Section 8. Engineering sciences in general

https://doi.org/10.29013/ESR-23-3.4-46-49

Bukarev Anton,
National Research University of Electronic Technology, Russia

KEY FEATURES IN THE ANDROID SOFTWARE TESTING PROCESS

Abstract. This article examines the challenges and complexities of testing native Android applications, focusing on their interaction with the operating system (OS) and other apps. It highlights critical error reproduction and emphasizes the importance of a meticulous approach to ensure a seamless user experience. The article also provides valuable insights and recommendations for app developers and testers to address these challenges and create high-quality, user-friendly applications.

Keywords: Android applications, testing challenges, Android testing specifics.

1. Introduction

In recent years, the Android platform has become the predominant operating system in the mobile market, with research from analytical agency Statista in 2022 indicating a 71% market share [1]. As a result, numerous companies are increasingly focusing on this platform to expand their user base. The open-source nature of Android has significantly contributed to its market dominance; however, the variety of devices presents challenges in device support and software compatibility. Manufacturers often refrain from updating older devices to the latest operating system version due to economic considerations, encouraging users to acquire new devices with updated specifications. Software developers, in turn, prioritize application development for leading market devices. These factors, alongside programming errors, can lead to decreased demand for specific services and a decline in user experience. Consequently, comprehensive testing of software released to a wide user base is of paramount importance [2].

2. Features associated with hardware in the testing process

The Android platform, as an open-source and cost-free system, has played a crucial role in garnering its substantial market share. Nevertheless, the market encompasses a diverse array of devices, from flagship to lower-tier models. Furthermore, ongoing advancements in electronics and manufacturing technologies prompt annual updates in smartphone lineups from various manufacturers, including Samsung and Huawei. These factors contribute to considerable market segmentation, despite the existence of a single operating system, thereby presenting additional challenges when testing native Android applications compared to their iOS counterparts [3]. It is imperative to consider the device's hardware characteristics during the testing process, such as:

RAM

Insufficient random-access memory (RAM) can markedly affect an application's speed and performance. Moreover, fully utilized RAM may result in instability and disruptions in the application's functionality. It is vital to consider this aspect during the testing process by examining the application's performance under conditions where memory is allocated to other applications. Under such circumstances, besides application crashes, functional issues might emerge, including errors, absent user interface elements, and the suspension of background services within the application, among other potential complications.

Storage size

This parameter is notably specialized, as a considerable amount of internal memory is predominantly needed for gaming and media applications. Nevertheless, in situations like streaming audio playback under conditions of inadequate internal memory, the playback will understandably stop. If the application's source code does not accommodate such a scenario, a critical error may transpire. As a result, it is recommended to incorporate this scenario into the test plan and conduct testing with memory prepopulated with arbitrary data [4].

Network speed

In instances where the application's source code incorporates a request timeout, the application might not await the arrival of incoming information, and with exceedingly slow connection speeds, interface elements may not load within the designated time. This can result in unforeseeable consequences such as application termination or compromised usability due to the absence of some or all interface elements. The connection type predominantly influences application functionality involving user data loading. Certain applications enable users to disable data downloading or background updates via mobile networks, necessitating verification during the testing process, as it can potentially impact user expenses and, ultimately, user satisfaction with the application's performance.

Display size

Screen size constitutes a critical parameter for any application, influencing the dimensions and placement of user interface elements. Consequently, it is vital to accord particular attention to devices with minimum and maximum screen dimensions. On devices with minimal screen parameters, interface elements may overlap, rendering them inoperable. Conversely, on larger screens and resolutions, elements may be diminutive to the extent that users struggle to interact with them accurately using their fingers. Hence, during the software quality assurance process, these parameters ought to be considered, and dynamic sizing of elements should be implemented to adapt to varying screen sizes and resolutions [5].

Additional sensors

In smartphones, the accelerometer plays a vital role by measuring free-fall acceleration across three axes and determining the device's spatial orientation from these measurements. This sensor enables the transition of the interface between portrait and landscape orientations when the device is rotated. While the accelerometer should be taken into account during testing if the application accommodates both interface orientations, its importance is secondary to the processes involved in altering orientation and rotating the interface, which will be addressed later.

Various sensors, including proximity, ambient light, heart rate, gyroscope, barometer, and thermometer, are integrated into modern smartphones to serve specific application needs. The primary testing of these sensors should occur at the manufacturing facilities where the sensors are produced.

3. Testing and error documentation rely on device management tools

Numerous actions can be recognized as common in the routine usage of a smartphone, which users frequently execute:

App minimization

In instances where the application is designed to operate in the background, minimization often presents a typical scenario for errors to transpire. Upon being minimized, the app may either stop functioning or exhibit improper behavior.

App termination

Though infrequent, errors may arise when an application persists in functioning even after termination, exemplified by the ongoing audio playback in a closed player.

Orientation change

A common error includes the displacement of UI elements or instances where elements extend beyond the boundaries of the display.

In each of the aforementioned scenarios, if an error is identified in a particular area and can be consistently replicated, the error documentation within the bug tracker must encompass the conditions in which the error transpired, along with the conventional steps for reproduction [6].

4. App interaction with OS and others: Critical error reproduction

An examination of the interaction between a test application, the operating system, and other applications can be conducted through the example of an audio player that utilizes the standard audio service of the Android platform.

Interaction between players

Two cases can be identified in the interaction of audio players: when players utilize the same audio service, and when a third-party application employs its own service. In the first scenario, it is essential for the application to inform the operating system that it has initiated audio playback. Consequently, when a second audio player is activated while the first one is playing, the initial player ceases audio playback, pauses, and cedes focus to the second player starting playback. However, if the application fails to convey this information, both players may play audio simultaneously, leading to a prevalent error. In some instances, this error emerges due to a third-party application, in which case notifying its developers of the issue would be appropriate [7].

Default applications

Among standard Android applications utilizing the audio service, the "Phone" app serves as a prime example. The main interaction between the phone and audio player involves shifting focus between a call and audio playback. When the phone application notifies the operating system of an incoming call, the audio-playing application must relinquish focus to the phone for ringtone playback and pause its own audio. Once the phone application communicates that the call has concluded, the audio application regains focus and resumes playback from where the focus was transferred to the phone. Given that the phone application is error-free, any related errors reside solely within the tested application.

Interaction with application notifications

When the tested application incorporates a notification feature, proper configuration of the notification operation is essential. If a notification includes interactive functions, such as managing audio playback without launching the application, scenarios may arise where the interactive controls fail to function or the application does not open in full screen upon clicking the notification.

Interaction with third-party application notifications

In cases where a third-party application notification is received while the tested application is playing audio, the optimal solution involves muting the audio without pausing playback or momentarily transferring focus to the notification, pausing and then resuming playback. Nonetheless, situations may arise where the application relinquishes focus to the notification but fails to regain it, resulting in paused playback, which could be inconvenient for the user.

5. Conclusion

The testing of native Android applications is a laborious and resource-intensive endeavor due to significant market segmentation, necessitating a scrupulous approach to testing. However, by pinpointing specific functionalities where errors arise and employing auxiliary tools for data collection, application development and testing become comparatively manageable, enabling the creation of truly outstanding applications that delight users. Mobile software development does not require build servers for application assembly, as seen in web application development, which reduces the intermediaries between programmers and users while mitigating potential (and frequent) build errors. As a result, code errors transform into situations amenable to timely discussion and resolution.

References:

1. URL: https://www.statista.com/statistics/272698/global-market-share-held-by-mobile-operating-systems-since-2009/ Access on 2023.
2. URL: https://developer.android.com/training/testing/fundamentals/ Access on 2023.
3. Gao J., Bai X., Tsai W. and Uehara T. "Mobile Application Testing" IE.
4. Pavithra L. and Sandhya S. "Survey on Software Testing". Journal of Network Communications and Emerging Technologies (JNCET). – Vol. 9. – Issue 3. March, 2019.
5. Zein S., Salleh N., Grundy J. A systematic mapping study of mobile application testing techniques. Journal of Systems and Software. 2016
6. Mohd. Ehmer Khan & Farmeena Khan. Importance of Software Testing in software Development Life Cycle, International Journal of Computer science Issues, – Vol. 11. – No. 2. March, 2014.
7. Shalini Gautam and Bharti Nagpal. "Descriptive Study of Software Testing & Testing Tools". International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE). – Vol. 4. – Issue 6. June, 2016.