# Automating Cross-Boundary Bug Reporting: A Case Study on Scaling Dogfooding and Actionable Feedback with ODM Partners

Eberth Felipe Castro da Cruz
Instituto de Pesquisas Eldorado
Manaus, Amazonas, Brazil
eberthc@eldorado.org.br

José Alberto Gurgel Cardoso Neto
Motorola Mobility
Manaus, Amazonas, Brazil
jgurgeln@motorola.com

## Abstract

In the competitive smartphone industry, Dogfooding is a cornerstone for validating user experience and system stability before public release. However, when vendors choose to partner with Original Design Manufacturers (ODMs) to expand their portfolios (smartphones, IoT, and Wearable), traditional internal diagnostic tools face significant scalability and security challenges when crossing corporate boundaries. This paper presents a case study of a global smartphone vendor that transitioned from a fragmented, manual spreadsheet-based feedback loop to a re-engineered internal bug-reporting and diagnostic ecosystem. We describe the technical and process-level adaptations required to enable automated log collection and real-time feedback loops outside the company's native domain. Our findings demonstrate that this automation enabled scaling the user testing base by over 500% and increasing the volume of actionable feedback by 14 times. Most notably, the system resulted in a 15-fold increase in effective code changes and bug fixes, significantly improving defect detection in diverse hardware-software integrations. This approach enables global vendors to bridge the quality gap between internal engineering standards and external manufacturing partners, significantly accelerating time-to-market for complex, distributed product portfolios.

## Keywords

Dogfooding, Software Quality Assurance, Experience Report, Mobile Software Engineering, ODM, Internet of Things (IoT)

## 1 Introduction

Smartphones are an essential part of modern life, serving as cameras, work tools, entertainment devices, and educational platforms. Android is part of the mobile ecosystem, having the trust of the users by the time that it's been commercialized, and because it's powered by Google as an open source software in partnership with OEM (Original Equipment Manufacturer) around the globe. In addition to massive use, companies that manufacture tablets and smartphones have a large market share to maintain and conquer, since Android is the most popular platform for mobile devices [9].

After manufacturing, implementing, and testing mobile devices in a development environment, one of the biggest challenges for companies that produce and sell worldwide is ensuring their quality in real-world conditions. Afterward, they will be used for different goals, and different network conditions as each country has divergent infrastructure, technology access [1, 2, 8, 11], and plans to continue evolving [8]. For instance, Latin America has a constant demand for extending the reach of mobile networks [3].

A key driver of this shift is the growing collaboration with Original Design Manufacturers (ODMs), who are responsible for designing and producing many of the hardware components and devices that are later branded and sold by larger companies. ODMs often operate outside the internal enterprise environment, yet require access to diagnostic tools, feedback systems, and testing workflows to ensure product quality and compatibility. This dynamic environment introduces the need for secure, scalable, and flexible integration between internal systems and external stakeholders.

One strategy to overcome the challenge of product validation in a real scenario using a diverse population is called Dogfooding (also known within the company as User Trials), which refers to the concept that companies provide their own products to employees to use them as conventional customers [5, 10]. However, as organizations expand their product ecosystems to include a broader range of connected devices, such as smart watches, earbuds, and other Internet of Things (IoT) products, the need to extend these internal capabilities to external users introduces significant architectural and operational challenges.

The introduction of external device support required a fundamental change in how these systems are operated. It was no longer sufficient to rely on static rules, internal-only access, or tightly coupled authentication mechanisms. Instead, the architecture had to evolve to support dynamic decision making, secure external access, and compatibility with devices that lacked privileged system access. This transformation involved updating multiple interconnected systems, introducing new authentication flows, and implementing secure data sharing mechanisms, all while preserving the integrity and functionality of existing internal workflows.

This paper details the technical and compliance strategies for adopting Dogfooding with ODM partners. We share lessons on decoupled architectures, secure data handling, and interoperability, concluding with proposed improvements for system scalability.

## 2 Dogfooding Ecosystem Overview

During product development, validating user acceptance in real-world scenarios is critical. To achieve this, companies often recruit employees to participate in the validation and homologation of pre-release prototypes, a practice commonly known as User Trials or Dogfooding [5, 10]. The core concept is to democratize participation across diverse departments (e.g., software development, engineering, finance, sales, and marketing). This ensures a heterogeneous and statistically significant user base capable of simulating authentic end-user behaviors, including varied geolocations, heavy application usage, gaming performance, media consumption, battery drain, and standard telephony.

Dogfooding aims to preemptively identify and resolve critical defects that impact perceived product quality [5, 10]. Because qualitative feedback alone is insufficient for debugging, collecting system logs is crucial for enabling developers to perform root-cause analysis and resolve issues before public release.

Successfully adopting Dogfooding requires a robust architectural solution that empowers users to seamlessly report issues while enabling engineering teams to manage configurations and collect system logs. To this end, a global smartphone vendor developed a proprietary ecosystem to support its Dogfooding operations, structured as follows.

- **Log Collection Application:** A system-privileged Android application extracts comprehensive device logs (e.g., application crashes, Wi-Fi, Bluetooth, and cellular network states) directly from the trial device. It simultaneously captures user-provided feedback, including media evidence (images/videos), timestamps, and reproduction steps.
- **Secure Web Server:** A centralized backend receives the diagnostic payloads and automatically provisions a task in Jira. For security and intellectual property protection, this server is strictly accessible only by internal employees.
- **Issue Triage:** The generated Jira ticket undergoes a triage phase and is routed to the appropriate engineering domain.
- **Resolution Workflow:** The assigned engineering team retrieves the logs from the web server, performs root-cause analysis, and resolves the issue, either through Android source code modifications or by documenting the expected technical behavior.

Beyond traditional smartphones, the OEM is expanding its portfolio to include connected devices such as IoT and Wearable. Depending on the corporate strategy, the development life cycle for these products follows one of three models: (i) strictly in-house hardware and software development; (ii) a hybrid approach (e.g., internal hardware paired with third-party software, or vice versa); or (iii) fully outsourced hardware and software development.

In the latter scenario, Original Design Manufacturers (ODMs) play a pivotal role. By leveraging the ODM's specialized manufacturing expertise, companies can acquire hardware, apply brand customizations [7], and significantly accelerate time-to-market while mitigating extensive development overhead.

Because ODMs operate as external entities, they are inherently isolated from the OEM's secure internal Dogfooding ecosystem. Nevertheless, these externally manufactured devices must undergo the same rigorous product development and quality assurance phases, including Dogfooding. Ensuring secure diagnostic pipelines across the IoT supply chain requires overcoming significant coordination and firmware integration challenges between OEMs and ODMs [6]. Therefore, this paper explores the technical and procedural integration of ODM partners into an established Dogfooding ecosystem, demonstrating how to effectively validate the reliability of connected devices under real-world usage conditions.

## 3 ODM Dogfooding Process

The fundamental premise of this context is the use of an IoT or Wearable manufactured by an ODM, paired with a smartphone. The interaction between these devices is facilitated by a companion application, available via the Google Play Store, which must be installed on the smartphone to establish communication between the Android system and the IoT peripherals.

The primary focus of this specific User Trial category is the ODM hardware. However, maintaining a stable connection between the peripheral and the smartphone is critical for functional verification. Many IoT devices lack a dedicated Graphical User Interface (GUI) and rely entirely on this pairing for initial configuration and continuous data transmission. Numerous use cases can trigger system failures, necessitating detailed logs for root-cause analysis, particularly concerning Wi-Fi and Bluetooth connectivity issues that must be investigated across both devices.

By adopting global connectivity standards, these IoT devices are expected to be compatible with smartphones from various manufacturers. This cross-vendor compatibility is a highly relevant and feasible scenario when selecting a representative user base for Dogfooding. Consequently, the following conditions were established for conducting Dogfooding in this context:

- Employees from both the primary vendor and the ODM can be recruited for the Dogfooding;
- Smartphones from either the primary vendor or third-party brands may be used (provided they run the Android operating system);
- The companion application must provide a secure mechanism for sharing diagnostic logs from the associated IoT or Wearable device.

Since the internal systems designed for the Dogfooding process were originally restricted to the vendor's employee credentials and system-privileged applications, several architectural adaptations were implemented to securely accommodate external devices and users.

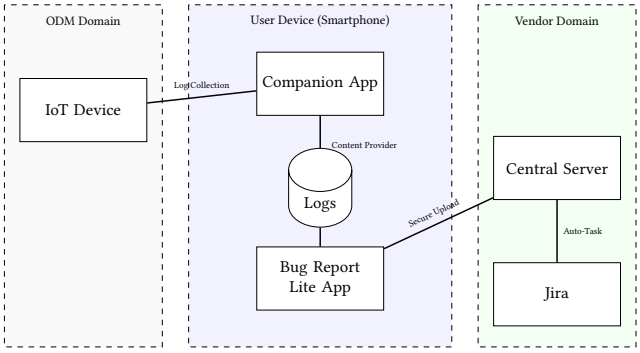


**Figure 1: Cross-boundary interaction model between ODM hardware and Vendor infrastructure.**

- **External Authentication:** Implementation of an IT-approved registration and authentication method via a dedicated SDK, allowing the creation of external profiles within a domain controlled by the vendor.
- **Lite Application Development:** Development of a "Lite" version of the Android reporting application. As illustrated in Figure 1, this application interfaces with the Companion App to extract IoT logs and enables feedback submission on

third-party smartphones, despite possessing fewer system-level capabilities than the internal version.
- **Profile Validation:** External users authenticate their profiles directly with the platform server responsible for receiving and storing problem reports.
- **Device Whitelisting:** Mandatory pre-registration of the external device's serial number. Without this registration, the system prevents report submission even for users with valid credentials. This additional security layer mitigates malicious usage and protects against unauthorized access.
- **Isolated Infrastructure:** The centralized server remains inaccessible to any party outside the vendor's internal network.
- **Automated Triage:** Upon receiving a report, the server automatically opens a task in the vendor's internal Jira, which is then triaged by specialized teams.
- **Data Segregation:** If the issue is related to the vendor's smartphone, it is assigned to the internal team. If the defect lies within the ODM device, a task is created in an external Jira accessible to the partner. In this case, only the relevant logs for the specific device are attached, ensuring that only strictly necessary diagnostic data is shared.

To standardize communication across heterogeneous components developed by different ODMs, the implementation of Android Content Providers was established. An Android Content Provider is a fundamental component responsible for managing access to a centralized data repository. It provides a standardized interface for data exchange and inter-process communication (IPC), allowing applications to securely share data with other authorized applications [4].

## 4 Results and Discussion

Prior to the implementation of the proposed process and the expansion into IoT and Wearable devices, the vendor and its ODMs already had an established partnership for smartphone development. However, the legacy Dogfooding procedure was costly, complex, and entirely manual. This legacy process operated as follows.

- An online spreadsheet was created for each specific product under test.
- Participants were required to manually enter their personal information and device details. Furthermore, feedback was provided without any system logs; at best, users provided manual reproduction steps for the encountered issue.
- ODM developers were responsible for reviewing this spreadsheet daily and interacting with users asynchronously through spreadsheet comments, as there was no direct communication channel (e.g., chat). If a reported problem persisted, a task was manually created in the external Jira. Once resolved, the fix details and the target software version were documented back in the spreadsheet so that the user could verify the solution.

By applying the new process and adapting the internal diagnostic systems, the Dogfooding workflow was automated, streamlining the operational steps and making them significantly more effective and practical. Consequently, overall productivity increased drastically.

Table 1 presents a comparative analysis of metrics from recent projects utilizing the new automated ecosystem compared to historical data from legacy manual projects. In this study, actionable feedback is defined as reports containing sufficient technical metadata (logs and system state) to allow engineers to begin root-cause analysis without further user interaction. Verified bugs are reports confirmed during triage as valid defects, and effective code changes refer to specific commits in the version control system that directly resolved a Jira task originated from the dogfooding process.

**Table 1: Comparison of Productivity Metrics: Automated vs. Legacy Manual Process**

| Process Type | Project | Users | Jira Tasks | Code Changes |
|---|---|---|---|---|
| Automated | 1 - A | 187 | 1957 | 1198 |
| | 2 - B | 193 | 1987 | 584 |
| | 3 - C | 57 | 1136 | 378 |
| Legacy (Manual) | 4 - D | 18 | 122 | 25 |
| | 5 - E | 19 | 79 | 3 |
| | 6 - F | 33 | 164 | 115 |

Automation of the diagnostic process allowed for the inclusion and management of a significantly larger user population. In the automated model, the average number of participants scaled from roughly 23 to 145 users per project. Although individual project scopes vary, the projects compared in Table 1 represent Wearable devices sharing a similar software baseline. To ensure a fair comparison, we selected projects with equivalent development milestones, covering the entire pre-release validation phase and involving multiple software iterations. These projects were developed through hybrid and fully outsourced models, all necessitating a companion application for proper device interaction and diagnostic data collection.

Automated log integration into Jira tasks facilitated faster analysis and resolution, eliminating manual spreadsheet bottlenecks, and significantly increasing the volume of actionable feedback. Most importantly, the quality and actionability of the feedback improved. While the manual process resulted in an average of only 47 code changes per project, the automated log-enabled process facilitated an average of 720 effective code changes and bug fixes. This shows that providing developers with automated logs and direct tool integration leads to a higher rate of actual defect resolution.

## 5 Lessons Learned

Developing and maintaining large-scale Dogfooding ecosystems is complex, and expanding the operation to include ODMs has added another layer of difficulty. Nevertheless, the resulting productivity gains justified the technical effort. Therefore, we outline the lessons learned regarding the Dogfooding ecosystem.

The development team initially planned to create an Android AAR (Android Archive) library to expose the APIs that communicate with the server. This library would have an interface that allows users to fill in the information required to report a bug. However, other applications are part of the Dogfooding ecosystem, and they rely on the report app to get information stored in the database, take actions via broadcast receivers, and create bug reports directly, not requiring the user to fill in anything. Despite the

fact that Android AARs support their own databases and broadcast receivers, we couldn't overcome the fact that a centralized source of data orchestrated interactions better between Dogfooding and Companion apps.

Architecting the solution with a single app that communicates with others and controls the business rules was the best approach to ensure that the internal tools remained functional, even when used by external users, and to reuse almost all the codebase of the more robust internal app. Removing the capability of managing deep Android log systems resulted in the creation of a lighter version of the same application, which significantly accelerated the integration with ODMs.

Providing a regular app to get crucial information from IoT devices across different mobile manufacturers was very challenging: since a system app cannot be shared due to its reliance on the internal vendor build, the number of permissions to get necessary information is strictly limited. Developing a safe mechanism that works for any Android model was crucial, because the main functionality of this "Lite" app (figure 1) is to provide a user report with the necessary logs to validate the product.

A critical aspect of crossing corporate boundaries involved navigating stringent data privacy requirements. The vendor's security and legal teams mandated that for every report submitted to an external ODM, users must provide explicit consent. To meet stringent privacy requirements, we implemented a mandatory disclosure dialog for reports shared with external ODMs. This ensures explicit user consent and transparency regarding potential Personally Identifiable Information (PII), balancing the need for deep technical diagnostics with data protection in a distributed environment.

Investing in software documentation and providing a guideline for ODM developers to follow the Content Provider pattern, also detailing the methods that they must have in the Companion App, was a key aspect to achieve great results.

This productivity boost is also critical because Dogfooding cycles are extensive and continuous, encompassing multiple software releases. In the manual process, tracking issues over various releases often resulted in loss of information. The automated ecosystem mitigates this by embedding the exact software build version into every diagnostic payload. This ensures continuous traceability, empowering users to verify fixes across sequential updates, and allowing engineers to reliably monitor defect lifecycles throughout the project.

Validating the user experience and system stability of an ever-expanding portfolio of IoT and Wearable devices is a major challenge for global smartphone vendors. This experience report demonstrated that expanding internal Dogfooding practices to external Original Design Manufacturers (ODMs) is not only feasible but highly effective when supported by the right architectural adaptations.

By transitioning from a fragmented, manual spreadsheet-based process to an automated, log-enabled reporting ecosystem, the company successfully bridged the quality gap between internal engineering and external manufacturing. As demonstrated, this approach enabled scaling the user testing base by over 500% and increased the volume of actionable feedback by 14 times, ultimately resulting in a 15-fold increase in effective bug fixes. The decision to centralize data via a "Lite" application and standardize communication through Android Content Providers proved crucial in overcoming the security and scalability constraints of crossing corporate boundaries.

Unlocking the boundaries of the company and integrating with ODMs seamlessly has opened new possibilities to expand the Dogfooding operation even further. As future work, we plan to evolve the platform's architecture to support data collection and reporting beyond the Android operating system, ensuring comprehensive quality assurance across a truly agnostic device ecosystem.

## Artifact Availability

The study does not have any artifact available.

## Acknowledgements

The R&D project challenges described for this work are part of an agreement established based on the resources of the Informatics Law no. 8.387/91 under the auspices of Suframa.

The authors acknowledge the use of Google Gemini to assist with language translation and grammatical refinement during the preparation of this manuscript. All technical content, data analysis, and conclusions remain the sole responsibility of the authors.

## References

[1] Peter Boyland. 2019. *The state of mobile network experience: Benchmarking mobile on the eve of the 5G revolution.* Retrieved Jun 14, 2024 from https://cdn.opensignal.com/public/data/reports/global/data-2019-05/the_state_of_mobile_experience_may_2019_0.pdf

[2] Peter Boyland. 2020. *The state of mobile network experience: One year into the 5G era.* Retrieved Jun 14, 2024 from https://cdn.opensignal.com/public/data/reports/pdf-only/data-2020-05/state_of_mobile_experience_may_2020_opensignal_3_0.pdf

[3] André Mendes Cavalcante, Maria Valéria Marquezini, Luciano Mendes, and Carlos Sallé Moreno. 2021. 5G for Remote Areas: Challenges, Opportunities and Business Modeling for Brazil. *IEEE Access* 9 (2021), 10829–10843. doi:10.1109/ACCESS.2021.3050742

[4] Android Developers. 2026. *<Content Providers> | Android Developers.* Retrieved May 09, 2026 from https://developer.android.com/guide/topics/providers/content-provider-basics

[5] Warren Harrison. 2006. Eating Your Own Dog Food. *Software, IEEE* 23 (06 2006), 5– 7. doi:10.1109/MS.2006.72

[6] IoT Security Foundation (IoTSF). 2022. *Securing the Internet of Things Supply Chain.* Whitepaper Release 1.0.0. IoT Security Foundation. https://www.iotsecurityfoundation.org/best-practice-guidelines IoTSF-SCIP V1.0.0.

[7] Yaqi Lou, Zhen He, Lipan Feng, Keyuan Cai, and Shuguang He. 2022. Original design manufacturer's warranty strategy when considering retailers' brand power under different power structures. *International Transactions in Operational Research* 29, 2 (2022), 1308–1325. arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/itor.12795 doi:10.1111/itor.12795

[8] Ascensión López-Vargas, Manuel Fuentes, and Marta Vivar. 2020. Challenges and Opportunities of the Internet of Things for Global Development to Achieve the United Nations Sustainable Development Goals. *IEEE Access* 8 (2020), 37202–37213. doi:10.1109/ACCESS.2020.2975472

[9] statcounter 2024. *Statcounter GlobalStats.* Retrieved Jun 12, 2024 from https://gs.statcounter.com/os-market-share/mobile/worldwide

[10] Edgar Tanaka, Edilson Silva, and Gustavo Tordin. 2019. Dogfooding: "Eating our Own Dog Food" in a Large Global Mobile Industry Player. In *2019 ACM/IEEE 14th International Conference on Global Software Engineering (ICGSE).* 62–67. doi:10.1109/ICGSE.2019.00024

[11] Ermias Andargie Walelgne, Alemnew Sheferaw Asrese, Jukka Manner, Vaibhav Bajpai, and Jörg Ott. 2021. Understanding Data Usage Patterns of Geographically Diverse Mobile Users. *IEEE Transactions on Network and Service Management* 18, 3 (2021), 3798–3812. doi:10.1109/TNSM.2020.3037503