テスト技術者資格制度

Foundation Level Specialist シラバス
モバイルアプリケーションテスト担当者

Version 2019.J01

International Software Quality Institute (iSQI)提供
International Software Testing Qualifications Board

Copyright Notice

改訂履歴

ISTQB®

バージョン	日付	摘要
アルファ版	2018/5/11	アルファ版リリース
ベータ版	2019/1/27	ベータ版リリース
正式版	2019/3/28	正式版リリース
V2019	2019/5/10	ISTQB®リリース

JSTQB®

バージョン	日付	摘要
2019.J01	2021/7/15	V2019 の日本語翻訳版

目次

謝辞

本ドキュメントは International Software Testing Qualifications Board Mobile Application Testing Working Group のコアチームメンバにより執筆された。

Vipul Kocher (chair), Piotr Wicherski (vice-chair), José Díaz, Matthias Hamburg, Eran Kinsbruner, Björn Lemke, Samuel Ouko, Tal Pe'er, Ralf Pichler, Lloyd Roden, Nils Röttger, Angelina Samaroo, Yaron Tsubery

本コアチームは、レビューチームによる提案と意見に感謝したい。

次のメンバが、本シラバスのレビュー、意見表明、または投票に参加した。

Graham Bath, Veronica Belcher, Armin Born, Geza Bujdoso, YongKang Chen, Wim Decoutere, Frans Dijkman, Florian Fieber, David Frei, Péter Földházi Jr., Chaonian Guo, Attila Gyuri, Ma Haixia, Matthias Hamburg, Zsolt Hargitai, Hongbiao Liu, Ine Lutterman, Marton Matyas, Petr Neugebauer, Ingvar Nordström, Francisca Cano Ortiz, Nishan Portoyan, Meile Posthuma, Emilie Potin-Suau, Liang Ren, Lloyd Roden, Chaobo Shang, Mike Smith, Péter Sótér, Marco Sogliani, Michael Stahl, Chris Van Bael, Paul Weymouth, Salinda Wickramasinghe, Minghui Xu

本文書は、ISTQB®によって、2019 年 5 月 10 日に正式にリリースされた。

日本語訳については、以下の日本語翻訳ワーキンググループメンバにより行われた。

日本語翻訳ワーキンググループメンバ：

井芹 洋輝（トヨタ自動車）

須原 秀敏（ベリサーブ）

成安 和也（ベリサーブ）

松木 晋祐（ベリサーブ）

0 イントロダクション

0.1 本書の目的

本シラバスは、モバイルアプリケーションテスト Foundation Level 資格認定のベースとなる。
ISTQB®は、本シラバスを次の趣旨で提供する。

1. 各国の委員会に対し、各国語への翻訳および教育機関の認定の目的で提供する。各国の委員会は、本シラバスを各国語の必要性に合わせて調整し、出版事情に合わせてリファレンスを修正することができる。
2. 試験委員会に対し、各シラバスの学習目的に合わせ、各国語で試験問題を作成する目的で提供する。
3. 教育機関に対し、コースウェアを作成し、適切な教育方法を確定できるようにする目的で提供する。
4. 受験志願者に対し、試験準備の目的で提供する（研修コースの一部として、または独立した形態で実施）。
5. 国際的なソフトウェアおよびシステムエンジニアリングのコミュニティに対し、ソフトウェアやシステムをテストする技能の向上を目的とする他、書籍や記事を執筆する際の参考として提供する。

ISTQB®では、事前に書面による申請があった場合に限り、第三者がこのシラバスを先に定めた以外の目的での使用を許諾することがある。

0.2 モバイルアプリケーションテスト Foundation Level 資格制度

Foundation Level 資格認定は、モバイルアプリケーションテストに関する知識を深めることを希望する技術者、またはモバイルアプリケーションテストの分野で専門家としての経験を積むことを希望する技術者を対象にすることを意図している。

本シラバスを作成する際には、ISTQB®テスト技術者資格制度 Foundation Level シラバス [ISTQB_CTFL_2018]に記載されているモバイルアプリケーションテストに関する情報を考慮している。

0.3 ビジネス成果

本節では、モバイルアプリケーションテスト Foundation Level 認定を取得した認定資格者に期待されるビジネス成果をリストする。

MAT-01テスト戦略を作成するために、モバイルアプリケーションのビジネス的および技術的推進要因を理解し、確認する。

MAT-02モバイルアプリケーションのテストに関連する主要な課題、リスク、および期待を特定し、理解する。

MAT-03モバイルアプリケーションに固有のテストタイプとレベルを適用する。

MAT-04モバイル固有の状況で、[ISTQB_CTFL_2018]で言及されている一般的なテストタイプを適用する。

MAT-05ISTQB®テストプロセスで説明されている主要な活動の一環として、特にモバイルアプリケーションテスト向けに必要な活動を実行する。

MAT-06モバイルアプリケーションテスト向けの適切な環境とツールを特定し使用する。

MAT-07特にモバイルアプリケーションテスト自動化に役立つ方法とツールを理解する。

0.4 試験対象の学習の目的

これらの学習の目的はビジネス成果を支援し、モバイルアプリケーションテスト Foundation Level 認定を達成するために合格する必要のある試験を作成するために使用できる。学習の目的には、知識の認知レベル（K レベル）が割り当てられている。

K レベル（認知レベル）は、Bloom の改訂版分類法に従って学習の目的を分類するために使用される[Anderson 2001]。ISTQB®ではこの分類法を使用して、シラバスに対する試験をデザインする。

本シラバスでは、3 つの異なる K レベル（K1～K3）を考慮する。詳細については、第 7 章を参照されたい。

0.5 コンピテンシーのハンズオンレベル

モバイルアプリケーションテスト Foundation Level では、実践的なスキルおよびコンピテンシーに重点を置くハンズオンの目的を導入している。

コンピテンシーは、ハンズオン実習を実行することで達成できる可能性がある。その例の一部を次に示す。

- K3 レベルの学習の目的の実習。この際には、既存のさまざまな ISTQB®シラバスの場合と同様に、紙と筆記用具、またはワープロソフトウェアを使用する。
- テスト環境のセットアップと使用。
- 仮想デバイスおよび物理デバイスでのアプリケーションのテスト。
- デスクトップそして／またはモバイルデバイスでツールを使用して、インストール、照会、記録作成、監視、スクリーンショットの採取など、テスト関連のタスクをテストまたは支援する。

ハンズオンの目的には、次のレベルが適用される。

- H0：ライブデモまたは動画などによる実習。これは受講生によって実行されないため、厳密な意味で実習ではない。
- H1：講師によるガイダンスを伴う実習。受講生は講師が実行する一連の手順に従う。
- H2：ヒントを使用した実習。受講生には実習と関連するヒントが与えられる。ヒントは制限時間内に実習を解決できるように作成される。
- H3：講師によるガイダンスもヒントもない実習。

推奨事項。

- K1 の学習の目的では、H0 レベルを主として、必要に応じて、H1 または H2 のレベルを使用する。
- K2 の学習の目的では、H1 または H2 のレベルを主として、必要に応じて、H0 または H3 のレベルを使用する。

- K3 の学習の目的では、ハンズオン実習を行うことは必ずしも必要ではないが、行う場合は、通常、H2 または H3 のレベルを使用する。セットアップが複雑であるか、またはあまりに時間がかかる場合は、H0 レベルを使用する。

0.6 試験

モバイルアプリケーションテスト Foundation Level の認定資格試験は本シラバスに基づく。試験問題に解答するには、本シラバスの複数の節にまたがる知識を必要とする場合もある。本シラバスの節のすべては、「イントロダクション」と付録を除いて試験対象である。標準、文献、および他の ISTQB シラバスを情報源としているが、それらに関して本シラバス自体の中で要約されている以上の内容は試験対象ではない。

モバイルアプリケーションテスト Foundation Level 試験の形式は、多肢選択式である。問題の数は 40 である。問題の 65%（26 の問題）以上を正解した場合に合格となる。ハンズオンの目的と実習は、試験の対象外である。

試験は、認定トレーニングコースの一部として、または（例えば試験センターや公的試験で）独立して実施してもよい。認定トレーニングコースの受講完了は試験のための前提条件ではない。

認定教育機関による講習を受講せずに受験を希望する場合、『Accreditation and Competence Guidelines（認定およびコンピテンスガイドライン）』文書[CTFL-MAT-2019-Accreditation-and-Competence-Guidelines.pdf]として提供されているコンピテンスガイドラインを熟読し、これらの実習を自身で試行する必要がある。これは、認定教育機関による講習で教わるコンピテンシーを習得するのに役立つ。ただし、モバイルアプリケーションテスト Testing Foundation Level 認定試験は本シラバスとその学習の目的のみに基づくため、これは、試験に関係ないことに注意願いたい。

0.7 推奨される学習時間

最少学習時間は、本シラバスの学習の目的ごとに定義されている。各章の合計時間が、章見出しで示されている。

他の ISTQB®シラバスでは、K レベルに応じた固定時間を割り当てる「標準時間」手法が適用されることに、教育機関は注意する必要がある。モバイルアプリケーションテストシラバスでは、この手法を厳密に適用していない。このため、教育機関は、各学習の目的に対してより柔軟に現実的な最少学習時間を割り当てることができる。

0.8 受験資格

本試験の受験者は、ISTQB® Foundation Level 認定を取得している必要がある。

0.9 参照情報

本シラバスで使用されている用語は、ソフトウェアテストで使用されている ISTQB®用語集[ISTQB_GLOSSARY]で定義されている。

モバイルアプリケーションテストに関して推奨される書籍および記事の一覧が、第 6 章に記載されている。

1 モバイル分野 -ビジネスとテクノロジーの原動力 | 175 分

キーワード

リスク分析、リスク軽減、リスクベースドテスト、テスト戦略

「ビジネスとテクノロジーの原動力」の学習の目的

1.1 モバイル分析データ

MAT-1.1.1 （K2）利用可能なモバイル分析データをテスト戦略とテスト計画の入力として使用する方法を説明する。

HO-1.1.1 （H3）1 つ以上の分析データソース（地域、プラットフォーム、オペレーティングシステムバージョン、およびデバイスタイプの分布）から収集されたデータに基づいて、テスト対象のデバイスタイプとそれぞれの優先度を選択する。

注：HO-1.1.1 と次の HO-1.7.1 は 1 つにまとめることも可能。

1.2 モバイルアプリケーションのビジネスモデル

MAT-1.2.1 （K2）モバイルアプリケーションのさまざまなビジネスモデルを識別する。

1.3 モバイルデバイスの種類

MAT-1.3.1 （K1）さまざまな種類のモバイルデバイスを想起する。

1.4 モバイルアプリケーションのタイプ

MAT-1.4.1 （K2）さまざまな種類のモバイルアプリケーションを識別する。

1.5 モバイルアプリケーションアーキテクチャ

MAT-1.5.1 （K2）モバイルアプリケーションの一般的なアーキテクチャタイプを識別する。

1.6 モバイルアプリケーションのテスト戦略

MAT-1.6.1 （K3）テスト戦略を準備する際にモバイル市場の特徴と特殊性を考慮する。

1.7 モバイルアプリケーションテストの課題

MAT-1.7.1 （K2）モバイルアプリケーションのテストに付随する課題を例示する。

HO-1.7.1 （H1）選択した地域におけるデバイスまたはオペレーティングシステムの市場占有率などの市場データを収集する。画面サイズや画素密度のデータを収集する。5 つのデバイスのリストを作成し、このリストでの期待される市場カバー率を計算する。

注：前述の HO-1.1.1 と HO-1.7.1 は 1 つにまとめることも可能。

1.8 モバイルアプリケーションテストのリスク

MAT-1.8.1　（K2）モバイルアプリケーション固有のリスクを軽減する方法を説明する。

1.1 モバイル分析データ

モバイル分野には、多くのステークホルダーが存在する。これらのステークホルダーには、製造業者、プラットフォームプロバイダー、オペレーティングシステム（OS）プロバイダー、市場データプロバイダー、ツールプロバイダー、もちろんのことながら、アプリケーション開発者およびテスト担当者などが含まれる。

テスト計画の検討とテスト分析を効果的に行うために、モバイルアプリケーションのテスト担当者は、次の項目について認識し、精通している必要がある。

- プラットフォーム分布のビジネスへの影響
- プラットフォームごとのアプリケーションダウンロード数
- OS バージョンの量と分布
- さまざまなデバイスの種類の市場分布（地域ごとの特性を含む）
- さまざまな画面サイズと解像度
- さまざまな入力方法
- カメラタイプ

これらに関しては、無償および有償の両方で、さまざまな情報ソースが存在する。これらの情報ソースには、StatCounter GlobalStats [URL1]、OS ベンダー自身、およびその他のサードパーティがある。

モバイル分析データを使用して、対象市場にとって適切なデバイスポートフォリオを、テスト実行のために選択する。このポートフォリオから読み取れるデバイスの重要性に従ってデバイスを選択し、そのデバイス上でアプリケーションをテストする。デバイスに関連するデータおよびデバイス特有の機能（存在する場合）も使用して、デバイスタイプに固有のテストを設計する。例えば、デバイスに心拍センサーが搭載されている場合、特別なテストケースが必要な場合がある。

1.2 モバイルアプリケーションのビジネスモデル

モバイルアプリケーションの作成において、完成した製品を収益化するために使用できるビジネスモデルがいくつか存在する。これらには、例えば、フリーミアム、広告ベースド、取引ベースド、有償、エンタープライズなどのアプリケーションがある。さらに、これらのモデルの一部では、アプリ内課金が適用できる場合がある。

これらの手法にはそれぞれ長所と短所があり、モバイルアプリケーションのテスト担当者は、ビジネスモデルを念頭に置いておく必要がある。

フリーミアムモデルでは、アプリケーションは無償であるが、追加機能は有償である。アプリケーションはユーザーが使いたくなる機能を十分に備えている必要があり、しかも、大多数のユーザーが購入したくなる高度な機能を提供する必要がある。

広告ベースドアプリケーションでは、ユーザーがアプリケーションを操作すると、画面に広告が表示される。この収益創出の戦略は、アプリケーションの使用時間が長くなるほど効果的である。ユーザーインターフェースの設計者は、広告を表示するタイミングに注意する必要がある。広告は人目を十

分に引き付け、アプリケーションの重要な部分を隠すことなく、ユーザーがアプリケーションの使用時に不快感や嫌悪感を抱くことがないようにする必要がある。

取引ベースドアプリケーションでは、取引ごと、定額、または取引価格の一定割合などのいずれかの方法でユーザーに課金する。このタイプのビジネスモデルはごく少数のアプリケーションのみに適しており、主に、モバイルウォレットなどビジネスアプリケーションおよび金融アプリケーションで使用される。

有償アプリケーションの場合、ユーザーは料金を支払って、アプリケーションのダウンロードおよびインストールを行う。ほとんどのアプリケーションタイプに対して、無償またはフリーミアムの選択肢が多く存在するため、有償ビジネスモデルを採用する場合は、十分に検討する必要がある。このようなアプリケーションについてユーザーの購買意欲が高まるのは、アプリケーションが傑出した機能または使用性を備えているか、競合するアプリケーションが利用できない場合である。

無償アプリケーションおよびエンタープライズアプリケーションでは、ユーザーに課金されない。エンタープライズアプリケーションは、組織内での内部使用のために開発され、提供されるサービスへのインターフェースを提供する。そのようなアプリケーションは、金融機関や E コマース企業などの組織から多数提供されている。これらのアプリケーションでは、一般的に、アプリケーション自体を収益化することに重点を置かないが、それらの組織が提供するサービスにユーザーをアクセスさせることで収益の創出を図る。

1.3 モバイルデバイスの種類

さまざまな種類のアプリケーションに対応したモバイルデバイスが利用可能である。

モバイルデバイスには、次のような種類がある。

- ベーシックフォン
- フィーチャーフォン
- スマートフォン
- タブレット
- コンパニオンデバイス（ウェアラブルおよび一部の IoT（Internet of Things）デバイスを含む）

テストにおいては、デバイスの種類固有のニーズに対する特定の機能があることに注意する必要がある。

ベーシックフォンは電話と SMS だけに使用され、内蔵のアプリケーションやゲームはほとんどない。アプリケーションのインストールやインターネット参照は不可能である。

フィーチャーフォンでは、アプリケーションとアプリケーションのインストールに対する限定されたサポートを提供する。内臓のブラウザを介してインターネットアクセスが可能であり、カメラなどの追加ハードウェアを搭載している場合もある。

スマートフォンは、複数のセンサーを搭載した電話である。搭載されているオペレーティングシステムにより、アプリケーションのインストール、マルチメディアおよびインターネットの閲覧などの機能がサポートされる。

タブレットはスマートフォンに似ているが、物理的なサイズがスマートフォンに比べて大きい。より大きな画面が必要とされる場合に使用されることが多く、バッテリー駆動時間もスマートフォンより長い。

コンパニオンデバイスと一部の IoT 製品は、コンピューター駆動のデバイスで、スマートフォンやタブレットと連携して使用され、利用可能な機能を拡張する、または、スマートフォンやタブレット上のデータへのより使いやすい方法でのアクセスを可能にする。

ウェアラブルは、利用者が身に着けることができるデバイスである。これらは、既存デバイスと対になって動作するか、独立して機能する。普及しているウェアラブルの例としては、スマートウォッチやスマートブレスレットがある。

1.4 モバイルアプリケーションのタイプ

モバイルアプリケーションは、主に次の 3 つのタイプに分類される。

- ネイティブ
- ブラウザベース
- ハイブリッド

アプリケーションの各タイプには、それぞれ長所と短所があり、アプリケーション開発を開始する前に、ビジネス上の意思決定を行う必要がある。

ネイティブアプリケーションの開発は、プラットフォーム固有のソフトウェア開発キット（SDK）、開発ツール、およびプラットフォーム固有のセンサーや機能を使用して行う。ネイティブアプリケーションは、サプライヤのストアからダウンロード、インストール、およびアップデートを行うことができる。これらのアプリケーションは、サポート対象の全デバイスでテストする必要がある。

ネイティブアプリケーションは、一般的に優れた性能を発揮し、プラットフォームの機能をフルに活用し、開発対象となるプラットフォームに求められるものに適合する。開発コストは高くなる傾向にあり、複数プラットフォームの使用や大量のデバイスでのインストールやテストなどの課題が追加で発生する可能性もある。

ブラウザベースドアプリケーションは、モバイルブラウザを介してアクセスされる。このタイプのアプリケーションでは、典型的な Web 開発技術とブラウザが使用されるため、複数プラットフォームのサポートを容易に実現でき、開発コストも一般的に低い。

モバイル Web アプリケーションは、主に次の 4 つの方法を使用して開発される。

- モバイル固有バージョンの Web サイトおよびアプリケーション（これらは m（dot）サイトとも呼ばれる）。この方法では通常、モバイルブラウザからアプリケーションにアクセスすると、モバイルバージョンのアプリケーションが提供される。例えば、モバイルデバイスから facebook.com にアクセスすると、m.facebook.com に転送される。
- レスポンシブ Web アプリケーションにより、ビューポートとして一般的に表現されるフォームファクターと画面サイズに合わせてデザインが調整される。
- アダプティブ Web アプリケーションでは、いくつかの事前定義のサイズに従ってデザインを調整する。これらのサイズ向けに異なるデザインが存在し、ユーザーが利用できる機能は調整できる場合が多い。
- プログレッシブ Web アプリケーションでは、特定の Web ページのショートカットをモバイルホーム画面上に作成できる。このタイプのアプリケーションはネイティブアプリケーションのように表示され、オフラインで動作するものもある。

モバイル Web アプリケーションの開発は、一般的な Web 技術を使用して行われるため、ネイティブアプリケーションやハイブリッドアプリケーションに比べて、開発や管理が容易であることが多い。

ただし、機能面ではネイティブアプリケーションやハイブリッドアプリケーションに比べて劣る可能性があり、プラットフォームのネイティブアプリケーションプログラミングインターフェース（API）へのアクセスが制限される場合もある。モバイルデバイスのセンサーへのアクセスも制限される。各デバイスでの設置性テストは必要ないが、ブラウザ互換性テストは必要である。

ハイブリッドアプリケーションでは、ネイティブアプリケーションと Web アプリケーションを組み合わせて使用する。Web アプリケーションは、Web ビューを含むネイティブアプリケーションラッパーを使用して、ネイティブアプリケーション内で実行する。これらのアプリケーションはサプライヤストアからダウンロードされ、デバイスの全機能にアクセスできる。開発、アップデート、およびメンテナンスは比較的容易であり、デバイスにインストールされているアプリケーションをアップデートする必要はない。これらのアプリケーションの開発に必要なスキルは、Web 開発の場合とほぼ同じである。これらのアプリケーションの弱点として、ラッパー使用による性能の問題、プラットフォームによっては期待されるルックアンドフィールからの逸脱などが発生する可能性がある。

ネイティブアプリケーションとハイブリッドアプリケーションは、デバイスに物理的にインストールされる。従って、デバイスがインターネットに接続されているかどうかに関係なく、利用可能である。これに比べて、ブラウザベースドアプリケーションは、インターネット接続を必要とする。

一部のアプリケーションはモバイルデバイスに予めインストールされている。その他のアプリケーションは、さまざまな配布チャネルを介してインストールできる。これらのチャネルには、Apple App Store、Google Play Store、エンタープライズアプリケーションストア（エンタープライズネットワーク内での利用可）、サードパーティアプリケーションマーケットなどがある。

これらのアプリケーションタイプごとに、異なるテスト方法が必要になる場合がある。次の項目を考慮する必要がある。

- 異なるタイプのデバイスをサポート
- センサーおよびデバイス機能を使用
- さまざまなネットワーク条件での可用性
- 設置性、互換性、性能効率性、使用性

1.5 モバイルアプリケーションアーキテクチャ

モバイルアプリケーションを設計するためのソリューションは複数存在する。

特定のアーキテクチャの選択または設計の意思決定を行う際には、次の点を考慮する必要がある。

- 対象にするユーザー
- アプリケーションタイプ
- さまざまなモバイルプラットフォームおよび非モバイルプラットフォームのサポート
- 接続性のニーズ
- データストレージのニーズ
- IoT アプライアンスを含む他のデバイスへの接続

アーキテクチャに関しては、次の点を意思決定する必要がある。

- シンクライアント、ファットクライアントなど、クライアント側のアーキテクチャ
- シングルティア、マルチティアなどのサーバー側のアーキテクチャ
- Wi-Fi、モバイルデータ通信、近距離無線通信（NFC）、Bluetooth
- ストアアンドフォワード、プッシュ／プル、同期／非同期通信などのデータ同期方法

シンクライアントアプリケーションは、デバイス固有のアプリケーションコードを含まず、モバイルオペレーティングシステム機能の使用は最小限である。これらのアプリケーションでは、一般的に、フロントエンドとして Web ブラウザを、クライアント側のロジックを実装するための言語として JavaScript を使用する。

シック／ファットクライアントアプリケーションでは、複数レイヤのアプリケーションコードを持つことが可能であり、モバイルオペレーティングシステム機能を使用できる。これらは、ネイティブアプリケーションまたはハイブリッドアプリケーションであることが多い。

サーバー側では、次のアーキテクチャを採用できる。

- シングルティアアーキテクチャ：一体型であり、すべてのサーバーが同一マシン上に存在する。拡張性に乏しく、安全に保護することが困難である。
- マルチティアアーキテクチャ：さまざまなユニット全体でサーバー側コンポーネントを分散配置する。2 ティアアーキテクチャでは、Web サーバーとデータベースサーバーを分離し、3 ティアアーキテクチャでは、アプリケーションサーバーも分離する。マルチティアアーキテクチャでは、処理の分担が可能であり、データベース専用化を実現し、柔軟性、拡張性、セキュリティに優れている。ただし、開発、管理、およびホスティングの費用が、シングルティアアーキテクチャに比べて、極めて大きい。

さまざまな接続方法を利用できるモバイルデバイスは、Wi-Fi などの接続タイプや、2G、3G、4G、5G などのモバイルデータ通信を介してサーバーに接続できる。モバイルアプリケーションは、通常、次の 3 つのモードのいずれかで動作する。

- 接続されることのないアプリケーション：オフラインで動作し、接続を必要としない。このようなアプリケーションとしては、単純な「電卓」がある。
- 常時接続アプリケーション：動作時に常時ネットワーク接続を必要とする。すべてのモバイル Web アプリケーションはこのカテゴリに分類される。一部のアプリケーションは必要時接続のモードでも限定的に動作できる。
- 必要時接続アプリケーション：長時間にわたって接続なしで動作でき、データ転送などのタスクにおいて接続を必要とする。

クライアントとサーバーの間のデータ同期は、次のようなモードで実現される。

- 連続モード：データは、サブミットされるとすぐに転送される。
- ストアアンドフォワードモード：特に接続が利用できない場合に、データはローカルに保存され、その後転送される。

データ転送には、次の 2 つの方法が利用できる。

- 同期データ転送：呼び出した側の関数は、呼び出された側の関数が完了して実行権を返すまで待機する。

- 非同期データ転送：呼び出された側のサーバー関数は、実行権を直ちに返して、データをバックグラウンドで処理し、タスクの完了後に呼び出した側のクライアント関数にそのことを通知する。この方法では、ユーザーの待ち時間は短くなる。サーバーがコールバックを返す際のクライアントまたはネットワークの可用性を考慮する必要があるため、ハンドシェイクメカニズムの実装は複雑性を増大させる。

1.6 モバイルアプリケーションのテスト戦略

モバイルデバイス用のテスト戦略を作成する場合、テスト担当者は、本章でこれまでに説明している全項目について考慮する必要がある。さらに、本節で説明されているリスクと 1.7 節で説明されている課題についても、考慮する必要がある。

一般的なリスクの例を、次に示す。

- 特定の地域でのデバイス普及率データを知らずして、継続可能な方法でアプリケーションをテストする必要のあるデバイスを選択することはできない。
- ビジネスモデルの種類を知らなければ、アプリケーションの振る舞いがビジネスモデルにうまく適合しているかどうかをテストすることはできない。

モバイルアプリケーションのテスト用にテスト戦略を作成する場合、次に記載する固有のリスクや課題を追加で考慮する必要がある。

- モバイルデバイスの多様性と、一部のデバイスでの固有の欠陥。
- 社内または外部テストラボの使用によるデバイスの可用性。
- アプリケーションライフサイクルにおける新しい技術、デバイス、またはプラットフォームの導入。
- 多様なチャネルを介してのアプリケーション自体のインストールとアップグレード（アプリケーションのデータと基本設定の保持を含む）。
- アプリケーションに影響を及ぼす可能性のあるプラットフォームの問題。
- 世界中で使われることを考えた場合のネットワークのカバレッジとアプリケーションに及ぼす影響。
- 多様なサービスプロバイダーのネットワークを使用してのテストの可能性。
- 特定のテストレベルおよびテストタイプ向けのモバイルエミュレーター、シミュレーター、そして／または実デバイスの使用。

これらの課題の詳細については、1.7 節を参照されたい。

テスト戦略では、リスクと課題を考慮する。そのような例を次に示す。

- 開発の初期段階でモバイルデバイスのエミュレーター／シミュレーターの使用を、後続の段階では実デバイスの使用を、テスト戦略で指定される可能性がある。特定のテストタイプでは、モバイルエミュレーター／シミュレーターでテストを実行できるが、できないテストタイプも存在する。詳細については、4.3 節を参照されたい。
- テスト戦略では、大量の異なる種類のデバイスの準備という課題に対して、次のアプローチのいずれかを採用することを考慮することが必要な場合がある。

- シングルプラットフォームアプローチ：1 つのデバイスタイプ、1 つの OS バージョン、1 つの通信事業者、1 つのネットワークタイプにテストスコープを削減する。
- マルチプラットフォームアプローチ：モバイルトラフィックまたは他の分析データに基づいて、ターゲット市場の大半の顧客により使用される代表的なデバイスと OS にテストスコープを削減する。
- 最大カバレッジアプローチ：すべての OS バージョン、デバイス、製造元、通信事業者、ネットワークタイプを対象にする。これは実際には全数テストであり、特に、市場で提供されているデバイスおよび OS バージョンの数が多い場合、経済的観点からは採用できるものではない。

- テスト戦略では、デバイス、ネットワーク、または実生活での状況が利用できないという課題に対して、次のような外部リソースの使用を考慮することが必要な場合がある。
  - リモートデバイスアクセスサービス：所有していないデバイスに、Web を介してアクセスする方法である。
  - クラウド（群衆）テストサービス：これは、大人数のボランティアおよび彼らのデバイスにアクセスする手段を提供するサービスである。
  - 個人のネットワーク：友人や同僚など、個人のプライベートソーシャルネットワークを活用する。
  - バグハンティング：ゲーム化されたテストイベントで、企業または一般大衆からのボランティアを活用する。

テスト戦略では、[ISTQB_CTFL_2018]で説明されているテストレベルに加えて、モバイルアプリケーション向けに共通して使用されるテストタイプ（3.1 節参照）および必要なあらゆる追加テストレベル（3.2 節参照）を考慮する必要がある。

1.7 モバイルアプリケーションテストの課題

モバイル分野には、デスクトップまたはサーバーソフトウェアで一般的でないか、重要でない多くの課題が余計に存在する。テスト担当者はこれらの課題およびそれらがアプリケーションの成功に及ぼす影響の度合いに留意する必要がある。

モバイル分野での一般的な課題を次に示す。

- プラットフォームおよびデバイスの多様性：複数の OS タイプとバージョン、画面サイズ、ディスプレイの質。
- さまざまなデバイスのハードウェア上の相違：センサータイプの多様性、CPU や RAM リソースの制約に対するテストの条件のシミュレーションの困難さ。
- プラットフォームごとに必要な開発ツールの多様性。
- ユーザーインターフェースデザインとプラットフォームに期待されるユーザー操作性（UX）の多様性。
- 複数のネットワークタイプとプロバイダー。
- リソース不足のデバイス。
- アプリケーションの配布チャネルの多様性。

- ユーザーおよびユーザーグループの多様性。
- アプリケーションのタイプと接続方法の多様性。
- バグに起因するフィードバックの可視性の高さ。ユーザーに大きな影響を及ぼすバグが存在した場合、ユーザーはオンラインマーケットプレイスにフィードバックを容易に公開できる。
- マーケットプレイスでの公開。公開には、Google Play や Apple App Store などのマーケットプレイス所有者の承認サイクルが余計に必要である。
- モバイルエミュレーター／シミュレーターの使用が必要となる新発売デバイスの使用可能性。

これらの課題の影響を次に示す。

- 大量の組み合わせをテストする必要がある。
- 大量のデバイスをテストする必要があり、コストを上昇させる。
- 旧バージョンのプラットフォームでアプリケーションを実行するための下位互換の必要性。
- オペレーティングシステムのバージョンごとにリリースされる新機能。
- さまざまなプラットフォームで考慮する必要のあるガイドライン。
- リソース不足の CPU、メモリとストレージの容量の制限。
- さまざまなネットワークの帯域幅とジッターの多様性。
- データ（契約）プランに基づく、アップロードとダウンロードの利用可能な速度の変化。

次に、典型的な課題とそれらの潜在的な影響の 2 つの例を示す。

- 異なるデバイスには異なるタイプのセンサーが搭載されており、これらに対応するためのテストが必要である。ハードウェアに新しいセンサーが追加されるごとに、余計な下位互換性テストが必要になる場合がある。
- ネットワークの課題のいくつかは、ネットワークの状況の変化に応じて、適切なキャッシング戦略およびプレフェッチング戦略を使用することで、対処できる場合がある。ただし、ほとんどのアプリケーションでユーザーはサーバーにログインしたままになっているため、コストの上昇や、大量のオープン接続がサーバー側の性能に影響を及ぼす可能性がある。

1.8 モバイルアプリケーションテストのリスク

1.7 節で説明されている課題は、単独で発生することも、他との組み合わせで発生することもある。これは、モバイルアプリケーションに追加のリスクをもたらす可能性がある。

テスト担当者は、プロダクトのリスク分析に寄与できる必要がある。[ISTQB_CTFL_2018]、5.5 節で説明されている一般的なリスク分析と軽減方法は、モバイル環境でも適用できる。さらに、モバイル固有の次に示すリスクと軽減の戦略も存在する。

リスク	可能性のある軽減策
市場での多種多様なデバイス	テスト実行用にデバイスを適切に選択する。例えば、最も一般的に使用されているデバイスをテストする。
複数のプラットフォームをサポートするコスト	最も使用されているプラットフォームを把握するために分析を実行し、対象外のデバイスのテストを回避する。
新しい技術、プラットフォーム、およびデバイスの導入	それらの技術の試作（正式リリース前）バージョンを使用する。
テスト実行用のデバイスが使用不可	リモートデバイスアクセスサービスまたはクラウドテストサービスを適用する。
外出中に使用されるモバイルアプリケーションの予測される使用パターンのリスク	フィールドテストなど、適切なテスト方法を適用する。

2 モバイルアプリケーションのテストタイプ 265 分

キーワード

共存性、互換性、接続性、ブラウザ間互換性、相互運用性、テスト対象システム（SUT）、テストタイプ、使用性

「モバイルアプリケーションのテストタイプ」の学習の目的

2.1 デバイスハードウェアとの互換性テスト

MAT-2.1.1 （K2）テストにおいて考慮しなければならないデバイス固有機能とハードウェアを説明する。

HO-2.1.1 （H1）テスト対象システム（SUT）が正しく機能していることを検証するためにSUTを使用しながら、モバイルデバイスの複数の機能向けのアプリケーションをテストする。

MAT-2.1.2 （K3）画面サイズ、アスペクト比、および画面密度に関してアプリケーションの互換性をテストするための準備を行う。

HO-2.1.2 （H3）アプリケーションを複数のモバイルデバイス（仮想または物理）でテストして、解像度と画面サイズがアプリケーションのユーザーインターフェースに及ぼす影響を示す。

MAT-2.1.3 （K2）テスト対象システムでデバイスをオーバーヒート状態にした場合の潜在的な影響を、テストでどのように示すことができるかを説明する。

MAT-2.1.4 （K1）モバイルデバイスで使用されるさまざまな入力センサーのテスト向けの異なるテストタイプを想起する。

MAT-2.1.5 （K1）さまざまな入力方法に対して実行する必要のあるテストを想起する。

HO-2.1.5 （H0）複数のインストール済みキーボードでのキーボード関連のテスト、ジェスチャー関連テスト、カメラ関連のテストなど、さまざまなタイプの入力に対してアプリケーションをテストする。

MAT-2.1.6 （K2）画面の向きを変更した場合のユーザーインターフェースの問題を、テストでどのように明らかにすることができるかを説明する。

HO-2.1.6 （H3）データ保持やユーザーインターフェースの正しさなどのアプリケーションの機能に、画面の向きの変更が及ぼす影響をチェックするために、アプリケーションをテストする。

MAT-2.1.7 （K3）典型的なモバイルデバイスの割り込みを使用して、アプリケーション向けのテストを準備する。

HO-2.1.7 （H3）アプリケーションを使用しながら、いくつかのモバイルデバイスの割り込みを行うことによって、そのアプリケーションをテストする。

MAT-2.1.8 （K3）デバイスの機能に対してアプリケーションによって要求されるアクセス許可の変更に関するテストを準備する。

HO-2.1.8 （H3）アプリケーションのアクセス許可管理をテストする。方法として、フォルダおよびセンサーに対して必要なアクセス許可をインストール時に「拒否」に設定するか、またはインストール後に変更した場合に、そのアクセス許可の付与／拒否への変更を行い、振る舞いを観察する。

MAT-2.1.9 （K3）アプリケーションがデバイスの電力消費に及ぼす影響と、バッテリーの状態がアプリケーションに及ぼす影響を検証するためのテストを準備する。

HO-2.1.9 （H3）バッテリーのさまざまな電力レベルでアプリケーションをテストして、消費データと、バッテリー充電率が低または空の場合の性能を把握する。

2.2 アプリケーションとデバイスソフトウェアの連携性のテスト

MAT-2.2.1 （K3）テスト対象システムからの通知の処理に関するテストを準備する。

HO-2.2.1 （H2）アプリケーションがフォアグラウンドで動作している場合とバックグラウンドで動作している場合で、通知受信の影響をテストする。通知設定の変更がアプリケーションの機能性に及ぼす影響をテストする。

MAT-2.2.2 （K2）クイックアクセスリンクの意図した機能性を、テストでどのように検証できるかを説明する。

HO-2.2.2 （H3）ショートカット／クイックアクセスの機能性に関してアプリケーションをテストする。

MAT-2.2.3 （K3）オペレーティングシステムによって提供されるユーザー設定がアプリケーションに及ぼす影響に関するテストを準備する。

HO-2.2.3 （H3）アプリケーションを実行しながら、オペレーティングシステムによって提供される基本設定の入力値オプションを変更してテストする。

MAT-2.2.4 （K2）ネイティブアプリケーション、Web アプリケーション、およびハイブリッドアプリケーションの間で異なるテスト要件を識別する。

HO-2.2.4 （H0）（オプション）アプリケーションのテスト要件をアプリケーションタイプごとに識別する。

MAT-2.2.5 （K1）複数のプラットフォームまたはオペレーティングシステムバージョンで利用可能なアプリケーションのテスト要件を想起する。

MAT-2.2.6 （K1）他のアプリケーションとの共存性および相互運用性に関するテスト要件を想起する。

2.3 さまざまな接続方法のテスト

MAT-2.3.1 （K2）ネットワーク間、Bluetooth 使用時、機内モードへの切り替え時を含む、接続性テストを要約する。

HO-2.3.1 （H0）（オプション）データをサーバーに転送中、および利用可能な信号強度に基づいてデバイスの接続を Wi-Fi とモバイルデータ通信間で切り替えた場合について、アプリケーションをテストする。

2.1 デバイスハードウェアとの互換性テスト

2.1.1 デバイスの機能のテスト

さまざまなタイプのデバイスにさまざまな機構が搭載されており、大量のデバイスに対して互換性テストを実施する必要がある。このためには、対象デバイスの優先度付けをテスト用に行う必要がある。優先度付けに際しては、1.1 節で説明されている市場データを使用して、対象市場にとって最も適切なデバイスポートフォリオを選択する。デバイスポートフォリオ選択においては、通常、市場カバー率、コスト、およびリスクを考慮する。

アプリケーションは、次の特性を備えるさまざまなタイプのデバイスにインストールできる。

- 異なる方法による電源オフ
- 異なる方法によるナビゲーション
- ハードウェアキーボードとソフトウェアキーボードの使用
- 次に示すさまざまなハードウェア機能：
  - 無線
  - USB
  - Bluetooth
  - カメラ
  - スピーカー
  - マイクロフォン
  - ヘッドフォンアクセス

これらのいずれの機能もアプリケーションの動作に悪影響を及ぼしてはならない。

デバイス機能には多くのバリエーションがあり、同一製造元によって製造されている異なるデバイスモデル間でも異なる場合がある。一般的に、市場セグメント間での差別化のために使用され、時間経過に伴い、短期間で変更される可能性がある。例えば、最近のほとんどのハイエンドおよびミッドレンジのデバイスには指紋センサーが搭載されているが、ローエンドのデバイスには搭載されていない。この特性は、時間経過とともに変化する。わずか数年前には、どのモバイルデバイスにも指紋センサーは搭載されていなかった。 この変更性に対応するために、テスト担当者はデバイスとユーザーが期待する機能について、明確に理解する必要がある。テスト担当者は、デバイスポートフォリオを作成し、それに従って対応するテストを設計する必要がある。

期待される機能に関してアプリケーションが正しく動作するかどうかをテストするだけでは十分でない場合が多い。さらに、特定の機能が搭載されていない場合でも、アプリケーションが期待通りに動作することをテストする必要がある。例えば、前面カメラと背面カメラの使用をサポートするアプリケーションは、複数のカメラを搭載するデバイス、1 つのカメラのみを搭載するデバイス、またはカメラをまったく搭載していないデバイスのいずれにインストールされ実行された場合でも、クラッシュしてはいけない。

2.1.2 さまざまなディスプレイのテスト

デバイスのディスプレイには、さまざまな画面サイズ、ビューポートサイズ、アスペクト比、インチ当たりピクセル数（ppi）で表される解像度、インチ当たりのドット数（dpi）が存在する。デバイスの多様性を考慮して、優先度付けを行う必要がある。作成するテストでは、さまざまなデバイスで、

対象市場で最も使用されているさまざまな画面サイズ、解像度、アスペクト比を使用してユーザーインターフェースを操作する必要がある。

さまざまなディスプレイでテストを実行して、次の項目を確認する必要がある。

- アプリケーションは、使用されている画面密度およびサイズに従って、すべてのユーザーインターフェース要素を拡大／縮小する。
- ユーザーインターフェース要素の表示は重ならない。
- 使用性やタッチ操作の問題は発生しない。
- 高 dpi／ppi でも、問題となるような縮小は発生しない。

2.1.3 デバイス温度のテスト

モバイルデバイスはデバイス温度の上昇に対して、デスクトップコンピューターとは異なる反応をする。

モバイルデバイスは、バッテリー充電、過度の負荷、バックグラウンドでのアプリケーション実行、モバイルデータ通信や Wi-Fi、GPS の連続使用などのさまざまな理由によってオーバーヒート状態になる場合がある。

オーバーヒート状態になると、デバイスは温度上昇を抑え、バッテリーレベルを保持するための動作を行う。この動作には、CPU 周波数を下げる、メモリを解放する、システムの一部をオフにする、などがある。

これが発生すると、アプリケーションの機能性も影響を受けるため、テストを計画する際には考慮する必要がある。テストを設計する際には、長期間にわたり連続して熱を発生させるために、エネルギーを大量に消費させるように設計する必要がある。テスト対象ソフトウェアは、この状態で予期されない振る舞いを示さない必要がある。

2.1.4 デバイス入力センサーのテスト

モバイルデバイスは、センサーからさまざまなタイプの入力を受け取る。これらのセンサーには、例えば、GPS、加速度計、姿勢制御装置、3 軸磁力計を使用するものや、気圧、温度、湿度、心拍、光、または非接触の入力に反応するものもある。

さまざまなデバイス入力センサーのテストでは、次の項目をチェックする。

- アプリケーションは、利用可能な各センサーに対して、意図したとおりに動作する。例えば、アプリケーションは、（ウォーキング時のような）回転や前後の動きなどのさまざまなタイプの動きに対してテストされる必要がある。
- 外部の明るさに反応する機能は、さまざまな明るさ条件下で正しく反応する。
- サウンドの入力と出力の機能は、ソフトおよびハードのボリュームボタン、マイクロフォン、有線および無線のスピーカー、周囲のさまざまなサウンド条件と連動して正しく反応する。
- 位置情報は、次の条件下で正確である（日本語訳注：常に正確というわけではなく、信号品質などには影響を受ける）。
    - GPS のオン／オフの切り替え。
    - さまざまな GPS 信号品質。
    - アプリケーションが他のさまざまな位置決定方法（Wi-Fi、携帯電話基地局、または手動による場所入力）にフォールバックする必要がある場合。

2.1.5 さまざまな入力方法のテスト

さまざまなデバイスへの入力方法のテストでは、次の項目をチェックする。

- モバイルデバイスでさまざまなソフトキーボードのインストールが可能な場合、少なくとも、主要なデバイス製造元が提供しているキーボードおよび広く普及しているキーボードでアプリケーションは動作できる。
- アプリケーションでは必要に応じて、適切なレイアウトおよびキーのデフォルトキーボードが確実にポップアップする。
- ユーザーが 1 本以上の指をタッチ画面に置いた場合、アプリケーションはそのパターンを特定のジェスチャーまたはコマンドとして解釈する。典型的なジェスチャーには、プレス／タッチ、ダブルタッチ、マルチタッチ、スワイプ、タップ、ダブルタップ、ドラッグ、ピンチオープン／ピンチクローズなどがある。
- アプリケーションの各画面は、画面で適切なジェスチャーや他の入力手段に対して正しく反応し、非サポートのジェスチャーや入力はすべて無視する必要がある。
- アプリケーションによって使用されるカメラは、イメージやビデオのキャプチャ、バーコードや QR コード、その他の文書のスキャン、距離の計測を行える必要がある。
- 前面カメラと背面カメラが利用できる場合、適切なカメラがデフォルトでオンになる。例えば、ビデオチャットで前面カメラがデフォルトでオンになる必要がある場合、アプリケーションについて、カメラ入力を使用する場合と使用しない場合をテストする必要がある。また、2 台のカメラではなく、1 台（前面または背面）のみ利用できる場合に、テスト対象のソフトウェアが正しく動作することをテストで確認する必要がある。これは特に、テスト対象ソフトウェアが 1 台の特定のカメラを使用し、そのカメラが搭載されていない場合に当てはまる。

2.1.6 画面の向きの変更のテスト

向きの変更はモーションセンサーを使用して検出され、横向きと縦向きのモード切り替えが発生する。この際、必要に応じて、ユーザーインターフェースでレイアウトが変更される。

画面の向きを変更した後のテストでは、次の項目をチェックする。

- 横向きと縦向きのモード切り替えが発生した場合に、使用性と機能的な振る舞いは想定通りである。
- アプリケーションは操作中のその状態を維持する。
- 入力データフィールドには、入力済みデータが保持される。
- 出力データフィールドには、使用中のセッションを維持しながら、同じデータが表示される。

画面の向きを変更した後のテストでは、単一の切り替えだけに重点を置かないようにする必要がある。これは、描画や状態の問題は、必ずしも単一の変更後に現れるとは限らないからである。このため、テストでは、横向きと縦向きのモード切り替えを複数回連続して行う必要がある。

ユーザーインターフェースのさまざまな状態で向きを複数回切り替えるテストを、データの有無を考慮しながら、設計する必要がある。アプリケーションは期待通りに動作し、データの喪失や変更が発生せずに状態が維持される必要がある。

2.1.7 典型的な割り込みのテスト

デバイス割り込みの一般的なタイプとしては、着信、メッセージ、充電開始、メモリひっ迫、その他の通知などがある。ユーザー起因の割り込みが、アプリケーションの切り替え、アプリケーション実行時のデバイスのスタンバイ移行などのアクションによって発生する。

割り込みのテストでは、次の項目をチェックする。

- アプリケーションは、前述のすべての割り込みを、アプリケーションの振る舞いに悪影響を及ぼすことなく、正しく処理する。
- アプリケーションは、発生した割り込みに関係なく、その状態やデータ、セッションを維持しながら、機能を正しく続行する。
- 通知をブロックする「おやすみ」モードがデバイスに備わっている場合、さまざまな条件がアプリケーションで正しく使用されることを確認する必要がある。これらのテストは、「おやすみ」モードを長期間にわたってオンにしていた後にオフに切り替えた状況でも実行する必要がある。結果として、多数の通知が一度に受信される。
- テストは、アプリケーションの使用中に割り込みを発生させるように設計する必要がある。このようにすることで、割り込みが悪影響を及ぼさないことを確認できる。例えば、アプリケーションの使用時に電話の着信に応答し、電話を切った後で、割り込み時点の状態に戻ることを確認する。

2.1.8 デバイス機能に対するアクセス許可のテスト

アプリケーションは、連絡先や写真などのさまざまなフォルダ、およびカメラやマイクロフォンなどのセンサーにアクセスする必要がある。アクセスがインストール時に拒否されるか、またはインストール後に拒否に変更されると、アプリケーションの振る舞いに影響が及ぶことがある。

アクセス許可のテストでは、次の項目をチェックする。

- アプリケーションは、アクセス許可の権限が不足している場合でも動作できる。ユーザーにこれらのアクセス許可の付与を求め、原因不明で失敗することはない。
- アクセス許可は、アプリケーションの機能性に関連するリソースに対してのみ要求される。関連しないリソースに対する広範なアクセス許可は付与されない。
- アクセス許可が削除されるか、インストール時に拒否されても、アプリケーションの機能性は正しく反応する。
- アプリケーションによって発行されるアクセス要求はいずれも、正確で、正当である。

アクセス許可をテストするために、テスト担当者は、アプリケーションが各アクセス許可を必要とする理由と、アクセス許可が削除されるか、インストール時に拒否された場合の機能性への影響を把握する必要がある。テストは、インストール時にアクセス許可が拒否された場合と、インストール後に付与された場合用に設計する必要がある。

2.1.9 電力消費とバッテリーの状態のテスト

電力消費とバッテリーの状態のテストでは、次の項目をテストする。

- バッテリーの状態と電力消費に関連する欠陥。
- 低電力時およびバッテリー切れ時のデータ整合性。
- アプリケーションの通常実行時、高負荷時、低負荷時の電力消費。
- アプリケーションがバックグラウンドで動作している場合の電力消費。

これらのテストは、長期間にわたって割り込みが発生しない状況で実行する必要があるため、入念に計画する必要がある。例えば、アプリケーションをバックグラウンドまたはフォアグラウンドで実行したまま、デバイスを使用せずに、そのまま放置することが必要な場合がある。バッテリーの消費パターンに関する情報を抽出するために、ログアナライザーなどのツールが必要である。

2.2 アプリケーションとデバイスソフトウェアの連携のテスト

2.2.1 通知のテスト

オペレーティングシステムは、さまざまなメカニズムを使用して通知を表示する。状況によっては、電力消費を最適化するために、オペレーティングシステムは通知の表示を遅らせたり、まったく表示しなかったりすることがある。次のテスト条件を考慮する必要がある。

- 特に低バッテリー条件下で、アプリケーションがフォアグラウンドまたはバックグラウンドで動作している場合に受信された通知の正しい処理。
- 通知からアプリケーションのコンテンツに直接アクセスできる場合（つまり、アプリ自体を開かずに）、以降において、アプリケーションでユーザー操作が提供される必要がある。例えば、ユーザーが通知に応答した場合、以降において、その応答にアプリケーション内からアクセスできる必要がある。
- 通知からアプリケーションにアクセスでき、アプリケーションの対応するページへのディープリンクが通知に含まれる場合、ホーム画面ではなく、そのページが表示される必要がある。

2.2.2 クイックアクセスリンクのテスト

Android のアプリケーションショートカットや iOS の触覚タッチおよび 3D Touch などのクイックアクセスリンクが、テスト対象ソフトウェアで提供されている場合がある。これらの機能を使用すると、アプリケーション全体を実際に起動することなく、ホーム画面からアプリケーションの機能のサブセットを実行できる。

次のテスト条件を考慮する必要がある。

- 一部の機能がオペレーティングシステムの特定バージョンでのみ利用できる場合、インストール先のオペレーティングシステムでのその機能の提供の有無に関係なく、テスト対象システムは正しく振る舞う必要がある。
- クイックアクセスリンクで実行されるアクションは、アプリケーションが開かれた際に、アプリケーションに正しく反映される必要がある。

2.2.3 オペレーティングシステムで提供されるユーザー設定のテスト

オペレーティングシステムによってユーザーに提供されているすべての設定をテストする必要がある。アプリケーションで特定の設定が考慮されていない場合、ユーザーの操作性に悪影響が及ぶ可能性がある。例えば、デバイスが消音に設定されている場合、アプリケーションは音声を出力してはならない。

次のテスト条件を考慮する必要がある。

- ユーザーは、サウンド、明るさ、ネットワーク、低電力モード、日付と時刻、時間帯、言語、アクセスタイプ、通知などの一般的な設定オプションを変更できる。
- アプリケーションは、設定されている基本設定に従って振る舞う。

2.2.4 さまざまなタイプのアプリケーションのテスト

モバイルアプリケーションのタイプに応じて、固有のテストを実行する場合がある（1.4 節参照）。次のテスト条件を考慮する必要がある。

- ネイティブアプリケーションの場合：
  - デバイス互換性
  - デバイス機能の使用
- ハイブリッドアプリケーションの場合：
  - アプリケーションとデバイスネイティブ機能との連携
  - 抽象化レイヤの使用に起因する潜在的な性能の問題
  - 対象のプラットフォームのネイティブアプリケーションと比較した使用性（外観と操作感）
- Web アプリケーションの場合：
  - 普及しているさまざまなモバイルブラウザに対するアプリケーションのブラウザ間互換性を判断するためのテスト
  - さまざまな JavaScript エンジンによって機能性に影響を及ぼさないこと
  - OS 機能の使用（例えば、日付の選択や適切なキーボードの起動）
  - 対象のプラットフォームのネイティブアプリケーションと比較した使用性（外観と操作感）

2.2.5 複数のプラットフォームおよびオペレーティングシステムバージョンとの互換性のテスト

多くのソフトウェア企業は、アプリケーションを複数のオペレーティングシステムでサポートしている。各モバイルオペレーティングシステムには独自の制限があり、アプリケーションをテストする際には考慮する必要がある。テスト担当者は、テスト対象の各プラットフォームの特性を把握して、アプリケーションが意図したとおりに動作し、しかもプラットフォームの外観と操作感に準拠していることを確認する必要がある。

次のテスト条件を考慮する必要がある。

- 割り込み、通知、および最適化（例えば、省電力）の処理
- マルチプラットフォームアプリケーションが一部のコードを共有する場合、またはクロスプラットフォーム開発フレームワークを使用して作成された場合に機能性が損なわれないこと。アプリケーションがコードを共有していない場合、異なる 2 つのアプリケーションをテストしていることと同様であり、すべてをテストする必要があることに注意する。
- プラットフォームが複数のオペレーティングシステムバージョンをサポートする場合、下位互換性のテスト。
- プラットフォームに対して新規追加された機能または変更された機能のテスト。例えば、Android で導入された Doze フレームワークについては、このフレームワークをサポートするオペレーティングシステムバージョンおよびサポートしないオペレーティングシステムバージョンでテストする必要がある。

2.2.6 デバイス上の他のアプリケーションとの相互運用性と共存性のテスト

同じデバイスにインストールされたアプリケーションは、相互に連携することが極めて一般的である。典型的な例としては、連絡先アプリケーションと電子メールアプリケーションの関係が挙げられる。

次のテスト条件を考慮する必要がある。

- テスト対象アプリケーションと使用するアプリケーションの間のデータ転送は正しい。
- 使用するアプリケーション内に格納されるユーザーデータが破損することはない。
- 競合する振る舞い。例えば、電力消費を抑えるために GPS をオフにするアプリケーションと、GPS を自動的にオンにするアプリケーションが共存する可能性がある。

市場には大量のアプリケーションが存在するため、全アプリケーションに対して共存性をテストすることは現実的に不可能である。それでも、そのような潜在的な問題は、それらのリスクに従って考慮およびテストされる必要がある。

2.3 さまざまな接続方法のテスト

モバイルデバイスは、さまざまな方法を使用してネットワークに接続できる（1.5 節参照）。そのような方法には、2G、3G、4G、5G などのモバイルデータ通信、Wi-Fi および NFC や Bluetooth などのその他のワイヤレス接続タイプがある。

接続性テストを実行する際には、次の代替策を考慮する必要がある。

- デバイスエミュレーター／シミュレーターを使用することで、さまざまな接続をシミュレーションでき、一部のリモートデバイスアクセスサービスプロバイダーはそのような機能を提供している。ただし、エミュレーター／シミュレーターの能力には、限界がある。
- さまざまな接続タイプをサポートする独自のモバイルネットワークをセットアップして、帯域幅操作を適用する。これは、極めて高価な代替策である。
- フィールドテストは潜在的に費用対効果の高い代替策であるが、テストの再現性に関しては限定的である。

実世界における使用では、接続方法は異なる。ユーザーは 1 つの特定のモードを使用して連続して接続できるか、Wi-Fi からモバイルデータ通信（例えば、アプリケーションを使用しながら自宅を出るとき）など、モード間で切り替えできる。ユーザーは、さまざまな Wi-Fi／モバイルデータ通信およびバージョン間、GSM セル間で切り替えることができる。移動中には、ネットワークをまったく利用できないデッドスポットに遭遇することさえある。さらに、例えば機内モードに切り替えるなどして、ユーザーは意図的に切断することができる。

接続性テストでは、次のテスト条件を考慮する必要がある。

- さまざまな接続モードでのアプリケーションに求められる機能性。
- モードを切り替えても、予期しない振る舞いやデータ喪失が発生しない。
- ネットワーク接続が制限されているか利用できない場合、または帯域幅が狭い場合に、機能が制限されていることを明確に示す情報がユーザーに提供される。メッセージには、制限される機能とその理由が含まれる必要がある。

3 モバイルアプリケーションで一般的なテストタイプとテストプロセス	200 分

キーワード

異常終了、アクセシビリティ、探索的テスト、フィールドテスト、ヒューリスティック、設置性、性能効率性、性能テスト、リリース後テスト、セキュリティテスト、セッションベースのテストマネジメント、ストレステスト、テストレベル、テストプロセス、テストピラミッド、ツアー、使用性ラボ、使用性テスト、コードインジェクション、リリース後テスト

「モバイルアプリケーションで一般的なテストタイプとテストプロセス」の学習の目的

3.1 モバイルアプリケーションに適用可能な一般的なテストタイプ

MAT-3.1.1 （K3）モバイルアプリケーション用の設置性テストを準備する。

MAT-3.1.2 （K3）モバイルアプリケーション用のストレステストを準備する。

MAT-3.1.3 （K2）モバイルアプリケーションに関連するセキュリティ問題の例を提示する。

MAT-3.1.4 （K1）モバイルアプリケーションの時間とリソースの振る舞い上の考慮事項を想起する。

MAT-3.1.5 （K3）モバイルアプリケーションの使用性テストを準備する。

HO-3.1.5 （H2）セッションベースのテストマネジメントを用いて、アプリケーションの使用性テストのためのツアー、ニーモニック、またはヒューリスティックを選択する。

注：HO-3.1.5 と HO-3.3.1、HO-3.3.2 と HO-3.3.3 は 1 つにまとめることが可能。

MAT-3.1.6 （K1）モバイルアプリケーションのデータベーステストで必要なテストタイプを認識する。

MAT-3.1.7 （K2）モバイルアプリケーションの国際化（グローバル化）およびローカライズのテストで要求されるテストを要約する。

MAT-3.1.8 （K2）モバイルアプリケーションテストでのアクセシビリティテストの必要性を要約する。

3.2 モバイルアプリケーションに適用可能な追加のテストレベル

MAT-3.2.1 （K2）フィールドテストなどの追加のテストレベルと、モバイルアプリケーションを効果的にテストするために必要な追加の活動を説明する。

MAT-3.2.2 （K2）アプリケーションを公開するためのアプリケーションストアの承認を得るために必要なテストを説明する。

3.3 経験ベースのテスト技法

MAT-3.3.1　（K1）探索的モバイルテストにおけるセッションベースのテストマネジメント、ペルソナ、ニーモニックを想起する。

HO-3.3.1　（H2）セッションベースのテストマネジメントを用いてアプリケーションをテストするために、モバイルアプリケーションテストに固有のニーモニック（またはその一部）を選択する。

注：HO-3.1.5 と HO-3.3.1、HO-3.3.2 と HO-3.3.3 は一緒に実行することが可能。

MAT-3.3.2　（K2）モバイルアプリケーションのテスト用に探索的技法などのツアーやヒューリスティックの使用を説明する。

HO-3.3.2　（H2）モバイルアプリケーションのテスト用にモバイル固有のヒューリスティックを選択する。

注：HO-3.1.5 と HO-3.3.1、HO-3.3.2 と HO-3.3.3 は 1 つにまとめることが可能。

MAT-3.3.3　（K3）モバイルアプリケーションのテスト用にモバイル固有のツアー（フィーチャーツアーなど）を使用する。

HO-3.3.3　（H2）モバイルアプリケーションのテスト用にモバイル固有のツアーを選択する。

注：HO-3.1.5 と HO-3.3.1、HO-3.3.2 と HO-3.3.3 は 1 つにまとめることが可能。

3.4 モバイルテストプロセスと手法

MAT-3.4.1　（K2）[ISTQB_CTFL_2018]で説明されているテストプロセスをモバイルアプリケーションのテストに対する要求に一致させる。

MAT-3.4.2　（K2）モバイルアプリケーションのテストに固有の各テストレベルでのテスト手法を説明する。

3.1　モバイルアプリケーションに適用可能な一般的なテストタイプ

3.1.1　設置性テスト

テスト担当者は、アプリケーションのインストール、アップデート、およびアンインストールに重点を置く必要がある。アプローチとしては次のようなものがある。

- アプリケーションストア

インストールプロセスは、アプリケーションのユーザーによって異なる可能性がある。アプリケーションは、Google Play Store や Apple の App Store などのマーケットプレイスからインストールできる場合がある。エンタープライズアプリケーションの場合、リンクまたは HockeyApp や App Center などの配布サービスを介して、インストールテストを実行することが要求される場合がある。

- サイドローディング（アプリケーションのコピーおよびインストール）

オペレーティングシステムによっては、アプリケーションをモバイルデバイスにコピーして、そのファイルからアプリケーションをインストールできる。

- デスクトップアプリケーション

Apple iTunes（iOS 用）や Android App Installer などのデスクトップアプリケーションを使用して、アプリケーションをスマートフォンにインストールできる。この場合、対象のアプリケーションをこれらのデスクトップアプリケーション内にダウンロードし、有線接続を介してスマートフォンにイン

ストールする。これらのデスクトップアプリケーションのほとんどに、アプリケーションのアンインストール機能が備わっている。

インストールは、次の方法を使用して実行できる。

- Wi-Fi またはモバイルデータ通信経由の OTA（Over-the-Air）
- データケーブル

次を含むいくつかのテスト条件を考慮できる。

- 内部メモリおよび外部メモリ（サポートされている場合）でのインストール、アンインストール、およびアップグレード。
- 「アプリケーションデータを保持」オプションを選択してアプリケーションをアンインストールした後の再インストール。
- 「アプリケーションデータを保持」オプションを選択せずにアプリケーションをアンインストールした後の再インストール。
- インストールまたはアンインストールのキャンセルまたは割り込み。例えば、それらの処理中におけるモバイルデバイスのシャットダウンまたはインターネットからの切断。
- インストール、アンインストール、およびアップグレードの処理のキャンセルまたは割り込み後のそれらの処理の再開。
- アクセス許可関連のテスト。例えば、アドレス帳を使用するためのアクセス許可を必要とするアプリケーションがある。この重要なテストでは、ユーザーがアクセス許可を拒否した場合のアプリケーションの振る舞いを検証する必要がある。例えば、対応するメッセージがユーザーに表示されるか？
- アプリケーションをアップデートして、データ喪失が発生していないことの確認。

一部のアプリケーションは、Jailbreak（iOS）または root 化（Android）が行われており、ユーザーに管理者権限を付与可能なデバイスを必要とする。ほとんどのプラットフォームプロバイダーは、法律上の問題が発生する可能性があるため、Jailbreak および root 化の行為をサポートしていない。Jailbreak および root 化の行為を必要としないアプリケーションは、Jailbreak および root 化の行為が行われたデバイス用のテストを必要としない可能性がある。

3.1.2 ストレステスト

ストレステストは、過度に負荷がかかった条件でのアプリケーションの性能効率性を判定することに重点を置く。ここで説明するストレステストは、モバイルデバイスのみを対象にする。バックエンドのストレステストについては、ISTQB® Performance Testing シラバス（[ISTQB_CTFL_PT_2018]）で説明されているため、詳細については、そのシラバスを参照されたい。

ストレステストで考慮するテスト条件のいくつかを次に示す。

- 高 CPU 使用率
- メモリ不足
- 少ないディスクスペース
- バッテリー負荷
- 故障
- 貧弱な帯域幅

- 極めて多量のユーザー操作（このテスト条件では、実世界のネットワーク条件をシミュレーションすることが必要な場合がある）

これらのストレスフルな条件の一部は、Monkey などのツールを使用して作成できる。Monkey はコマンドラインツールで、ADB シェルコマンドライン[URL3]で実行することも、手動で実行することもできる。例えば、巨大なファイルや他のアプリケーションを使用して、高 CPU 使用率やメモリ消費の条件を作成できる。

3.1.3 セキュリティテスト

セキュリティテストは複雑なトピックであるため、ISTQB®では個別のスペシャリストシラバス[ISTQB_CTAL_SEC_2016]を提供している。モバイルデバイスでの主要なセキュリティの問題を次に示す。

- デバイス上の機密データへのアクセス。
- 暗号化されていない情報転送または安全でないストレージ。

セキュリティテストで考慮するテスト条件のいくつかを次に示す。

- コードインジェクションおよびオーバーフローを引き起こす入力。
- 転送済みデータの暗号化。
- ローカルに格納されているデータの暗号化。
- 使用後または異常終了後の一時データの削除。
- パスワードフィールドのテキストのクリア。

Open Web Application Security Project（OWASP）の上位 10 件のモバイル関連脆弱性もテストの対象にする必要がある[URL2]。

3.1.4 性能テスト

ユーザーがインストールしたアプリケーションのレスポンスが遅い（例えば 3 秒超）場合、アンインストールして別の代替となるアプリケーションをインストールする場合がある。時間とリソースの消費の側面は、アプリケーションの重要な成功要因であり、性能テストを実行して、これらの側面を測定する必要がある。

性能効率性は、単体のデバイス上でテストする必要があり、さらには、バックエンドシステムや他のモバイルデバイスとの連携についてもテストする必要がある。

システム全体の性能テストは、テスト戦略での定義に応じて実行する必要があり、モバイル固有のものではない。詳細については、性能テストに関する ISTQB®スペシャリストシラバス[ISTQB_CTFL_PT_2018]を参照されたい。

アプリケーション自体の性能テストには、最も重要なワークフローの時間測定法を含む必要がある。オンラインバンキングでのワークフローの例には、「ログイン」、「住所の変更」、「PIN および TAN による振込」などがある。テスト担当者は、この時間測定法を類似のアプリケーションと比較する必要がある。

時間測定法による測定に加えて、ユーザーが認識する性能も考慮する必要がある。ユーザーエクスペリエンスは、特定の機能が完了するまでにユーザーが待機できる時間の長さに大きな影響を及ぼす可能性がある。

3.1.5 使用性テスト

使用性はモバイルアプリケーションにとって極めて重要である。データによると、使用性と性能が貧弱である場合、大多数のユーザーはインストール中のアプリケーションをわずか数分でアンインストールしている。詳細については、[URL4]を参照されたい。

このため、ユーザーエクスペリエンス（UX）設計では、アプリケーションが使用される予定のプラットフォームの外観と操作感を考慮することが推奨される。選択するプラットフォームに対するユーザーの期待に UX がそぐわない場合、極めて悪い影響を及ぼす可能性がある。従って、テスト担当者は、使用されるプラットフォームの外観と操作感に注意する必要がある。

テスト担当者は、利用可能なさまざまなヒューリスティックやテストツアーを使用して、使用性テストを実行できる。ペルソナを考慮することも、使用性テストに役立つ。必要に応じて、この目的のために、ユーザビリティラボも使用できる。

プロジェクトでは、使用性テストで識別された発見事項は、欠陥ではなく単なる発見事項であることが多い。テスト担当者は、発見事項をチーム、プロダクト所有者または同等のステークホルダーに説明する能力を備えている必要がある。十分な使用性を達成するために、アプリケーションは次のような要素を備える必要がある。

- 自明で直感的である。
- ユーザーの誤りを許容する。
- 文言表現と振る舞いを一致させる。
- プラットフォームの設計ガイドラインを遵守する。
- 各画面サイズとタイプで、必要な情報の表示とアクセスが可能である。

詳細については、使用性テストに関する ISTQB®スペシャリストシラバス[ISTQB_FLUT_2018]を参照されたい。

3.1.6 データベーステスト

多くのアプリケーションは、フラットファイルやデータベースなどのさまざまなデータストレージメカニズムを使用して、データをローカルに格納する必要がある。モバイルアプリケーションのデータベーステスト用に考慮する必要のあるテスト条件のいくつかを次に示す。

- データストレージ問題の検証：
  - 同期
  - アップロード競合
  - データセキュリティ
  - データに関する制約
  - CRUD（作成（Create）、読み込み（Read）、アップデート（Update）、削除（Delete））の機能性
  - 検索
- デバイス（例えば、連絡先）またはサードパーティアプリケーション（例えば、写真、動画、メッセージ）によって提供されるデータに対するデータ統合テスト。
- データをデバイスに格納する性能。

3.1.7 グローバル化とローカル化のテスト

アプリケーションの国際化（I18N）／グローバル化テストには、異なる地域、日付／数値／通貨の形式、実際の文字列と疑似文字列の交換に対するアプリケーションのテストが含まれる。

ローカル化（L10N）テストには、特定の地域向けにローカル化された文字列、画像、およびワークフローを備えるアプリケーションのテストが含まれる。例えば、ロシア語とドイツ語の単語は、他の言語の単語よりもかなり長くなる場合がある。モバイルデバイスはさまざまな画面サイズや解像度を備えているため、画面サイズに制限がある場合、翻訳された文字列で問題が発生する可能性がある。これらの問題は、標準のグローバル化／ローカル化テストとして確認する必要がある。

確認する必要のある極めて重要な側面は、年 – 月 – 日または日 – 月 – 年などの日付の形式である。

3.1.8 アクセシビリティテスト

アクセシビリティテストは、身体的な制約を持つ人を含むユーザーが、どの程度容易にコンポーネントやシステムを利用できるか判定するために行う。モバイルアプリケーションの場合、これはデバイスのアクセスビリティ設定を使用し、設定ごとにアプリケーションをテストすることによって実施できる。

アクセシビリティガイドラインがプラットフォームベンダーから提供されており、これらを参照する必要がある。例えば、Google [URL5]と Apple [URL6]の両方がそれぞれのプラットフォームのアクセシビリティガイドラインを公開している。アクセシビリティを必要としている人からのフィードバックも役に立つ。

モバイル Web に対しては、アクセシビリティガイドラインが W3C から公開されており、考慮する必要がある[URL7]。

3.2 モバイルアプリケーションに適用可能なその他のテストレベル

[ISTQB_CTFL_2018]で説明されているコンポーネントテストから受け入れテストに至る一般的なテストレベル以外に、モバイルアプリケーションテストでは、さらなるテストレベルも必要である。

3.2.1 フィールドテスト

モバイルアプリケーションによっては、フィールドテストを実施して、実ユーザーが期待する利用シナリオで正しく機能することを確認する必要がある。これには、Wi-Fi やモバイルデータ通信などのさまざまなネットワークおよびさまざまな通信技術タイプなどでのテストが含まれる。

フィールドテストでは、携帯電話基地局やネットワーク、Wi-Fi の使用、およびアプリケーション使用時の携帯電話基地局切り替えを行う必要がある。また、さまざまなダウンロード速度および電界強度を使用して実施する必要があり、不感地帯の処理も対象にする必要がある。

フィールドテストでは、入念な計画と、テストを実行するために必要なすべての項目の特定が必要である。項目としては、適切なデバイスタイプ、Wi-Fi、さまざまな事業者の料金プラン、適切なカバレッジを提供するために必要なさまざまな交通手段へのアクセスなどが含まれる。さらに、テストを実施するためのルートや交通手段、時間帯をスケジュールする必要がある。

また、アプリケーションの使用性もフィールドテストを実施する際に重要項目として対象にする必要がある。テストでは、温度や使用シナリオに関連する類似の条件などの環境要因も取り込む必要がある。

3.2.2 アプリケーションストアの承認のテストとリリース後テスト

アプリケーションを公開する前に、いくつかのチェックリストベースのテストに合格して、アプリケーションストアの承認を得られるようにする必要がある。リリースがアップグレードの場合、アップグレード関連のテストも実行する必要がある。

チェックリストは、オペレーティングシステム固有、ユーザーインターフェースデザイン、アプリケーションストアによって提供されているライブラリおよびAPIの使用などのガイドラインに基づくことが多い。

承認プロセスは、提出後にしばらく時間を要することがある。承認プロセスで問題が発見されると、新しいバージョンの発行が要求され、問題解決のためにさらに時間を要することがある。この状況について、プロジェクト計画とテストの期間に入念に考慮する必要がある。

さらなるテストレベルは、「リリース後」テストである。このテストレベルには、アプリケーションストアからのアプリケーションのダウンロードとインストールが含まれる。

3.3 経験ベースのテスト技法

3.3.1 ペルソナとニーモニック

ペルソナは空想上のキャラクターで、実際の顧客を表す。これらは、動機、期待、問題、習慣、および目標を持っており、実際のユーザーの振る舞いを模倣する必要がある場合に使用する。

ペルソナは、名前、性別、年齢、収入、学歴、および地域を持つこともできる。モバイルの環境では、他のアプリを使用したり、モバイルデバイスを1時間に複数回確認したりするかもしれない。また、他のデバイスや個人的な特徴も持つことがある。

ニーモニックは、記憶を助ける工夫である。テストの分野では、ニーモニックのすべての文字は、技法、テスト方法、またテストの焦点を表す。ニーモニックの一例として、SFiDPOT [URL8]を取り上げる。このニーモニックの各文字は、次の意味を持つ。

S – Structure（構造：例えば、ユーザーインターフェース要素、他のアプリケーションの要素、およびそれらの順序と呼び出し階層）

F – Function（機能：例えば、想定通りに機能が動作している、利用できる、要件に従って機能している、など）

i - input（入力：例えば、必要なすべての入力が利用でき、キーボード、センサー、カメラなどからの入力として想定通りに処理される）

D – Data（データ：例えば、データは、要件に定義されているように、（SDカードへも）格納、変更、追加、および削除を行うことができる）

P – Platform（プラットフォーム：例えば、アプリケーションのダウンロード用のストアなど、特定のオペレーティングシステム機能がデバイス設定に応じて利用できる）

O – Operations（運用：例えば、モバイルデータ通信とWi-Fi間の移動など、一般ユーザーの活動が可能である）

T – Time（時間：例えば時間帯、時刻、日付の設定と表示）

モバイルに特化しているニーモニックとヒューリスティックは、I SLICED UP FUN [URL9]である。このニーモニックの各文字は、次の意味を持つ。

I – Inputs（入力）

S – Store（ストア）

L – Location（場所）

I – Interactions and interruptions（連携と割り込み）

C – Communication（通信）

E – Ergonomics（人間工学）

D – Data（データ）

U – Usability（使用性）

P – Platform（プラットフォーム）

F – Function（機能）

U – User scenarios（ユーザーシナリオ）

N – Network（ネットワーク）

3.3.2 ヒューリスティック

ヒューリスティックアプローチは、問題の解決、学習、検出のための「経験則」を使ったアプローチであり、実践的な方法を採用する。これは、最適または完璧であることを保証するものではないが、当面の目標を達成するためには十分であると見なすことができる。

多くのヒューリスティックがモバイルテスト向けに存在する。ほとんどのニーモニックはヒューリスティックとして使用できるが、すべてのヒューリスティックがニーモニックであるとは限らない。

3.3.3 ツアー

ツアーは、アプリケーションを特定の観点および焦点から探索するために、探索的テストで使用される。アプリケーションの動作の仕組みを理解し、ワークフローのモデルを作成するために実行できる。ツアーは、フィールドテスト用の効果的な手法を提供する。

ツアーの例として、ランドマークツアーが挙げられる。このツアーでは、ユーザーは都市の著名なランドマークを訪れることによってツーリストの訪問を模倣する。次の表は、ツアーで行われる訪問を、モバイルテストで従うためのアナロジーとして使用できることを示している。

ランドマークツアーでの訪問	モバイルテスト用のアナロジー
旧市街	レガシーコード
ビジネス街	アプリケーションのビジネスロジック
ラッシュアワー	アプリケーションのスタートアップとシャットダウン
観光案内所	初心者が使用するアプリケーション部分
ホテル	おやすみモードでのみアクティブになるアプリケーション部分

セッションベースドテスト（3.3.4 参照）とツアーを組み合わせて、さらにヒューリスティックとニーモニックを使用することで、モバイルアプリケーションのテストの有効性を高めることができる。

次の表は、アプリケーションテスト用のツアーの好例と、テストのアイデアを与えるためにそれらがカバーする領域を示している。これらのいくつかは、[Kohl17]で説明されている。

アプリケーションテスト用ツアー	カバーされる対象
スーパーモデル	外観と操作感、使用性
ランドマーク	アプリケーションで最も重要な機能
サボタージュ	頑健性（堅牢性）
機能	新機能
シナリオ	ユーザーストーリーと組み合わせた、アプリケーションの全ワークフロー
接続性	Wi-Fi、GSM などの接続方法
場所	正しい言語、日付、数
明るさ	暗闇、戸外、赤い照明などのさまざまな照明条件での可視性
低バッテリー	低電力レベルによるアプリケーションでのデータ喪失
アプリケーションテスト用のその他のツアー	[Kohl17]でカバーされる対象
ジェスチャー	可能な限り全ジェスチャーを使用
向き	向きを変える
キャンセル	戻る
モーション	さまざまなタイプの動作を行う
場所	動き回る
接続性	接続タイプを変更するか、または移動して場所を変える
比較	他のタイプのデバイスと比較する
一貫性	画面、GUI の一貫性を確認する

3.3.4 セッションベースのテストマネジメント（SBTM）

セッションベースのテストマネジメント（SBTM）を使用すると、タイムボックス方式で探索的テストを管理できる。1 つのセッションは、次の 3 つのタスクで構成される。

- セッションセットアップ
- テスト設計と実行
- 問題の調査と報告

SBTM では通常、テスト目的を提供するテストチャーターが記載されているセッションシートを使用する。また、セッションシートは、実施されたテスト実行の活動を文書化するためにも利用される。

探索的テストは経験ベースのテスト技法であり、モバイルアプリケーションのテストにとって効果的な手法となり得る。経験ベースのテスト技法の詳細については、[ISTQB_CTFL_2018]を参照されたい。

3.4 モバイルテストプロセスと手法

3.4.1 テストプロセス

ISTQB®テストプロセスの主な活動の詳細については、[ISTQB_CTFL_2018]を参照されたい。これらの活動は、モバイルアプリケーションのテストにも適用できる。

モバイルテストには固有の側面があり、ISTQB®テストプロセスの一環として常に考慮する必要がある。

テストプロセスでの主な活動グループ	モバイルテストで考慮する必要のある典型的な領域	シラバスでの参照先
テスト計画作業	• テストする必要のあるデバイスの組み合わせ。 • テスト環境の一部としてのモバイルエミュレーターとモバイルシミュレーターの使用。 • モバイルアプリケーションテストでの特別な課題。 • モバイルアプリケーションテストで特に必要なテストタイプ。	• 4.3 節 • 1.7 節 • 3.2 節
分析と設計	• アプリケーションストア承認テスト。 • フィールドテスト。 • デバイス互換性。 • 使用するラボの種類。 • モバイルアプリケーションテストで特に必要なテストタイプ。	• 3.2.2 節 • 3.2.1 節 • 3.2 節
テスト実装とテスト実行	• フィールドテスト。 • ダウンロードと設置性のリリース後のテスト。 • 経験ベースの技法	• 3.2.1 節 • 3.1.1 節 [ISTQB_CTFL_2018]
テスト実装とテスト実行	• テストは、ユーザーインターフェースとアプリケーションストアのプラットフォームガイドラインに基づく。 • ガイドラインに基づくテストは通常、アプリケーションストア承認プロセス用に、プラットフォームプロバイダーによって実行される。	

	• 承認拒否の可能性を排除するために、アプリケーションをプラットフォームプロバイダーに提供する前に、アプリケーションプロバイダーとしてこれらを実行することが推奨される。	

3.4.2 テストアプローチ

モバイルアプリケーションのテストの活動は、開発者およびテスト担当者によって実行される。

テストレベル（すなわち、コンポーネントテスト、統合テスト、システムテスト、フィールドテスト、アプリケーションストア承認テスト、リリース後テスト、ユーザー受け入れテスト）ごとにテストの適切な深度を決定することは、優れた品質のプロダクトを提供するために重要である。テストレベルごとに必要なテストの深さは、アプリケーションのアーキテクチャ、複雑度、意図するユーザーなど、多くの要因によって決定される。

モバイル開発プラットフォームでは、さまざまなテストレベルでテストを支援するさまざまなツールを提供している。ツール自体と特定のレベルへの適用方法を理解することが、極めて重要である。例えば、プラットフォーム提供のフレームワークおよび計測用 API のメリットを活用する必要がある場合、モバイルシミュレーターそして／またはモバイルエミュレーターをコンポーネントテストレベルで使用できる。また、実際のデバイスを利用できない場合、モバイルシミュレーターそして／またはモバイルエミュレーターをシステムテストレベルで使用できる。こうすることで、機能性や、使用性およびユーザーインターフェースの一部の側面をテストできる。

さらに、ツールを早期に導入することは、デバイスが正しくセットアップされていることと、実行のためにすべての前提条件が予定通りに満たされることを確認するための重要なポイントとなり得る。

ユニットテストと統合テスト、（特にフィールドテストの段階では）手動テストも重要である。モバイルアプリケーションにとってテストピラミッドが逆さになることは極めて一般的である[Knott15]。これは、手動テストの量が大きくなる可能性があることを意味する。

4 モバイルアプリケーションのプラットフォーム、ツール、および環境	80 分

キーワード

フィールドテスト、プロキシミティベースドテスト、リモートテストラボ、エミュレーター、シミュレーター

「モバイルアプリケーションのプラットフォーム、ツール、および環境」の学習の目的

4.1 モバイルアプリケーション向け開発プラットフォーム

MAT-4.1.1 （K1）モバイルアプリケーション開発のために使用される開発環境を想起する。

4.2 一般的な開発プラットフォームツール

MAT-4.2.1 （K1）アプリケーション開発プラットフォームの一部として提供される一般的なツールのいくつかを想起する。

HO-4.2.1 （H1）スクリーンショットの取得、ログ記録の抽出、および入力イベントのシミュレーションを行うために、ソフトウェア開発キットに含まれるツールを使用する。

4.3 エミュレーターおよびシミュレーター

MAT-4.3.1 （K2）エミュレーターとシミュレーターの違いを理解する。

MAT-4.3.2 （K2）モバイルアプリケーションのテストにエミュレーターとシミュレーターを使用することについて説明する。

HO-4.3.2 （H1）シミュレーション／エミュレーションデバイスを作成して実行し、そのデバイスにアプリケーションをインストールして、テストの一部を実行する。

4.4 テストラボのセットアップ

MAT-4.4.1 （K2）テストラボをセットアップするためのさまざまな手法の違いを識別する。

4.1 モバイルアプリケーション向け開発プラットフォーム

さまざまなモバイルアプリケーション開発向けの統合開発環境（IDE）が、市場で利用できる。これらの IDE は、アプリケーションの設計、コーディング、コンパイル、インストール、アンインストール、監視、エミュレーション、ログ記録、およびテストを行う際に役立つ、さまざまなツールを備えている。

例えば、Android アプリの開発には Android Studio を、iOS アプリケーションの開発には Xcode を使用できる。これらの IDE は一般的な IDE とは異なり、モバイルプラットフォームのサポートが追加されている。

一部のクロスプラットフォーム開発フレームワークは、モバイルアプリケーションの開発にも利用できる。これらのフレームワークは複数のプラットフォーム上で動作し、プラットフォーム特有のコーディングを必要としない。

4.2 一般的な開発プラットフォームツール

一般的にソフトウェア開発キットは、アプリケーションの開発とテストに役立つさまざまなユーティリティを備えている。これらのユーティリティは広範な用途に対応しており、例えば、スクリーンショットの取得、ログ記録の抽出、任意のイベントおよび通知のデバイスへの送信、メモリや CPU の使用率などのさまざまなパラメータの監視、仮想デバイスの作成などが可能である。

これらのツールの例として、Android 用の Android Virtual Device（AVD）Manager、Android Debug Bridge（ADB）、Android Device Monitor、および iOS 用の Instruments がある。

4.3 エミュレーターおよびシミュレーター

4.3.1 エミュレーターおよびシミュレーターの概要

本シラバスでは、エミュレーターおよびシミュレーターの用語は、それぞれモバイルエミュレーターとモバイルシミュレーターを意味する。エミュレーターとシミュレーターという用語は時々同じものとして使用されることがあるが、それは誤りである。これらの用語の定義については、第 8 章の用語集を参照のこと。

シミュレーターは実行環境をモデル化するものであり、一方でエミュレーターはハードウェアをモデル化して、物理ハードウェアと同じ実行環境を使用するものである。シミュレーターでテストされるアプリケーションは、シミュレーターでは動作するが実デバイスでは動作しない専用のバージョンにコンパイルされる。このため、実 OS に依存しない。

一方、エミュレーターでの導入およびテスト用にコンパイルされたアプリケーションは、実デバイスでも使用可能な実際のバイトコードにコンパイルされる。

シミュレーターおよびエミュレーターは、一般的に開発環境に統合されており、アプリケーションのデプロイ、テスト、および監視を迅速に実施可能にするために、開発の初期段階で極めて役立つ。

シミュレーターは、実デバイスの代用品としてテストで使用されることもある。ただし、シミュレーターでテストされるアプリケーションは、配布されるアプリケーションとバイトコードレベルで異なるため、エミュレーター使用時に比べて制限が多い。

エミュレーターは、一部のテストで実デバイスの代用品として、テスト環境のコストを削減するためにも使用される。エミュレーターは、模擬対象のモバイルデバイスとは異なる動作をする可能性があるため、デバイスの完全な代用品となることはできない。さらに、（マルチ）タッチや加速度センサーなどの一部の機能がサポートされていない場合がある。これは、エミュレーターを実行するために使用されるプラットフォームの制限に一部起因する。

4.3.2 エミュレーターおよびシミュレーターの使用

モバイルテスト用にエミュレーターおよびシミュレーターを使用することは、さまざまな理由で役立つ。

各モバイルオペレーティングシステム開発環境には、独自のエミュレーターおよびシミュレーターが一般的にバンドルされている。またサードパーティ製のエミュレーターおよびシミュレーターも利用できる。

テスト担当者は、目的に合ったエミュレーターまたはシミュレーターを使用できる。エミュレーターまたはシミュレーターを使用するには、それらを起動し、必要なアプリケーションをそれらにインストールし、その後、実際のデバイス上であるかのようにアプリケーションをテストする必要がある。

エミュレーターおよびシミュレーターでは、通常、さまざまな利用時のパラメータを設定できる。これらの設定には、さまざまな通信速度や信号強度およびパケット損失率でのネットワークエミュレーション、方位の変更、割り込みの生成、GPS 位置情報データなどがある。これらの設定のいくつかは、グローバル GPS の位置情報や信号強度など、実デバイスでは再現が困難だったり、コストが高かったりする場合があり、極めて有用である。

インストールを目的としてエミュレーターに接続するには、Android 用の Android Debug Bridge（ADB）のようなコマンドラインツールの使用か、Xcode や Android Studio のような統合開発環境からの接続が求められる場合がある。

4.4 テストラボのセットアップ

モバイルテストラボのセットアップには次のアプローチがある。

オンプレミスラボ

オンプレミスラボでは、すべてのデバイス、エミュレーター、およびシミュレーターが敷地内に配置される。デバイスの選択は、デバイスの市場カバー率（Google や他の分析で分かる）、オペレーティングシステムとバージョン、画面サイズと画面密度、可用性とコスト、特別な機能、対象ユーザーにとっての重要性などのさまざまな要因に基づいて行うことができる。

オンプレミスラボのメリットには、特定のプロキシミティベースドテストや、バッテリー、タッチ、および強化されたセキュリティなどのセンサー固有の側面のための、デバイスの可用性が含まれる。

このタイプのラボのセットアップには、調達および保守されるデバイスに応じて、多額の予算が必要となる場合がある。その他の課題として、タイムリーな可用性や、異なる場所や環境でテストすることの困難さがある。

リモートテストラボ

これらのラボは、デバイスまたはネットワークが敷地内で物理的に利用できない場合のテストにとって重要かつ有益である。リモートデバイスアクセス（RDA）を使用すると、プロバイダーのデータセンターでホストされているさまざまなデバイスに、ネットワーク接続経由でアクセスできる。RDA プロバイダーの各利用候補は、要件に適合しているか、特にセキュリティの要件に適合しているかで評価される必要がある。

リモートラボのいくつかは次の追加機能を提供している。

- 専用の物理デバイスバージョン（例えば Samsung モバイルデバイスラボ）。
- 特定のオペレーティングシステムおよびバージョンだけのための汎用デバイス。
- タッチおよびジェスチャー関連の操作を実行するためのロボットアーム。
- デバイスへのアクセスを可能にするための仮想プライベートネットワーク（VPN）。
- さまざまな携帯電話事業者とのモバイルデータ通信接続
- 自動化ツールおよびサービス。

リモートテストラボを使用する際に留意すべき要点として、デバイスの反応が遅いことや、マルチタッチやジェスチャーなどのデバイス操作のオプションが制限されていることがある。これは、時々使用する場合はコスト効果が高いが、多様なデバイスについて長期間にわたって使用する場合は、高価になることが多い。

他の要点として、ローカルラボに存在しないデバイスにアクセスする必要性に対応できるオンデマンドプラットフォームの可用性、プロジェクトの進展に伴い拡大・縮小が可能なラボの拡張性が挙げられる。

NFC／Bluetooth やバッテリー消費といったセンサーを含むテストシナリオは、クラウドでのテストが困難であることが多い。しかし、リモートラボの地理的な位置が異なることで、ネットワークや GPS の接続を必要とするテストにとって有益な場合がある。

テストラボは、実行する必要のあるテストのタイプに応じて、2 つのアプローチのいずれか 1 つ、または組み合わせを使用できる。

5 テスト実行の自動化 55 分

キーワード

API、ユーザーエージェントベースドテスト、デバイスベースドテスト、テストレポート

「テスト実行の自動化」の学習の目的

5.1 自動化手法

MAT-5.1.1 （K2）モバイルアプリケーションテストでの一般的な自動化の手法とフレームワークの違いを識別する。

5.2 自動化方法

MAT-5.2.1 （K2）モバイルアプリケーションをテストするためのさまざまな自動化方法を説明する。

5.3 自動化ツールの評価

MAT-5.3.1 （K1）モバイルテストの自動化ツールを評価する際に考慮する必要のあるさまざまなパラメータを想起する。

5.4 自動化テストラボをセットアップする手法

MAT-5.4.1 （K2）テスト自動化に関して、テストラボを作成する一般的な手法間の違いを、長所と短所を含めて識別する。

5.1 自動化手法

モバイルアプリケーションテストで使用できるさまざまな自動化の手法とフレームワークが存在する。手法の選択は、アプリケーションのタイプによって一部左右される。

一般的に使用されるテスト自動化アプローチは、次の 2 つである。

- ユーザーエージェントベースドテスト
- デバイスベースドテスト

ユーザーエージェントベースドテストでは、ブラウザから送信されるユーザーエージェント識別子の文字列を使用して、特定のデバイス上の特定のブラウザを偽装する。この手法は、モバイル Web アプリケーションの実行に使用できる。一方、デバイスベースドテストでは、テスト対象のアプリケーションを直接デバイス上で実行する。この手法は、すべてのタイプのモバイルアプリケーションに使用できる。

アプリケーションのタイプが、そのアプリケーションに適したテスト自動化フレームワークを決定する場合もある。モバイル Web では、一般的な Web アプリケーション自動化ツールを使用してデスクトップ上でテストできるが、ネイティブアプリケーションでは、固有のツールが必要になる場合がある。プラットフォームプロバイダーが、自身の提供するプラットフォーム専用の自動化ツールを提供する場合もある。

従来のアプリケーションに使用される自動化手法は、モバイルアプリケーションにも同じように適用できることが多い。これらには、ISTQB® Foundation Level シラバス[ISTQB_CTFL_2018]および ISTQB® Advanced Level Specialist Test Automation Engineer シラバス[ISTQB_CTAL_TAE_2016]で説明されているキャプチャ／プレイバック、データ駆動、キーワード駆動、振る舞い駆動のテストが含まれる。

モバイルアプリケーションテストフレームワークが一般的に備える主要な機能を次に示す。

- オブジェクトの識別
- オブジェクトの操作
- テストレポート
- アプリケーションプログラミングインターフェースおよび拡張可能な能力
- 適切な文書
- 他のツールとの統合
- テスト開発のプラクティスからの独立

5.2 自動化方法

自動テストを開発するには、テスト担当者は、自動スクリプトの記録や作成のメカニズムを理解し、そしてアプリケーションのボタン、リストボックス、入力フィールドなどのグラフィカルオブジェクトにアクセスして操作する方法を理解する必要がある。

モバイルテスト自動化に使用されるグラフィカルオブジェクトを識別する方法はいくつか存在する。これらには、画像認識、OCR／テキスト認識、および（Web またはネイティブのようなアプリケーションタイプに応じた）オブジェクト認識がある。

モバイルアプリケーションのテスト担当者は、グラフィカルオブジェクトの検出と識別を行うだけでなく、非常に多種多様なモバイルデバイス上で並行かつ継続的なテストを成功させるために、どのオブジェクトの識別方法が最も有効なのかを理解する必要がある。

各スクリプト作成方法の主な相違点を次に示す。

比較項目	オブジェクト識別	画像／OCR 比較
信頼性	識別子が固定されている限り、画面レイアウトを変更できる。リスクとして、ユーザーから隠されているオブジェクトが、コード上で識別され操作される可能性がある。これは、偽陰性のテスト結果をもたらす可能性がある。	画像は画面サイズに応じて拡大／縮小される可能性があり、レイアウトが変更されるとすぐにテストは失敗する。

ユーザーエクスペリエンス	通常、手動でスクリプトを作成する必要があり、最低限、記録されたスクリプトの読みやすさやメンテナンス性を改善する必要がある。	スクリプト作成を必要としない完全な GUI ベースドテスト。
実行速度	特にシステム製造元により提供されるネイティブツールを使用する場合は、画像／OCR 比較より高速になる傾向にある。	画面をピクセル単位でベースラインの画像と比較する必要があるため、オブジェクト識別より遅くなる傾向にある。
メンテナンス	テストスクリプトの品質に依存する。	変更されたベースラインの画像の用意が主な作業。
作成における課題	持続可能な自動化ソリューションを実現するには、スクリプト言語とソフトウェア設計方法の知識が必要となる。	特にアプリケーションが頻繁に変更される場合では、ベースラインの画像の作成が課題になる。

5.3 自動化ツールの評価

テスト自動化ソリューションを作り出し成功するためには、テスト自動化チームは、適切なツールセットを選択する必要がある。また利用可能なツール間の主な違いを理解し、プロジェクトの要件に対するそれらの合目的性を考慮する必要がある（[ISTQB_CTAL_TAE_2016]も参照）。

テスト自動化ツールの評価パラメータは、次の 2 つのカテゴリに分類される。

- 組織的な適合
- 技術的な適合

組織的な適合のパラメータの説明については、ISTQB® Foundation Level シラバス [ISTQB_CTFL_2018]の 6.2 節を参照されたい。

技術的な適合のパラメータは、次のとおりである。

- 例えば FaceID、指紋、チャットボットなどの新しい機能をアプリケーションが使用するといった、テスト自動化の要件と複雑度。
- 例えばさまざまなネットワーク条件、テストデータのインポートや作成、サーバーサイドの仮想化といった、テスト環境の要件。
- テストレポートおよびフィードバックループの機能。
- ローカルまたはクラウドのいずれかのテストラボにおいて、大規模な実行を管理および促進するフレームワークの能力。
- テストフレームワークと、組織で使用されている他のツールとの統合。
- 現在および将来のアップグレードに対するサポートと文書の可用性。

5.4 自動化テストラボをセットアップする手法

モバイルアプリケーションのテストを行う際、開発者とテスト担当者は、テスト自動化の対象として、使用するデバイステストラボ関連の選択ができる。

- オンプレミスデバイステストラボ
- リモートデバイステストラボ

これらの手法のさまざまな組み合わせを適用できる。これらの主な特徴は、4.4 節で説明し比較している。

オンプレミスデバイステストラボは、一般的に、メンテナンスが困難であり、時間もかかる。モバイルアプリケーションの開発とテストの初期フェーズでは、エミュレーターやシミュレーターと並行して、デバイスをローカルで利用できるようにすることが、最善の手法となる。

アプリケーション開発がより進んだ段階に到達したら、チームは完全なリグレッションテスト、機能性テスト、および非機能性テストを実施する必要がある。これらのテストは、設備が完全に整ったデバイスラボで実施するのが最善である。この点において、クラウドで管理され、継続して更新されメンテナンスされるリモートデバイステストラボがある。そのようなデバイステストラボは、オンプレミスデバイステストラボを補完し、デバイスとオペレーティングシステムの十分な組み合わせを利用可能にし、最新に維持されている環境を備えている。一般的に利用可能なリモートデバイステストラボを使用することによって、機能がより豊富なテストレポートや高度なテスト自動化機能など、より広範なサポート機能のセットを利用できるようになる。

最後に、テスト自動化フレームワークまたは継続的インテグレーションジョブ（CI）を介して大規模に実行する場合、テストラボ全体の安定性が、テストの効率性と信頼性にとって重要になる。このようなラボでは、デバイスとオペレーティングシステムが常に利用可能で安定しているように設計されていることが多い。

リモートデバイステストラボは、アプリケーション開発の後期段階では、必ずしも必要ではない。適切に設計され、メンテナンスされているオンプレミスデバイステストラボは、どのようなリモートデバイステストラボにも勝るとも劣らない。

6 参考文献

6.1 ISTQB®ドキュメント

- [ISTQB_CTFL_2018]:
  ISTQB® Certified Tester – Foundation Level Syllabus – Version 2018

  JSTQB 訳注) 日本語版は"ISTQB テスト技術者資格制度 Foundation Level シラバス 日本語版 Version 2018V3.1.J03"

- [ISTQB_FLAT_2014]:
  ISTQB® Certified Tester – Foundation Level Extension Syllabus – Agile Tester – Version 2014

  JSTQB 訳注) 日本語版は"ISTQB テスト技術者資格制度 Foundation Level Extension シラバス アジャイルテスト担当者 日本語版"

Version 2014.J01"

- [ISTQB_FLUT_2018]:
  ISTQB® Certified Tester – Foundation Level Specialist Syllabus – Usability Testing – Version 2018
- [ISTQB_CTFL_PT_2018]:
  ISTQB® Certified Tester – Foundation Level Specialist Syllabus – Performance Testing – Version 2018
- [ISTQB_CTAL_SEC_2016]:
  ISTQB® Certified Tester – Advanced Level Specialist Syllabus – Security Testing – Version 2016
- [ISTQB_CTAL_TAE_2016]: ISTQB® Certified Tester – Advanced Level Specialist Syllabus - Test Automation Engineer - Version 2016

  JSTQB 訳注) 日本語版は"ISTQB テスト技術者資格制度 Advanced Level Specialist シラバス テスト自動化エンジニア 日本語版"

Version 2016.J01"

- [ISTQB_GLOSSARY]:
  ISTQB®'s Glossary of Terms used in Software Testing, Version 3.2

6.2 参考書籍

- [Knott15] Knott, D., "Hands-On Mobile App Testing", Addison-Wesley Professional, 2015, ISBN 978-3-86490-379-3
- [Kohl17] Kohl, J. , "Tap into mobile application testing", leanpub.com, 2017, ISBN 978-0-9959823-2-1

6.3 その他の書籍および記事

- Boris Beizer, "Black-box Testing", John Wiley & Sons, 1995, ISBN 0-471-12094-4
- Rex Black, "Agile Testing Foundations", BCS Learning & Development Ltd: Swindon UK, 2017, ISBN 978-1-78017-33-68

- Rex Black, “Managing the Testing Process”(3e), John Wiley & Sons: New York NY, 2009, ISBN 978-0-470-40415-7
- Hans Buwalda, “Integrated Test Design and Automation”, Addison-Wesley Longman, 2001, ISBN 0-201-73725-6
- Lee Copeland, “A Practitioner's Guide to Software Test Design”, Artech House, 2003, ISBN 1-58053-791-X、JSTQB 訳注）日本では「はじめて学ぶソフトウェアのテスト技法」（日経 BP 社, 2005 年）として発行されている。
- Rick David Craig, Stefan P. Jaskiel, "Systematic Software Testing", Artech House, 2002, ISBN 1-580-53508-9、JSTQB 訳注）日本では「体系的ソフトウェアテスト入門」 (日経 BP 社, 2004 年) として発行されている。

6.4 リンク（Web／インターネット）

免責事項：すべてのリンクは、2019 年 1 月 5 日の時点でアクセスできることが確認されている。

- [URL1] http://gs.statcounter.com/
- [URL2] www.owasp.org
- [URL3] https://developer.android.com/studio/test/monkey
- [URL4] https://www.google.de/amp/s/techcrunch.com/2016/05/31/nearly-1-in-4-people-abandon-mobile-apps-after-only-one-use/amp/
- [URL5] https://www.google.com/accessibility/
- [URL6] https://www.apple.com/uk/accessibility/
- [URL7] https://www.w3.org/WAI/standards-guidelines/mobile/
- [URL8] https://www.slideshare.net/karennjohnson/kn-johnson-2012-heuristics-mnemonics
- [URL9] http://www.kohl.ca/articles/ISLICEDUPFUN.pdf

7 付録 A – 学習の目的／知識の認知レベル

本シラバスに当てはまるものとして、以下の学習の目的が定義されている。学習の目的に従ってシラバスのそれぞれの課題を試験する。

7.1 レベル 1：記憶レベル（K1）

用語または概念を認識し、記憶して、想起することができる。

キーワード：識別（identify）、記憶（remember）、検索（retrieve）、想起（recall）、認識（recognize）、知識（know）

例：

「故障」の定義を以下のように認識できる。

- 「エンドユーザーまたは他の関係者にサービスを引き渡しできないこと」、または
- 「コンポーネントやシステムが、期待した機能、サービス、結果から逸脱すること」

7.2 レベル 2：理解レベル（K2）

課題に関連する記述について理由または説明を選択することができ、テスト概念に関して要約、比較、分類、類別、例の提示を行うことができる。

キーワード：要約（summarize）、一般化（generalize）、抽象（abstract）、分類（classify）、比較（compare）、配置（map）、対照（contrast）、例示（exemplify）、解釈（interpret）、変換（translate）、表現（represent）、推察（infer）、結論（conclude）、類別（categorize）、構造モデル（construct models）

例：

できるだけ早期にテストの分析と設計を行わなければならない理由を説明することができる。

- 早期に欠陥を摘出すれば除去コストが低くなるため。
- 重要な欠陥をより早く見つけるため。

統合テストとシステムテストの類似点と相違点を説明することができる。

- 類似点：複数のコンポーネントをテストし、非機能面をテストする。
- 相違点：統合テストではインターフェースと相互作用に着目し、システムテストでは、システム全体（例えばエンドツーエンド処理）に着目する。

7.3 レベル 3：適用レベル（K3）

概念または技法を正しく選択することができ、それを特定の事例に適用することができる。

キーワード：実装（implement）、実行（execute）、使用（use）、手順の実施（follow a procedure）、手順の適用（apply a procedure）

例：

- 有効／無効に分ける境界値を見分けることができる。
- すべての遷移をカバーするため、特定の状態遷移図からテストケースを選択できる。

参考資料（学習の目的の認知レベル用）

Anderson, L. W. and Krathwohl, D. R. (eds) (2001) A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives, Allyn& Bacon: Boston MA

8 付録 B – 本分野での用語集 - 固有の用語

用語	定義
2G	第 2 世代モバイルワイヤレス電気通信技術。
3D Touch	「感圧タッチ」を参照
3G	第 3 世代モバイルワイヤレス電気通信技術。
4G	第 4 世代モバイルワイヤレス電気通信技術。
5G	第 5 世代モバイルワイヤレス電気通信技術。
ADB	Android Debug Bridge（ADB） - デバイスとの通信を可能にするコマンドラインツール。
AVD	Android Virtual Device の頭字語。
Android Device Monitor（ADM）	Android アプリケーションのデバッグ／分析ツール用のユーザーインターフェースを提供するスタンドアロンツール。
Android Studio	Android 用の公式な統合開発環境（IDE）。Android Studio は、Android デバイスのすべてのタイプでアプリケーションを構築するためのツールを提供する。
Bluetooth	近距離ワイヤレス通信技術の 1 つ。
CPU 周波数	プロセッサのクロックレート。
CRUD	データを対象にする作成（Create）／読み込み（Read）／アップデート（Update）／削除（Delete）のニーモニック。
DPI／PPI	1 インチ当たりのドット／ピクセル数（Dots／pixels per inch）の頭字語 - ディスプレイの密度をドットまたはピクセル単位で表す数値。
GPS	全地球測位システム（Global Positioning System）の頭字語。地球上空を周回する衛星ネットワークが、時間信号を送信する。受信デバイスは 3 つ以上の衛星の信号を受信することで、三角法により衛星との相対位置を計算できる。
GSM	Global System for Mobile Communications（元々は Groupe Spécial Mobile）の頭字語。欧州電気通信標準化機構（ETSI）が開発した標準で、モバイルデバイスで使用される第 2 世代デジタルモバイル通信ネットワーク向けのプロトコルを規定している。現在、世界で最も使用されているモバイル通信用標準である。
GSM セル	GSM ネットワークの構成要素で、一意のセル ID で識別される。
I18N	国際化（Internationalization）の数略語（数値ベースの単語）。
IDE	統合開発環境（Integrated development environment）の頭字語。A コンピュータープログラマー向けにソフトウェア開発用の機能を包括的に提供するソフトウェアアプリケーション。

Instruments	Xcode ツールセットに含まれる性能分析とテストのツール。
IoT アプライアンス	Internet of Things（モノのインターネット）。インターネットに接続されているデバイスや例えば関心のあるセンサー。
Jailbreak	オペレーティングシステムによって実装されているソフトウェア制限を取り除くことを目的として、権限を昇格させる操作。iOS に対して使用される用語で、Android に対しては root 化が使用される。
L10N	ローカル化（localization）の数略語。
NFC	近距離無線通信（Near Field Communication）の頭字語。近距離内での無線通信技術。
OCR	光学式文字認識（Optical Character Recognition）の頭字語。電子的画像に含まれるテキストのイメージの認識およびその文字列のマシンコードへの変換。
OTA	Over The Air の頭字語。無線信号を介するデータ転送。アプリケーションをデバイスにインストールする際に、ケーブル接続を介することなく、ソースから直接インストールすることを意味することが多い。
QR コード	QR コード（Quick Response Code の略語）は、マトリクスバーコード（または 2 次元バーコード）の 1 つのタイプの商標である。
root 化	デバイスオペレーティングシステムに対する root アクセス権限を取得するプロセス。この用語は、Android プラットフォームに対して使用されることが多い。iOS では Jailbreak が使用される。
Web アプリケーション	インターネット上でホストされ、ブラウザを介して使用されるアプリケーション。
Xcode	OSX および iOS のアプリケーションを開発するために Apple によって提供されている統合開発環境。
アスペクト比	ディスプレイまたはイメージの横対縦の比。
アップロード競合	アップロード対象のファイルがアップロード先に既に存在しているというエラー。
アプリケーションショートカット	アプリケーション開発者によってアプリケーション内で定義される一連の固有アクションへのショートカット。Android 7.1 以降でサポートされる。
アプリケーションストア	アプリケーション配布プラットフォーム。このプラットフォームを使用することで、開発者はアプリケーションをアップロードでき、ユーザーはアプリケーションを検索して自身のプラットフォームにダウンロードおよびインストールできる。
アプリケーションデータの保持	ユーザー生成のデータそして／またはアプリケーションのコンテンツは、アプリケーションがアンインストールされてもデバイスに維持される。これは、他のアプリケーションによって、または同じアプリケーションを再インストールしたときに、アクセスできるようにするためである。

アプリ内課金	オプションのコンテンツおよび機能をアプリケーション内から直接購入できる仕組み。
ウェアラブル	時計や眼鏡など、身に着けるコンピューターデバイス。
エミュレーター	ハードウェアの動作を模倣するソフトウェアアプリケーション。
エンタープライズアプリケーション	組織内で内部的に使用するために作成されたアプリケーションで、公共での使用は意図されない。
オーバーフロー	入力データが収容先の容量を超える状況。
オンプレミスラボ	ラボのユーザーと同じ場所に物理的に存在するラボ。
外観と操作感	対象物に対する視覚的および感情的な印象。
下位互換性	以前のバージョンのプラットフォームで動作するという、アプリケーションの能力。
外部メモリ	標準のインターフェースを介してデバイスに追加されるメモリ。現在、モバイルフォンでは SD カードが最も一般的である。
仮想専用線（VPN）	公共ネットワークを介する、暗号化された専用チャネル。
画面リアルエステート	ディスプレイで提供される画面サイズ。
感圧タッチ	Apple Inc.が開発した技術で、トラックパッドとタッチ画面の表面に適用された圧力の違いを識別できるようにする。
機内モード	モバイルデバイスの特別な動作モードの 1 つ。航空機の操縦／通信システムへの干渉を防止するため、無線送信を無効化する。
基本設定	デバイスまたはアプリケーションの一般的な構成パラメータ。ユーザーが変更できる。
機密データ	パスワードや個人データなど、特別な保護を必要とするデータ。
クロスプラットフォーム開発フレームワーク	同じコードベースを使用してさまざまなプラットフォーム向けのアプリケーションを開発するためのフレームワーク。
グローバル化	「国際化」を参照。
広告ベースドアプリケーション	アプリケーション収益モデルの 1 つ。開発組織は、アプリケーション内に広告を表示することで収益を得る。
国際化	さまざまな地域向けのバージョンを収容するためにアプリケーションを準備するプロセス。
コンパニオンデバイス	スマートデバイスに依存し、それと連携して動作するように設計されているコンピューターデバイス。
サードパーティマーケットプレイス	プラットフォームプロバイダー以外によって運用されるアプリケーション配布プラットフォーム。
サイドローディング	アプリケーションストア以外の方法でアプリケーションのローディング／インストールを行うこと。

実行環境	実行モデルの実装環境。実行システムとも呼ばれる。
シンクライアント	クライアント／サーバーアプリケーションで、特別に小さくなるように設計されたクライアントで、データ処理の大部分はサーバーによって処理される。
シングルティア	バックエンドアプリケーション設計手法の 1 つ。単一（シングル）サーバーが、アプリケーションにとって必要なサービスすべてを提供する。
ジェスチャー	デバイスの定義済み機能を実行するために行う、ピンチやスワイプなどの特定の操作パターン。例えば、スマートデバイスの画面でズームイン／ズームアウトを行うために、ピンチが一般的に使用される。
ストアアンドフォワード	データ同期化手法の 1 つ。データはローカルに格納（ストア）され、適切なネットワーク接続が確立されるとサーバーに転送（フォワード）される。
スマートフォン	携帯型のパーソナルコンピューター。モバイルオペレーティングシステムを搭載し、音声通話、SMS（Short Message Service：テキストメッセージサービスとも呼ばれる）、およびインターネットデータ通信のためのモバイルブロードバンドモバイル通信ネットワーク接続機能を備える。
セッション	有限期間のイベント。
セッションシート	テストセッションのスコープ定義および記録を行うための文書。
センサー	デバイス、モジュール、またはサブシステムで、それが属する環境のイベントまたは変化を検出することを目的としている。それらの情報を他の電子機器（多くの場合、コンピュータープロセッサ）に送信する。
ソフトウェア開発キット（SDK）	特定のプラットフォーム向けにソフトウェアを開発するための一連のツールおよびライブラリ。
ソフトキーパッド	ソフトウェアで実現される仮想キーボード。ユーザーはディスプレイ上で使用する。
縦向きモード	縦長に表示されるデバイスの向き。
タブレット	モバイルデバイスのタイプの 1 つ。7 インチ以上の画面サイズを持つデバイスに対して使用されることが多い。
通知	デバイスによって送出されるお知らせ。
低電力モード	モバイルデバイスの動作モードの 1 つ。電力を節約するためにユーザーまたはデバイス自体が有効化できる。
データ完全性	ライフサイクル全体にわたるデータの正確性と一貫性。保管、処理、および読み込みの処理を含む。

データ妥当性確認	データが正しく、正確で、一貫性があり、使用できるかどうかの評価。
データ同期化	複数のソース間でデータを同じ状態にするプロセス。
デッドスポット	「ブラインドスポット」を参照。
デバイスの多様性	利用可能なデバイス、それら固有のハードウェア構成、およびアプリケーションやユーザーエクスペリエンスに及ぼす影響の多様性。
電力消費	電力の消費量。
同期通信	データ転送手法の 1 つ。データフローは同じ速度で送信（アップストリーム）および受信（ダウンストリーム）され、タイミング信号により分離される。
取引ベースドアプリケーション	ユーザーが取引ごとに料金を支払うアプリケーション。
内部メモリ	デバイスハードウェアに付属しているメモリ。
ニーモニック	記憶を助けるように工夫された略語。
ネイティブ アプリケーション	特定のプラットフォーム向けに特別に設計されたアプリケーション。プラットフォームの API とプラットフォーム提供の開発ツールを使用する。
ハイブリッドアプリケーション	ネイティブ技術と Web 技術を組み合わせて使用するアプリケーション。ハイブリッドアプリケーションは一般的に、ネイティブフレームを使用してデバイスにインストールされ、デバイスライブラリなどと連携する。さらに、Web サーバーから取得したコンテンツを表示する。
バーコード	機械判読が可能な、データの光学的表現。
バイトコード	ソフトウェアインタープリタによって効率良く実行されるように設計された命令語セットの 1 つ。ポータブルコードまたは p コードとも呼ばれる。
バックエンドシステム	他のシステム向けの機能を提供するサーバーシステム。
バックグラウンドアプリケーション	バックグラウンドで実行中のアプリケーション。
バッテリーの状態	電力消費に関連するユーザー定義または事前定義のプロファイル。モバイルデバイスで有効化できる。
非同期通信	通信のタイプの 1 つで、データを安定したストリームではなく、断続的に送信できる。
ビューポートサイズ	ブラウザがレイアウトを画面に調整するために使用する仮想画面サイズ。
ファットクライアント	クライアント／サーバーアプリケーションにおいて、データ処理の一部またはほとんどを処理するように設計されているクライアント。

フィーチャーフォン	モバイルフォンのクラスの 1 つ。ベーシックフォンよりも多くの機能（ブラウザなど）を備えているが、スマートフォンのフル機能は備えていない。
フォアグラウンドアプリケーション	デバイスのフォアグラウンドで実行するアプリケーション。ユーザーが直接に操作できる。
フラットファイル	内部階層を持たないファイル。
フリーミアムアプリケーション	ビジネスモデルの 1 つ。アプリケーションは無償でダウンロードできるが、アプリケーション内で有償オプションの購入をユーザーに勧誘する。
ブラインドスポット	ワイヤレス電気通信ネットワークが利用できない場所。
ブロック／おやすみモード	モバイルデバイスの動作モードの 1 つ。一般的な通知や着信などの機能を抑制するために、ユーザーが有効化できる。
プレインストールされたアプリケーション	デバイス製造元によってインストールされるモバイルアプリケーション。これらのアプリケーションは、ユーザーがアンインストールできないことが多い。
ベーシックフォン	発呼、電話番号の保存、SMS の送信、時計、アラームなどの最小限の機能を備えるモバイルフォン。
ペルソナ	特定のユーザーグループのモデル／アーキタイプ。
マルチタッチ	操作タイプの 1 つ。さまざまなタッチイベントを並列に使用してデバイスを操作する。
マルチティア	バックエンドアプリケーション設計手法の 1 つ。複数のサーバーがそれぞれ固有の機能を提供する。
マルチプラットフォームアプリケーション	複数のプラットフォームすべてに対して同じコードベースを使用して実行するように設計および開発されたアプリケーション。
向き	モバイル内でのオブジェクトの配置。デバイスの使用方法を表すために使用されることが多い。横向きまたは縦向きのいずれかである。
モバイル OS	特にモバイルデバイス向けに設計されたオペレーティングシステム。
モバイルアプリケーションモバイルテスト	モバイルアプリケーションのテスト。
モバイルエミュレーター	ハードウェアプラットフォームを仮想的に表現したもの。例えば、Android エミュレーターは、実 Android OS イメージを実行する仮想ハードウェアである。ほとんど同じ OS イメージをハードウェアに展開できる可能性があり、実 OS のように動作する。

モバイルシミュレーター	仮想実行環境の 1 つ。例えば、iOS シミュレーターは iOS の類似環境を提供するが、実際の iOS ではない。
モバイルスペース	市場やそれに関係する組織からデバイスやアプリケーションまでの、モバイルデバイスに関連するあらゆるものを含む総称。
モバイルデータ	モバイル通信ネットワークを介して転送されるデータ。
モバイルデバイスタイプ	基本的な機能によるモバイルデバイスの分類。よく使用される分類として、ベーシックフォン、フィーチャーフォン、スマートフォン、ファブレット、タブレット、およびウェアラブルがある。
モバイルプラットフォーム	モバイルオペレーティングシステムを取り巻くエコシステム。開発ツール、オペレーティングシステム自体、アプリケーション配布チャネルを含むことが多い。
モバイル通信ネットワーク	独立しているが接続されている複数のセルで構築されるネットワーク。
有償アプリケーション	アプリストアでの販売を通じて収益をもたらすアプリケーション。
横向きモード	横長に表示されるデバイスの向き。
ライブラリ	不揮発性リソースのコレクションの 1 つ。コンピュータープログラム（すなわちアプリケーションの機能）によって使用される。
リモートデバイスアクセス	ユーザーとは異なる場所に配置されているデバイスを操作すること。インターネット経由で行うことが多い。
ローカル化	翻訳およびフォーマット調整などの活動によってアプリケーションまたは製品を特定の地域向けに調整するプロセス。
割り込み	イベントの処理中に発生する別のイベント。