Adoc Teslab

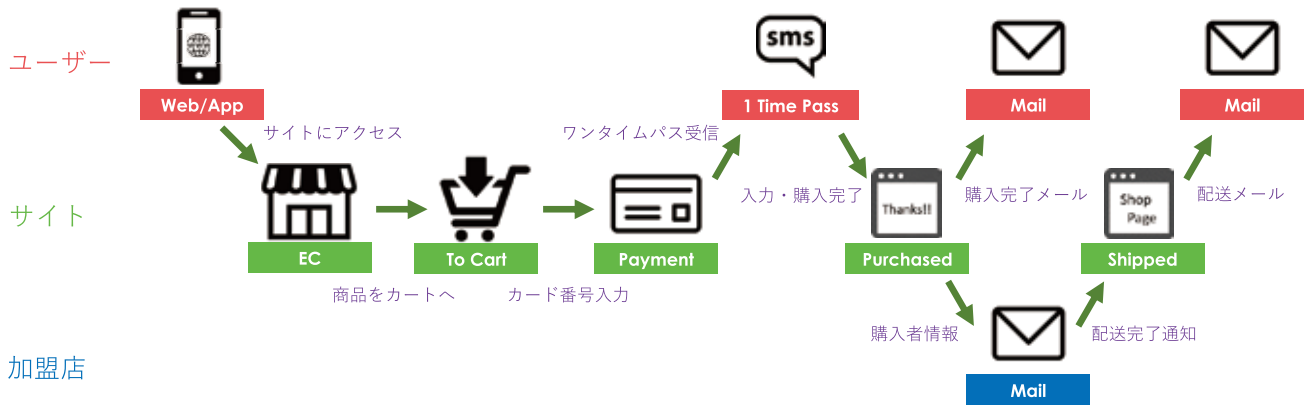
DX時代の革新的なテスト自動化テクノロジーとは

『AIによる探索型モデルベーステスト』の有用性を探る

開発現場におけるテスト工程の負荷は30-50%を占めるとされ、その手法はいまだ手動に頼らざるを得ない領域を多く残している。またテスト自動化の試みには課題が山積し、成果につなげるには相応の投資が求められ、現状では実態に即さない導入ケースも散見される。デジタルによって新しい商材・サービスの速やかな市場投入が求められるDX時代において、こうした課題の解消を目指し、開発現場の飛躍的なQCD向上を支援する先駆的なテスト自動化技術を解説する。

●本資料のサマリ
・現行のテスト手法ではDX時代の複雑なサービスに対応できない
・探索的テストは解決策となりえる利点が多いが、課題もある
・課題をクリアするモデルベーステストの適用について
・デジタルツインへの応用

DX時代のサービスフローと品質テスト



図は、ECサイトにおけるユーザーの購入フロー例であり、ECサイト開発にあたってはこの一連の挙動をテストする。

ECサイト本体の挙動は緑色のラベルで示されており、これに限っていえばSeleniumに代表されるブラウザ操作自動化ツールの活用を検討できる。ここで注目したいのは赤色や青色ラベルで示される外部システムの存在で、これらがECサイトの挙動に関与している点だ。ユーザーはECサイトで買い物をするために自分のスマートフォンを操作する。決済にあたってはワンタイムパスが採用されており、ECサイトが起動するWEBブラウザとは別にユーザーサイトでのSMSサービスの操作・確認手順が介在する。購入後は発送フローへ直結し、管理システムや発送システム、またユーザーとのコミュニケーションが発生。ここでもやはり外部アプリケーションとの行き来が生じる。

これら一連のやり取りは全てECサイトへのインプット、あるいはECサイトからのアウトプットとなり、ECサイトのテストに係るパラメータであるが、こうしたパラメータの急激な増加は最近のサービスの特徴の1つであり、システムテストが複雑化している理由が理解できる。さらには、フローの冒頭で登場するスマートフォン操作1つをとっても、スマートデバイスの種別、機種、OSの種別、WEBブラウザの種別といった多種多様な分岐があり、実際のテストに

あたってはこれらの組み合わせを加味した設計を求められる。

このように、システムの複雑化や多様なパラメータはテスト手順を掛け算式に膨らませ、テスト実行にあたって費やされる工数は莫大なものとなるばかりか、もはや品質基準の再考を視野に入れないかぎり、人力で行うには、限界があるという状況が起こっている。

探索的テストのメリット

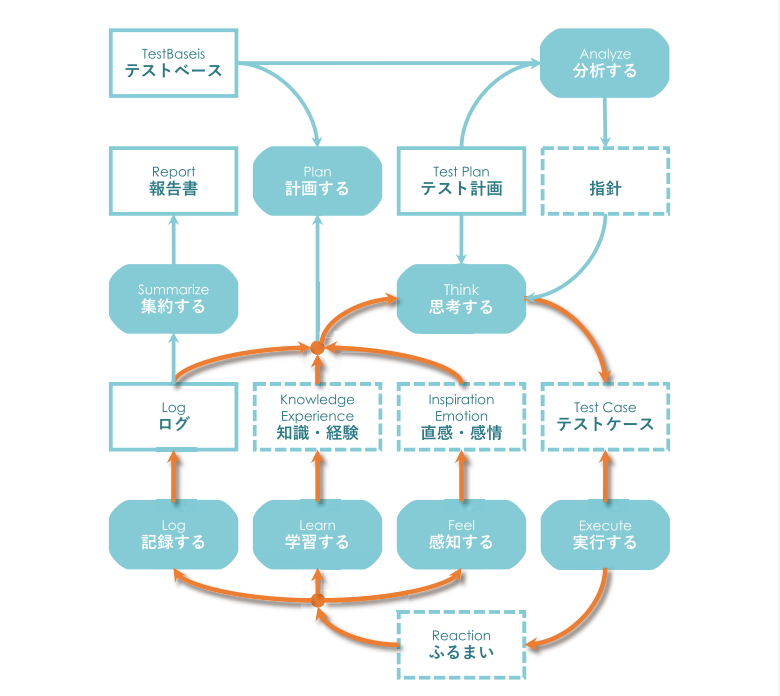
探索的テストの必要性は、前述のような課題の解決にあると考えられる。従来の手法では、製品仕様や開発設計からのインプットをもとにテスト観点を洗い出し、テスト項目の数やテストの回数を増やすことでサービスの品質を担保する。一方で探索的テストはテスター個人に自由意思と責任とを持たせ、その知見と経験をもとにテストを行う。具体的には、テスターがテスト機を操作しながらテスト対象システムを学習し、バグの潜んでいそうな箇所を類推し、テストケースを考案する。その考案されたテストケースに基づきテストを実行。そして結果をフィードバックし、テストケースを再考する。これらを繰り返し、テスターの知見と経験とを元手に質の高いテストケースを作成する。

探索的テストの優れたところは、この質の高いテストケース＝バグ検知率の高いテストケースを効率的に生成できる点にある。ECサイトの例のような現代の複雑・多様なテストにおいても同様の効果があり、指数関数的な検討／テスト項目数の増加を回避しつつ、より多くのバグを検出できるようになる。また、テスターの視点がユーザー体験を含むところも特徴であり、機能テスト、性能テスト、ユーザビリティテストを同時に行えることは大きなメリットだ。従来は不可能だったテストを可能にするばかりか、テストの生産性も向上するという点では良いところづくめの手法であり、この時点では採用の検討を躊躇する理由はないといえるだろう。

探索的テストが抱える課題

表に示されているのは、探索的テストの概念的なフローである。このフローを短いサイクルで回していくことで探索的テストは実装される。

表に従って振り返ると、(1)テストベースを分析し、テスト計画を立案。(2)テスケース作成。(3)テスト実行。(4)テストレポートと経験知へのフィードバック。(5)テストケースの修正。このフローを処理するにはシステムと開発への理解が不可欠で、そうした素養は開発チーム内にあるが逼迫する開発リソースを割くことが現実的ではないというケースは多い。またユーザー視点を取り入れる必要性からテストチームの独立性は保たれるべきで、折衷案としてテスターが開発チームと机を並べてテストするかたちが採用されることもある。しかしいずれにせよ素養と処理能力とを備えた高いエンジニアスキルの稼働が前提であり、このテストエンジニアの確保が課題となる。



優秀なテストエンジニアを確保できたとして、改めて表に戻る。探索的テストの場合、PlanとKnowledge Experienceが対になるが、Knowledge Experienceは体系的に表現しづらいためPlanもその影響を免れず、テストの根拠は非体系的となる傾向にある。またTest Case KnowledgeもExperienceと対にな

る。Test Caseもまた非体系的な傾向を帯びるとともに、Test Caseの更新は探索的テストの重要な要素であることから、Test Planに予めその観点を表現することは難しい。

つまり「今回のテストは前回と比べてどうか？」「テストの進捗はどうか？」といった基本的な問いに対して、エビデンスの用意が困難なのである。このようにテストの評価機軸が曖昧であることと、またドキュメントの欠如による属人化は探索的テストの大きな課題であると考えられる。

探索型モデルベーステストの適用

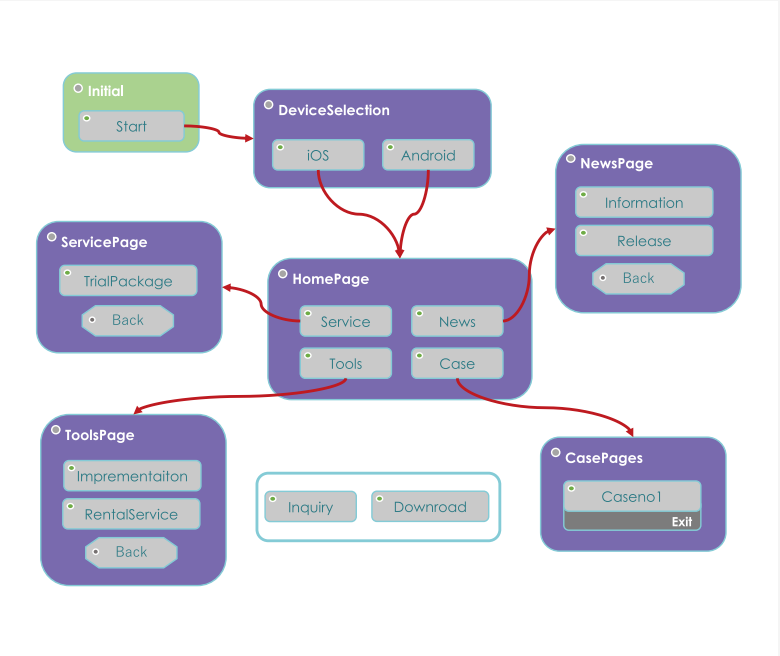
ADOC Testing Service（以下ATS）では、探索的テストにモデルベーステストの手法を組み合わせ、その上で動くAIを実装し自動化を施すことでこの課題を解消した。以下にその手順を記載する。

※ATSは自動領域と手動領域とを分けて実行するが、以下は自動領域での手順である。

モデルベーステストでは最初に"モデル"を準備する。テストベースがそのインプットとなるが、作成されるモデルはテスト対象の状態遷移を表す。状態遷移であるため、モデルにはサービスを受けるユーザーが想定し得るすべての状態を表現することが望ましい。言い換えると、モデルには想定し得るすべてのテストケースが含まれる。

具体的には、モデルは"ステート"と"アクション"とに分けて記載される。サービスによって現れる状態の単位がステートであり、その状態の中で行える行動がアクションである。各ステートとアクションには前提や制約、ルールを補足し、併せて遷移の方向付けを行う。注意したいことは、モデルは状態遷移であり画面遷移ではないという点だ。ステートの大きさはモデル作成者の意図により変わり得るが、ステートを多く配置することでより複雑なテストが可能となる。ただしステートやアクションの数が増えることで相対的に各状態の特徴が薄れて行く点は留意すべきである。

モデルを記載した例が下図である。緑色のボックスはフローの起点を表しており、紫色のボックスがステート、ステートの中にある灰色のボックスがアクションとなる。また状態遷移の方向は赤色の矢印で示されて、すべての状態で発生し得るアクションに関しては白抜きのボックスに抜き出して記述されている。すべてのアクションの中から少なくとも1つのアクションへExitが補足され、フローの終点を示している。



AIよる高カバレッジと高精度のバグ検出

作成されたモデルの中を動く経路は通常ランダムで選択されて、Exitの補足されたアクションを選択するまでの一連の動きが1つのテストシナリオに該当する。各アクションの自動化とモデルとの紐づけ、テスト観点の付与を行うと、機械化の特性を活かした総当りテストを実行することができる。テストを繰り返すたびに実行カバレッジを算出し、進捗が可視化される。

さらにAIを活用し、選択を半恣意的に行うことで、探索的テストを実装する。AIは各ステートとアクションの特徴、開発履歴、過去のバグ情報等を学習し、リスクの高いケースを考案して実行アクションを選択する。実行結果はアクションの特徴に追記され、以降の選択に反映される。

ここまでの手順で、現代のテスト事情におけるい

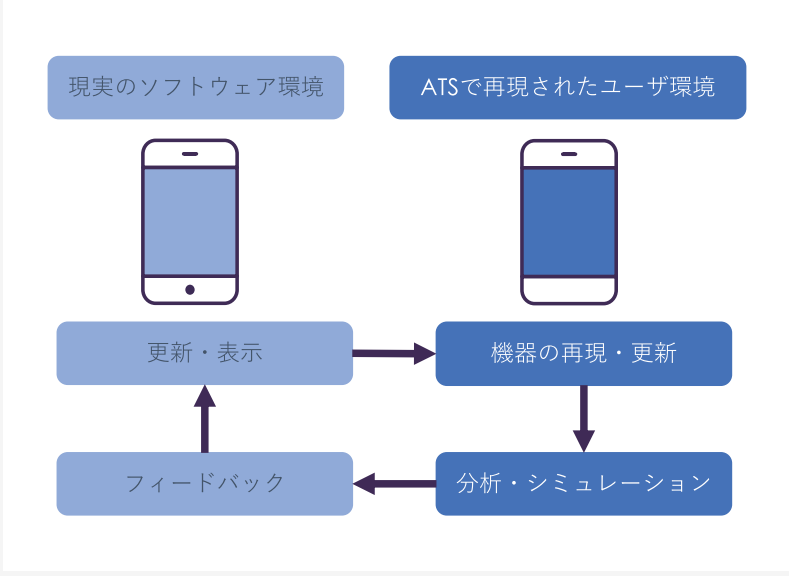
くつかの課題が解決されたことを確認してみよう。探索的テストの実装によって複雑かつ多様なシステムのテストを可能にし、より生産的かつ精度の高いバグ・ハントが実行できるようになった。またモデルを用意することで全テストケースの明示と実行したテストカバレッジを可視化することができるようになり、過去のテストとの比較やテスト品質のコントロールが可能となる。モデルとその実行ログはテストのエビデンスとなり、従来のテストについてまわる膨大なドキュメント作成が不要となることもメリットの1つである。

さらにテストケースの作成プロセスをAIに渡すことで、テストにおける自動化領域が拡張され、大幅な工数削減を期待することができる。またテストシナリオもその意味が変わり、自動生成が前提となって、リリース毎に行われる回帰テスト等で発生するシナリオ書き直し作業も不要となる。そしてテストの知見はAIに収納され、探索的テストの課題である属人化を回避し、テストの標準化およびテスト規模の拡大等にもスムーズに対応できるようになった。

終わりに ― デジタルツインへの応用

最後に、ここまで述べてきたATSの技術の活用例をあげておく。世の中を変革する技術・アイデアが満ちている現代では、リリースされるソフトウェアの数はこの数年のうちにも膨大なものとなっていく。このためサービスリリースの迅速化および新規サービス開発へのインプット増大は、もはや社会的要求である。

これを解決するため、昨今活用が進む「デジタルツイン」の手法をテストへ応用する動きも進むだろう。ATSにおいて用意されるモデルに、ユーザーの実使用データを連携することでそれが可能となる。現代のテストの多様性は関与するシステムの数だけではなく、人間によるインプットの自由さ・豊富さによっても影響されることを考慮すべきである。テストへのデジタルツイン実装についてはまた改めて詳述したい。



株式会社アドックインターナショナル
マーケティンググループ　〒190-0012　東京都立川市曙町 2-36-2 ファーレ立川センタースクエアビル 6F
サービス内容、導入についてのご相談 TEL：042-528-8733(平日 9:00-17:00)　MEIL：info@adoc.co.jp　www.adoc-teslab.com