou, J.-G. Lou, H. Zhang, H. Lin, H. Lin, and T. Qin, "An empirical study on quality issues of production big data platform," in *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, vol. 2. IEEE, 2015, pp. 17–26.

[73] F. S. Ocariza, K. Bajaj, K. Pattabiraman, and A. Mesbah, "A study of causes and consequences of client-side javascript bugs," *IEEE Transactions on Software Engineering*, vol. 43, no. 2, pp. 128–144, 2016.

[74] K. M. Chan, J. Bishop, J. Steyn, L. Baresi, and S. Guinea, "A fault taxonomy for web service composition," in *International Conference on Service-Oriented Computing*. Springer, 2007, pp. 363–375.

[75] W. Zou, D. Lo, P. S. Kochhar, X.-B. D. Le, X. Xia, Y. Feng, Z. Chen, and B. Xu, "Smart contract development: Challenges and opportunities," *IEEE Transactions on Software Engineering*, 2019.