Topics

Efficient Corporate User Smartphone Kitting with Android Kitting Tool

Solution Service Department Takeshi Ishii Shinya Hasegawa

When delivering Android™*1 smartphones to corporate customers, initial settings unique to each company called "kitting" must be performed for each handset. For example, these settings include screen lock settings, Wi-Fi®*2 connection settings, adding and settings of applications and settings to position applications on the home screen. Kitting is performed in advance so that customers can use the Android smartphones in their business as soon as they are delivered, and it is thus a necessary process in corporate business which improves customer satisfaction levels.

The number of kitting work items is on the rise due to the many settings for Android smartphones and the variety of customer usage methods and control items of Mobile Device Management (MDM)*3 are expanding. Depending on the case, it can take several hours to kit one device, or several months to kit all devices to be delivered to a company. The timing of kitting work for each project is unclear, and is not possible to increase the number of staff to cope without planning. Additionally, many delivery cycles are short, which makes shortening work time an urgent matter. Also, accuracy must be improved to prevent settings errors and raise productivity.

To solve these issues, NTT DOCOMO has developed and deployed an Android kitting tool (hereinafter referred to as "AKT") for its in-house PCs to automate kitting work. AKT has been reducing the amount of work, shortening delivery time and improving accuracy.

This article describes the AKT development.

1) The Initial AKT

At first, we designed AKT as a tool to automatically set approximately 50 items, using an Application Programming Interface (API)*4 method and a UI Automator*5 method (**Figure 1**).

(1) API method

In Android smartphones, there are APIs available for applications and their settings, and terminal settings. For example, screen brightness settings and so forth can be changed from other applications etc. via connection to the screen settings API (**Figure 2**). Not necessarily all settings are accessible through the API, but all settings available with the API are performed.

(2) Mechanism with the UI Automator method

UI Automator can acquire on-screen information, and can simulate key events such as tap and scroll using commands. For example,

*1 Android™: A trademark or registered trademark of Google, Inc.
*2 Wi-Fi®: A registered trademark of the Wi-Fi Alliance.
*3 MDM: A method of managing and controlling the functions and applications of smartphones supplied to staff of a company etc.

key event commands can be used to start downloading a particular application, analyze the characters displayed when download finishes, and then execute the subsequent action. UI Automator is used for kitting items not executable through the API (**Figure 3**).

(3) Issues with the initial AKT

Because there was only a limited number of items that could be set with the initial AKT, not all items required for kitting

*4 API: Interfaces specified for communications between different software.

*5 UI Automator: A test tool program provided by Google, Inc.

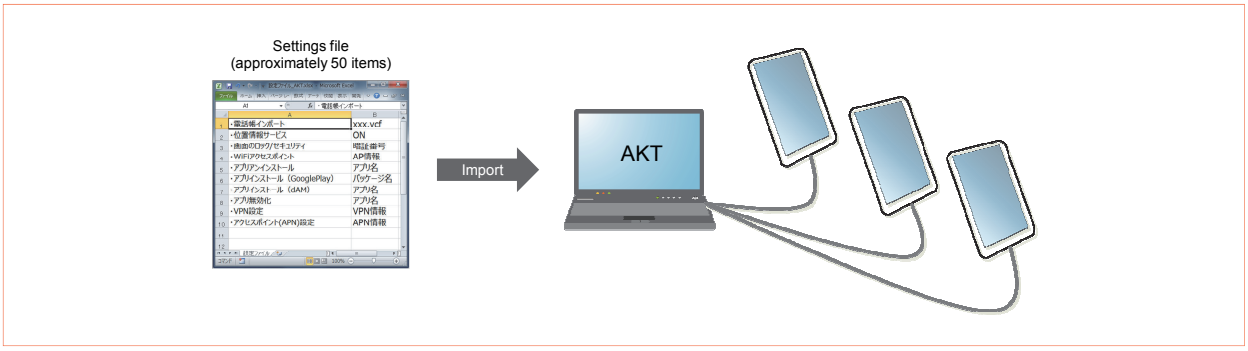

Figure 1 The Initial AKT

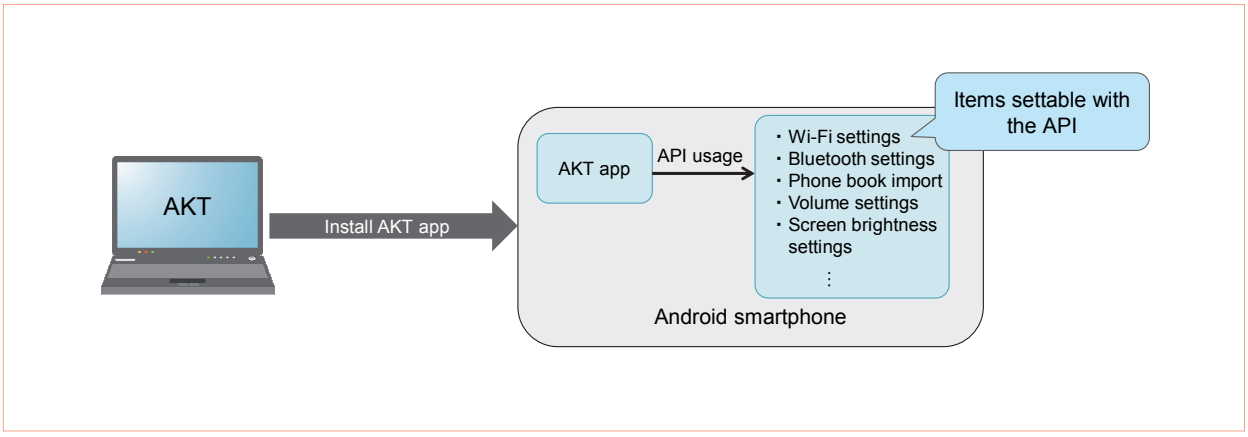

Figure 2 API method

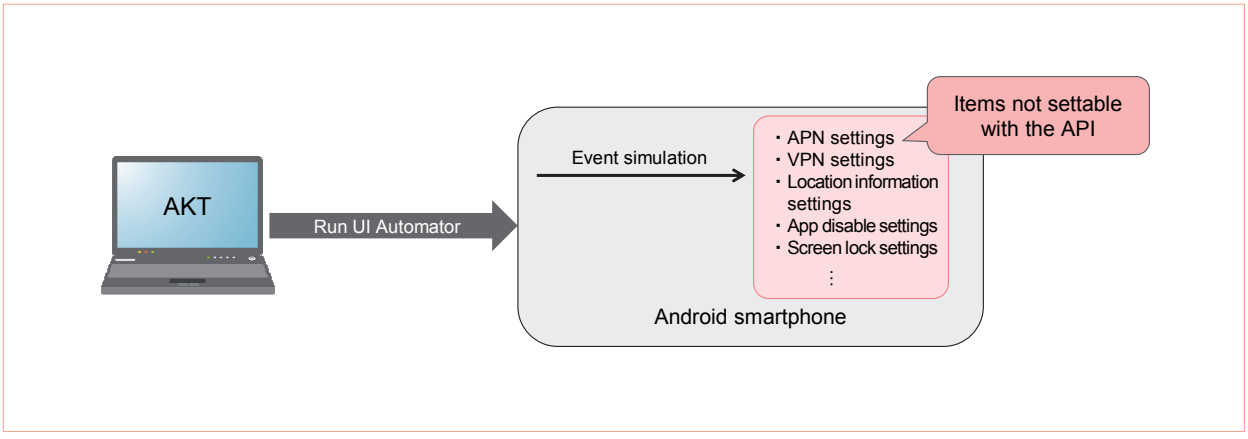

Figure 3 UI Automator method

were covered. Furthermore, because the scope varied depending on the details of kitting, usage of this system required judgments to identify levels of efficiency and effectiveness. Hence, the system was not always used and did not become established in-house.

Also, there was the operational issue of additional development required with AKT so that UI Automator could handle new models, which entailed long development periods and difficulty in reducing development costs.

2) AKT Improvements

We broadened the scope of AKT functions so that it can be used more efficiently with a wider range of projects.

(1) Operation scenario function development

Because the initial AKT only offered limited number of settings items, we developed and added a function for operation scenarios to pseudo-replay Android smartphone operations.

With this function, an Android smartphone is first connected to the AKT and kitted, which is recorded in AKT as a scenario. Next, multiple unkitted Android smartphones are connected to AKT, and the scenario is run to automatically reproduce the same operations as the Android smartphone that was connected first (**Figure 4**).

Firstly, with this function, we assumed use of getevent (acquire touch event)/sendevent (send touch event) operations available with Android Debug Bridge (ADB)*6, but as a result of testing, the replay processing speed was slow, and it was clear that the actions were not the same as they were recorded. This was caused by sendevent processing delays due to internal processing congestion with sendevent processing for each touch event (**Figure 5** (a)). Therefore, to prevent internal processing congestion, we changed to write the touch event acquired with getevent directly in the Android system using Android Native Development Kit (Android NDK)*7, without using sendevent. With this change, we were able to eliminate processing latency with replay and confirm correct operation of the function (Fig. 5 (b)).

The inclusion of the operation scenario function makes it possible to automate all

*6 ADB: A development tool program provided by Google, Inc., that enables Android smartphone control from PC.

*7 AndroidNDK: A development kit provided by Google, Inc. for Android application development in the C/C++ languages. This software has a high processing speed because it is not run via an execution environment for Java® language applications. Oracle and Java are registered trademarks of Oracle Corporation and its subsidiaries and related companies in the United States and other countries.

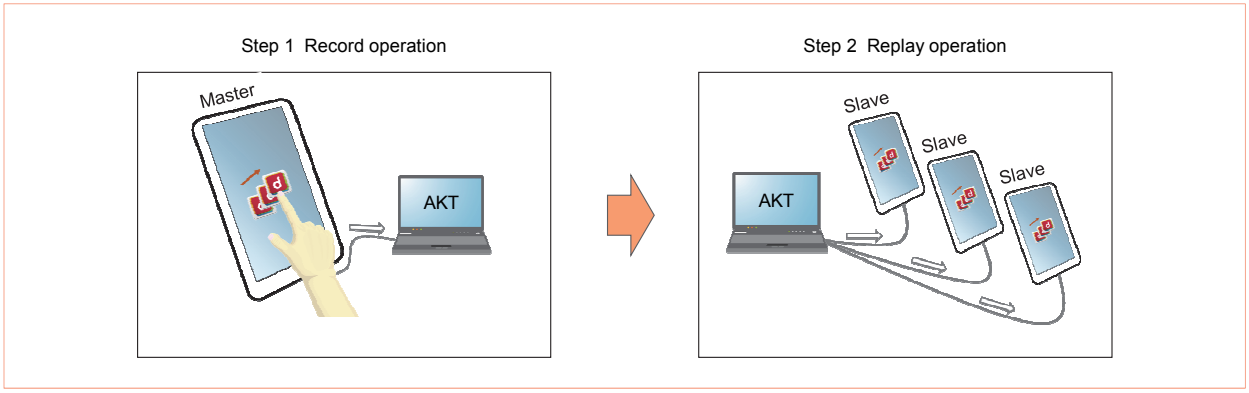

Figure 4 Image of operation scenario function

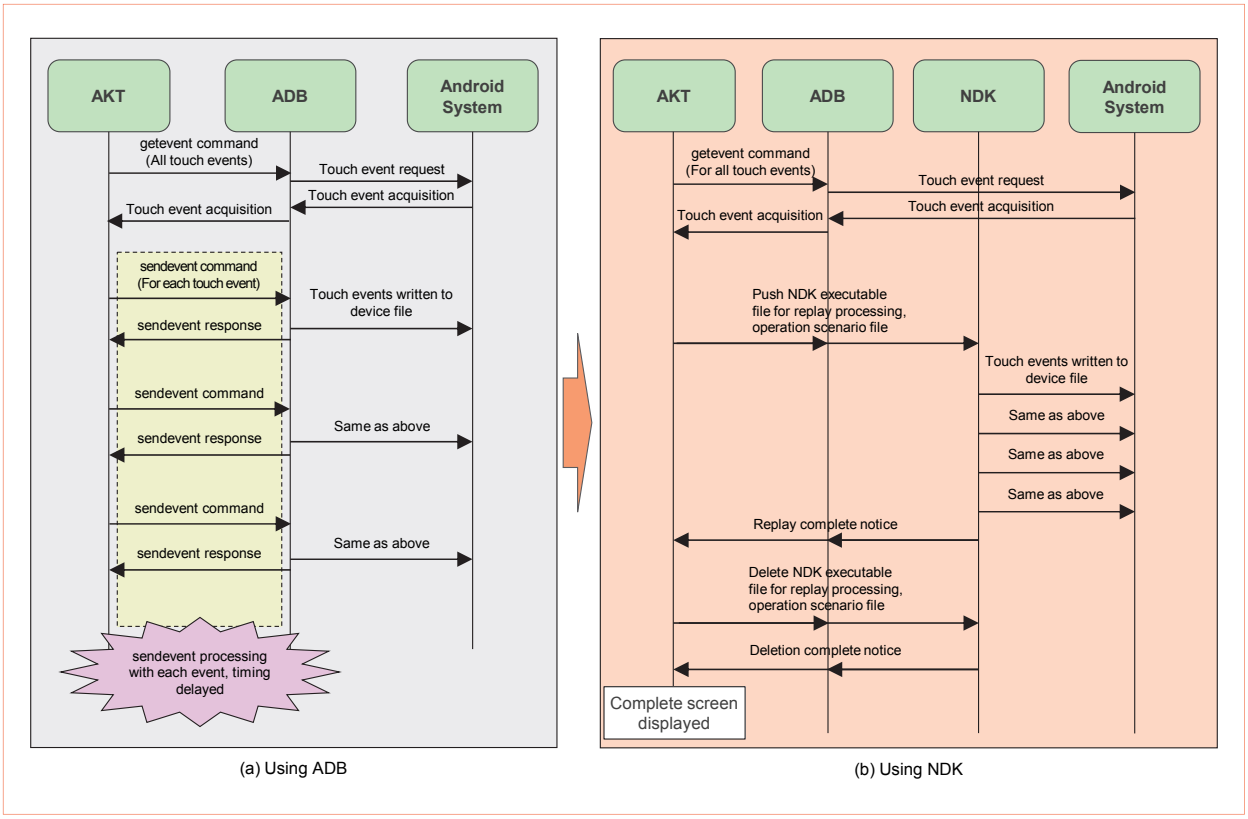

Figure 5 Operation scenario function sequence

operations, and greatly expands the AKT usage scope. Also, this simple-to-use function makes the tool easy to use for people who are troubled by difficult operations.

(2) Text paste function development

We also developed a function to automatically input different values into multiple Android smartphones.

This function enables input of values specified for individual handsets with AKT by loading a list of values corresponding to Android smartphone serial numbers in advance.

This function, in combination with the aforementioned operation scenario function, enables automation of inputs such as individual IDs which formerly required settings for each unit, and makes it possible to automate almost all operations required for kitting (**Figure 6**).

(3) API method expansion

We expanded items settable with APIs in cooperation with terminal manufacturers. Unlike UI Automator, in many cases, APIs are not model-dependent, and can be used with any model. By replacing the UI Automator with APIs, we were able to minimize the amount of AKT development required to support new models, which sped up development and reduced development costs.

AKT became established in-house because we broaden the scope of kitting that can be automated. This has greatly contributed to reducing work time, shortening delivery times and improving accuracy with kitting work.

Improving functionality greatly raised effectiveness. The following two points were key to this success.

- We held close and patient discussions with

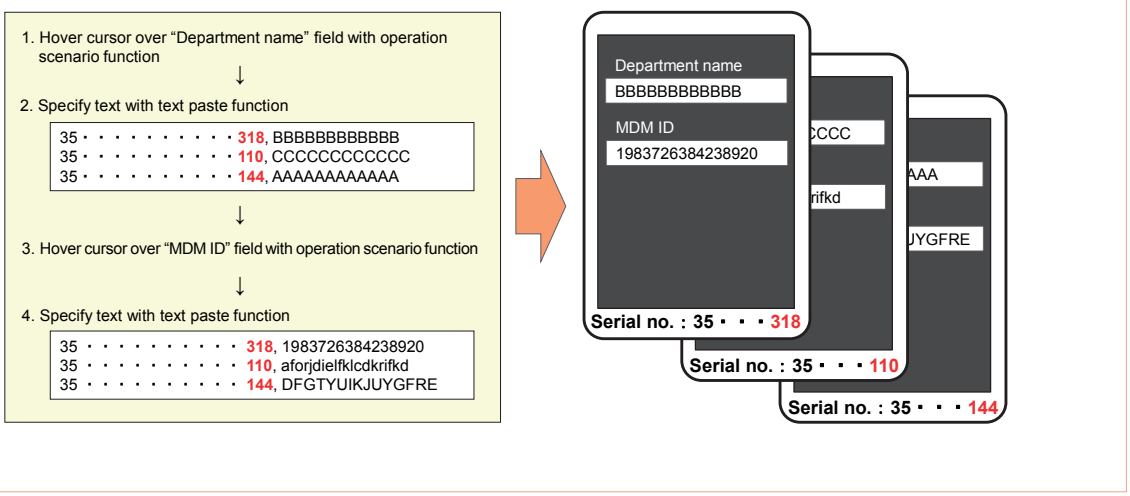

Figure 6　Text paste function

those in the workplace to accurately understand needs and determine the required functionality.

- To develop the exact functionality required, we avoided stereotypes and conceived new ideas from the ideal.

Thoroughly focusing on these two aspects enabled us to produce the current AKT and achieve more efficient kitting work. We will continue to advance AKT to further reduce workloads in the workplace.