ロボットアームを用いた、スマホアプリのテスト自動化の事例紹介

氏名　宮越一稀、中里大貴
所属　株式会社 NTT データ

はじめに・背景

物理カード読み取り操作が必要なスマホアプリの開発において、テスト自動化を行った事例について紹介する。
スマホアプリの開発において、スマホの機種特性による挙動の差異を確認するため、EtoE の多端末テスト実施が要件となっており、アプリの対象機種×対象 OS バージョンを網羅する必要があった。また本アプリの特徴として、スマホの NFC を用いて物理カードを読み取る操作が必要であり、物理カード読み取り操作を含む多端末テストの実施には、多大なコストが継続的にかかることが問題であった。

概要

物理カード読み取り操作が必要なスマホアプリの開発において、物理カードを用いた操作が自動化の障壁となるため、機種や OS バージョン毎に EtoE の多端末テストを人手で行うことが必要であり、実施には多大なコストが発生する。その対策としてロボットアームやスマホ操作を自動化する仕組みを導入することによって、物理カード読み取りを含めた一連のテスト操作の自動化を実現した。加えて、テスト実行の効率化のために、並列でのテスト実行の制御/モニタリング可能な仕組みを実装することにより、自動化前と比べて 37%の工数削減を達成した。

問題提起

当初は本テストを全て人手で実施する計画となっていたため、運用においてもデグレード確認のためにテストを実施し続けることを考慮すると、多大なコストが継続的にかかることが問題であった。そのためテストの自動化を行い、継続的なテスト実行のコスト削減を行うことを目標とする。

課題

テストの自動化について、実行上の課題は以下のとおりである。

【課題①】

自動化の対象とする操作には物理カード読取操作が含まれるが、従来のスマホの画面操作を自動化するツールはスマホの物理操作が想定されていないため、新規で物理カードを読み取る仕組みの構築が必要となる。

【課題②】

テスト対象の端末数が多いため、自動化した試験の実行結果の確認や資材インストールの際の手間が多い。

対策

【課題①の対策】

スマホの機種ごとにカードの読取位置が異なるため、カードのかざす位置を柔軟に変更可能なロボットアームを使用して、カード読取操作を自動化した。さらにスマホ操作の自動化は Appium で実装し、ロボットアームと連携することで、一連の操作の自動化を実現した。1 テスト実行環境にてスマホ 4 台分並列でテスト実行できるように実装し、現状 15 環境（スマホ端末計 60 台分）で運用している。

図 1 に、1 テスト実行環境の構成を示す。本自動化ツールは、コーディング不要でのテスト実装のため、キーワード駆動でテストスクリプトを作成可能であり、Excel で作成したテストシナリオをインプットとすることで、テスト実行可能としている。図 2 にテストシナリオの作成例を示す。テストシナリオに記入した順にスマホの操作を実施し、手順 5 のように、「カード読取」を記入した場合に、ロボットアームが呼び出される仕組みとなっている。

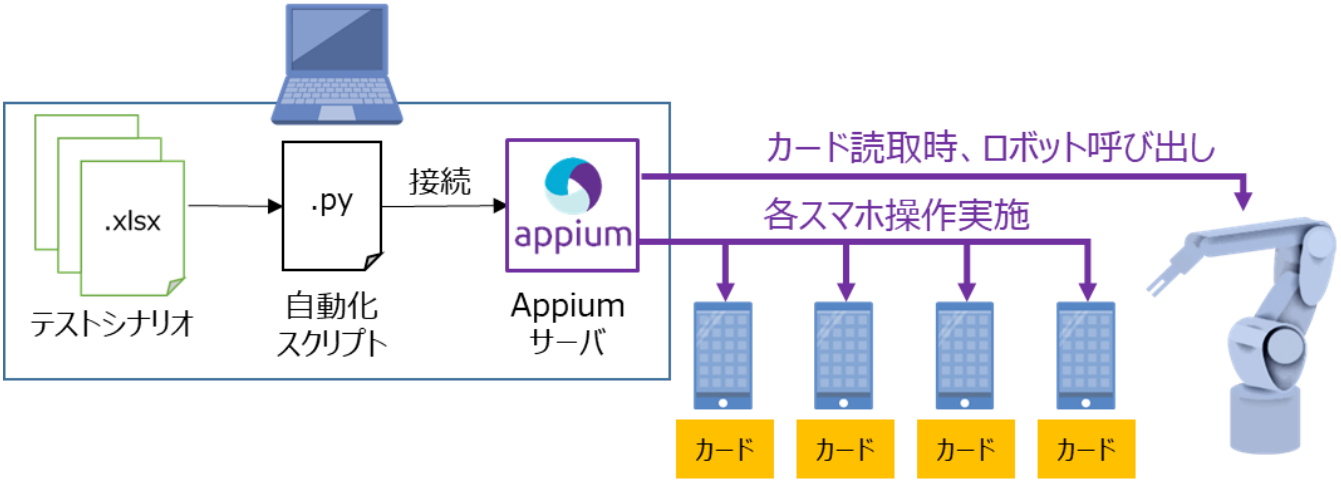

図１．１テスト実行環境の構成

手順	画面名	要素名	動作	入力値
1	トップ画面	メニューボタン	タップ	-
2	メニュー画面	カード登録ボタン	タップ	-
3	文字入力画面	入力フォーム	入力	123456
4	文字入力画面	次へボタン	タップ	-
5	カード読取画面	-	カード読取	-
6	読取完了画面	次へボタン	タップ	-

ロボットアームの動作

図２．テストシナリオの作成例

上記対策から、全テストケースの約 41%の自動化に成功し、対策導入前と比べ、約 31%の工数削減を達成した。

【課題②の対策】

GitLab CI の仕組みを活用し、図２のように、環境構成を改善した。これにより、各テスト実行環境へのリモートアクセスを不要とし、かつ全実行環境並列でのテスト実行の制御/モニタリング可能な仕組みを構築した。こちらの対策により、課題①の対策に加え、約 6%の工数削減を達成した。

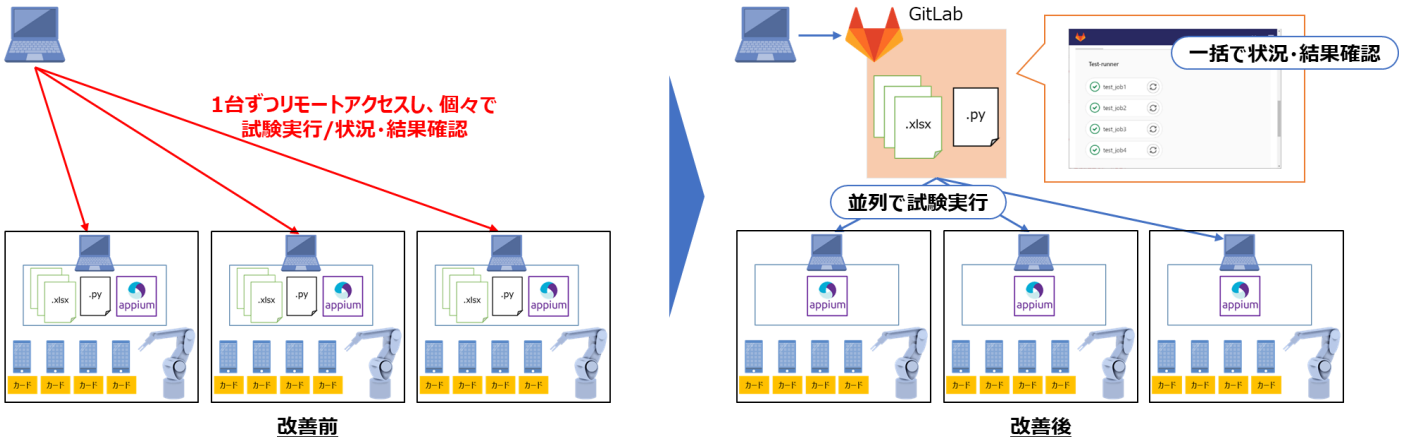

図３．GitLab CI を活用した環境構成の改善

【その他対策、苦労点】

上記対策に加え、以下の対策を実施した。

- 欠陥レポート起票内容の自動出力機能実装

  エラー時に起票する欠陥レポートについて、記載する情報が多く、起票に時間がかかっていた（1 起票 5～10 分程度）。この課題に対し、起票内容をテキストファイルとして自動出力するように追加実装した。同一内容の重複起票を避けるため、故障管理システムへの投入は手動で実施している。

- エラー時のチャットツールへの通知機能実装

  自動化ツール側のエラーによりテストが途中で止まった際、事象に気づくまでに時間がかかっており、すぐに気づくためには実施状況を GitLab からこまめに確認する必要があった。この課題に対し、チャットツールへの通知機能を実装することで、エラー時に即座に気づけるように改善した。

- 物理デバイス特有の課題対応

  スマホの機種によって、カードの読取位置やかざし方が異なり、カード読取ミスが頻発した。読み取り位置についての対策として、全機種分の読取位置を調査し、読取位置の定義ファイルを作成することで、実施機種に応じた読取位置へかざすように工夫した。また、かざし方については、スマホの読取位置の 1 点にカードをかざすのみで読取可能な機種、不可能な機種が混在していた。こちら調査したところ、手振れ（微小なスライド動作）が必要であることが分かり、読取位置の 1 点から上下左右にスライドするように追加実装したところ、全機種に対し、安定してカード読取を実行できるようになった。

結果・まとめ

上記の対策実施により、対策導入前と比べ、約 37%の工数削減を達成した。ただし、対策実施にかかった工数、及び物品にかかったコスト（ロボットアームなど）を全て工数換算すると、追加で約 80%要しており、今後の運用にて同様のテストを約 3 回実施することで、効果が出る見込みとなっている。

将来的な展望としては、Web ブラウザ操作との連携についても自動化することによる、テスト自動化ツールの汎用性向上や、画面判定を自動化することによる更なる生産性向上が挙げられる。