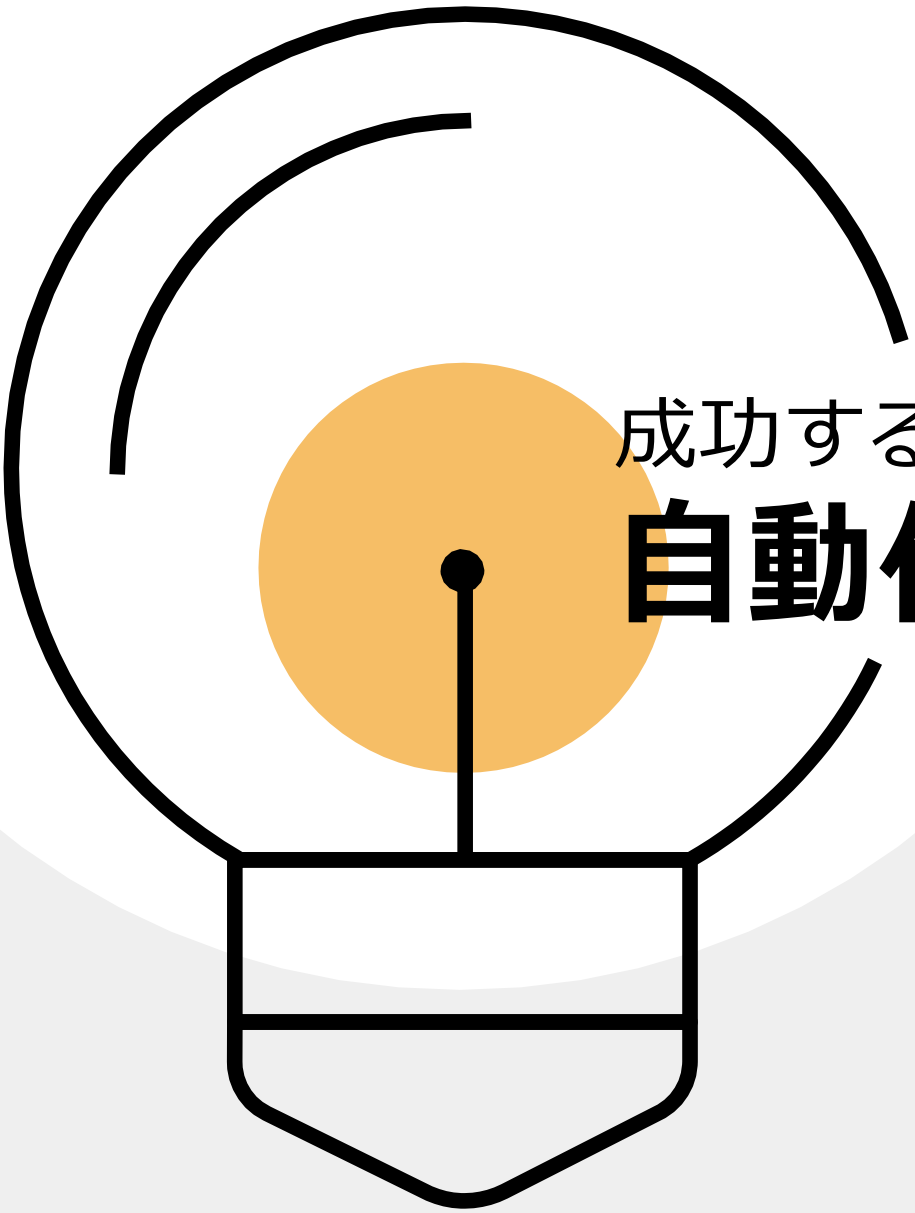
Ranorex®



成功するテスト自動化

自動化プロジェクトを成功に導くための

10 の重要なストラテジー

自動化プロジェクトを成功に導くための 10 の重要なストラテジー

エグゼクティブ サマリー

本ホワイト ペーパーでは、自動化プロジェクトを成功させるための推奨事項を、ステップバイステップ形式でご紹介します。これらの推奨事項は、品質保証において長年にわたって顧客を支援してきた、経験豊富なテスト自動化の専門家が提唱するものです。他のあらゆるプロジェクトと同様に、テスト自動化プロジェクトの場合も、計画段階における決定が成功 (あるいは失敗) を左右します。このため、専門家は次の 2 点を推奨しています。ひとつは「自動化プロジェクトを開始する前に、目標の設定、現状のプロセスの分析、および適切な実装チームの構築のために必要な時間をかけること」、もうひとつは「選択したツールが実際のテスト環境で適切に機能することを保証するために、自動化のためのサンプル テスト ケースを準備し、オンサイトで PoC を実施すること」です。本ホワイトペーパーの最後のセクションでは、テスト ケースの保守にかかるコストを管理し、自動化への投資から最大限の成果を得て、長期的な成功につなげるための戦略をご提案します。

目次

1. 現在のプロセスを分析する

テストの自動化を実装する前に、自動化が必要な理由と、自動化をおこなうためのチームの準備ができているかを判断します。自動化を検討している理由は、恐らく QA プロセスがリリース サイクルのボトルネックになったからでしょう。その原因が、「回帰テストを手動で完了することに時間がかかり過ぎていること」であれば、テスト ケースを自動化することで、ボトルネックが緩和される可能性が十分にあります。しかし、「要件の変化や、開発チームと QA チームのコミュニケーションの壁のために、開発サイクルの後半になるまでテストを開始できないこと」が原因だとしたらどうでしょうか？そのような場合、自動化プロジェクトはチームが期待するようなメリットをもたらさないかもしれません。自動化のメリットを得るためには、組織に適切なプロセスが整っているかどうかを評価してください。

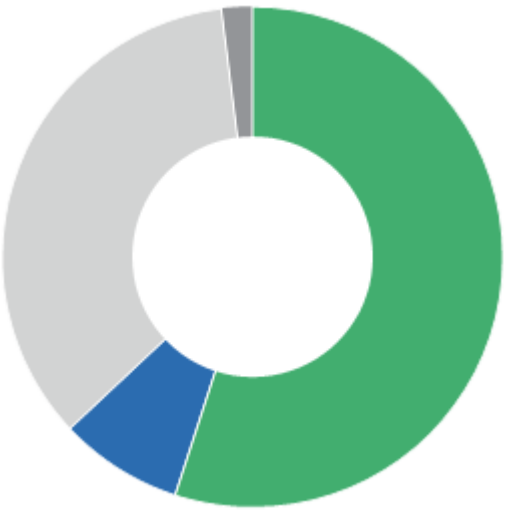
簡単に言うと、テストの自動化は、壊れたプロセスを修正することはできないのです。さまざまな利害関係者が、プロセスにおける自身の役割について異なる期待を抱いている場合は、特にそうです。Ranorex のシニア QA エンジニアである Larissa Stoiser は次のように説明しています。

> 時折、テストを自動化すればすべてが上手くいくと考える人たちがいます。しかし、チームは最初にワークフローを整えておく必要があります。そして、スタッフはテストの目的と品質の意味に関するトレーニングを受けると良いでしょう。スタッフは、まず QA としての役割を認識しなければなりません。テストの自動化は重要ではありますが、高品質なソフトウェア開発という大規模な業務の中のほんの一部に過ぎません。

そのため、自動化プロジェクトのできるだけ早い段階で、プロセス改善が必要なポイントを明らかにすることが重要です。検討すべきポイントには、チーム構造、コミュニケーション、および開発手法などがあります。Ranorex が行った 2017 年の調査によると、大半の組織が主にアジャイル型の開発アプローチを使用している一方で、非効率性が内在するウォーターフォール型あるいは混合型のアプローチを使用している組織も相当数あります。

Ranorex による調査: アジャイル vs ウォーターフォール

開発環境



本調査の結果および結論の詳細については、Ranorex ホワイトペーパー「Accelerating Toward Continuous Testing: 9 Things to Know About Software Testing on Agile Teams」を参照してください。

開発に対して完全に DevOps な手法を取っているか、あるいは従来のウォーターフォールの手法を取っているかに関わらず、各関係者 (開発者、QA、システム管理者、ビジネス担当者など) が緊密に協力する機能横断的な小規模チームを構築することで、コミュニケーションを向上させることができます。ただし、チーム間のオーバーラップを最小限にとどめて、できるだけ互いのチームが独立して行動できるよう、戦略的にチームを構築しましょう。Ranorex のシニア セールス エンジニア、Jason Branham は次のように述べています。

> 私はこれまでに何度か、中規模の組織が、具体的な戦略を持つことなく、単純にアプリケーションの画面や機能によって QA チームを分けるのを目撃しました。その結果、ある機能が 10 回テストされる一方で、別の機能は 50 回テストされるといった場合があります。分業自体は自動化プロセスではありませんが、適切に行われる必要があります。そうでなければ、自動化プロジェクトは非常に困難なものになるでしょう。複数の人が重複して同じ作業に取り組むようなことを起こさないためにも、コミュニケーションが重要です。QA プロセスは、自動化を成功させるために不可欠です。テスト範囲、そしてテストを行う理由を知っておく必要があります。自動化を容易にしたいのであれば、すべて文書化しましょう。

機能横断型のチームでは、ビジネス アナリスト、開発者、およびテスターの観点から要件を話し合います。製品が、要件を正しく実装し、ユーザーに最大の価値を提供できるように、開発およびテストされていることを確認するためです。

Gurock GmbH の TestRail 製品マネージャーである Simon Knight は、開発プロセスを効果的にするためのコミュニケーションの重要性について以下のように述べています。

> 通常、製品が正しく開発されているかどうかを判断するのは要件定義の問題です。つまり、「そのリリース/機能で何を達成しようとしているのか、あなたは十分理解していますか ?」ということです。要件に関する問題が発生する原因には、さまざまなものがあります。しかし多くの場合、その根本的な原因はコミュニケーションにあります。言い換えると、製品の所有者、管理者、利害関係者の頭の中にあるものを抽出し、その情報を開発者が実際に作業できるものに変えることができるかどうかです。

製品マネージャーとしての役割において、Knight はチーム内で、アイディア、コンセプト、機能、およびストーリーを明確に共有できるよう努めています。その目的は、開発者/テスターが、自分が何に取り組んでいるのか/何をテストしているのかについてよく理解すること、そしてチーム全体が、開発プロセスでどのようなリスクに対処する必要があるのか、どのような受け入れ基準とテストによってこれらのリスクを最も軽減できるのか、についてよく理解することです。

2. 自動化プロジェクト計画を作成する

組織の規模に応じて、プロジェクト計画は略式のレビューになる場合もあれば、より正式な実現可能性分析になる場合もあります。しかしながら、プロジェクトの規模に関わらず、自動化の目標は明確にすることが重要です。

次の質問は、目標について考える際に役立つかもしれません:

- あなたの自動テストの定義は何ですか？
- 自動化によってどのような問題を解決しようとしていますか？
- 自動化プロジェクトに対する組織の目的は何ですか？
- テスト自動化の目的は、テスト全体の目標を裏付けるものになっていますか？
- 自動テスト計画において、手動テストはどのように適合させますか？
- テストをどの程度自動化したいと考えていますか？

現実的で測定可能な、自動化プロジェクトの「成功」の定義を作成しましょう。たとえば、 Ranorex の顧客は一般に、回帰テストの完了に要する時間が 50% 以上削減されたと報告しています。したがって、現在の回帰テストの完了に 16 時間かかっている場合、現実的な成功の定義は、「回帰テストが 8 時間以内に完了できれば、自動化プロジェクトは成功である」とすることもできるでしょう。組織にとってのプロジェクトの真の価値を明らかにするために、成功の定義を明確な数字で定義してください。この成功の定義に対する価値は、「我々のテスターが探索的テストにより時間を割けるようになる」、あるいは「手動テストは、アプリケーションが安定している (自動テストによるスモーク テストとサニティ テストに合格している) 場合にのみ実施すればよい」などとできるでしょう。

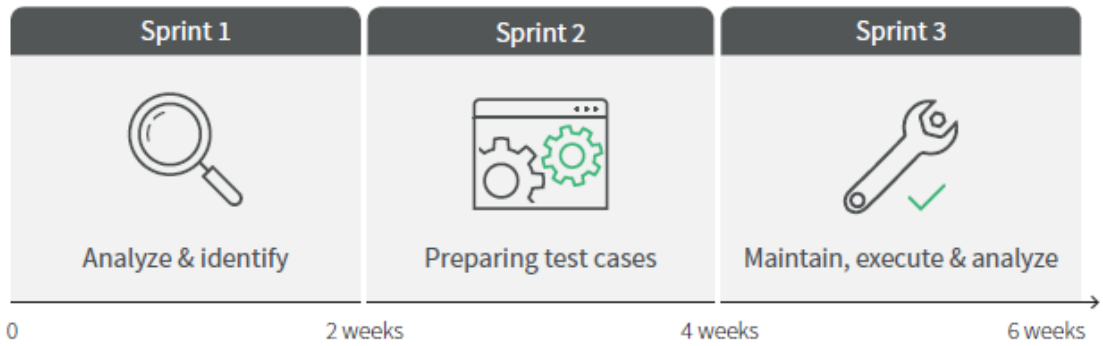
プロジェクト計画には、プロジェクトの範囲と初期予算も含めるべきです。また、以下のような情報を含めることも有効でしょう。

- 自動化プロジェクトを管理するためにどのようなメトリクスを使用するか？ (たとえば、コスト、スケジュール、コードカバレッジ、テスト実行時間、レポートされる欠陥の数、など)

- プロジェクト成果物は何か？(たとえば、プロジェクト計画自体、ツールを選択するための条件、ツールの評価結果、PoC テストの結果、自動テスト ケースの本体、など)
- 自動化プロジェクトは他の納期にどのように影響するか？
- テスト ケースの中央リポジトリをどこで管理するか？
- 自動化チームをどこに配置するか？(コラボレーションを促進するためには、メンバー同士が近接していることが理想的)
- いつどこでトレーニングを実施するか？

プロジェクト計画を作成するときは、プロジェクトに必要なハードウェア/ソフトウェア リソースを用意するための十分なリード タイムを確保してください。また、恐らくテスターが通常のスプリント テスト作業と並行して自動化タスクを詰め込むことはできない点を認識してください。代わりに、自動化それ自体を独自の開発プロジェクトとして扱うことを推奨します。自動化プロジェクトの管理においては、コラボレーションを重視した、段階的で反復的なアプローチの適用を検討してください。自動化プロジェクトも、開発と同様にスクラム プロセスで進めていきましょう。たとえば、最初の 2 週間のスプリントでは、現在のプロセスの分析、プロジェクト計画の作成、およびチーム メンバーの決定を行います。次の 2 週間のスプリントでは、自動化のためのサンプル テストケースの作成、候補となる自動化ツールの調査と選択、および PoC を実施するための環境の設定、といったことを行います。アジャイル手法は、テスト自動化においても、ソフトウェア開発と同じメリットがあります。つまり、コラボレーションを促進し、より小さな成果物をより頻繁にリリースし、要件の変化へ迅速に対応することによって、成功のチャンスを最大化します。

最後に、テストの自動化が整ったら、それを維持する必要があります。 自動テスト ケースを保守および実行して結果を分析するための長期的なリソースを必ず確保してください。



3. 適切なチームを立ち上げる

プロジェクトを成功させるには、適切なメンバーをチームに引き入れることが非常に重要です。成功する自動化プロジェクトには、1 人以上のチャンピオン (リード役) が関与しているものです。チャンピオンはテスト自動化エンジニアである必要はありませんが、多くの場合、品質保証、データベース、またはソフトウェア開発に関係するバックグラウンドの持ち主です。重要な要素は、自動化プロジェクトの推進を支援するという意欲と能力です。その他の重要なチーム メンバーとして、すべての利害関係者グループの代表が含まれます。たとえば、1 人以上の開発者、テスター、ビジネス アナリスト、およびシステム管理者などです。このチームは、目標を設定し、進捗状況をモニタリングし、戦略を調整し、最終的な解決策を承認するために、定期的に打合せを実施する必要があります。

Jason Branham は、テスト自動化チャンピオンの必要性について次のように説明しています。

> “現場において、チーム内に何らかの自動化の経験を持つ専門家が 1 人もいない場合、ほとんどのチームはテストの自動化を試みようとしません。しかし、自動化のバックグラウンドがあることは必須要件ではありません。なぜなら、手動テスターでも本格的なテスト自動化エンジニアでも、自分の資質に合ったツールセットを見つけることができるからです。”

社内の誰かがチャンピオンになる場合もあれば、外部からコンサルタントを招く方が良い場合もあります。たとえば、Simon Knight は外部の品質保証コンサルタントとして長年の経験があり、顧客が正しい解決策を正しく実行できるよう支援してきました。彼のコンサルティングは、時に既存のテスト チームのパートナーとなって、テストスクリプトを実装したり、プロセスを整えたり、ツールを使ってテスト自動化を支援したりするものでした。

同様に、Ranorex は品質管理のための独自の内部プロセスを開発するにあたり、外部コンサルタントを利用しました。Larissa Stoiser は、次のように述べています。

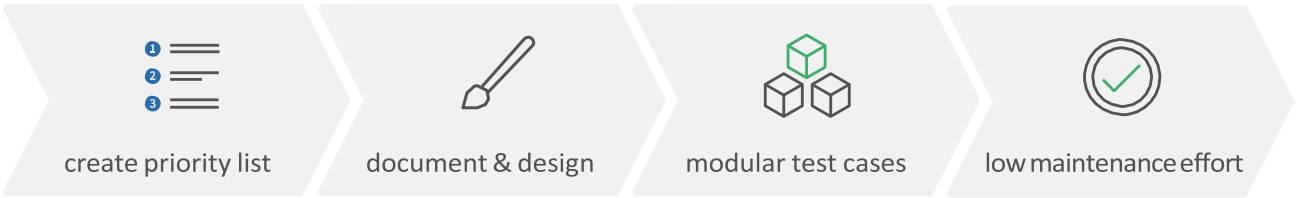
> 品質のためのツールを開発しているソフトウェア企業として、我々は品質が非常に重要であることを自然と認識していました。それでも、他の急成長中の企業と同様に、我々のプロセスには非効率的な部分があったため、品質プロセスの経験豊かなコンサルタントを外部から招きました。彼は会社全体の品質計画の作成を支援してくれました。品質計画では、各部門において「製品の品質が何を意味するのか」そして「品質計画に何を期待するのか」に関する見解を文書化しました。その後、会社は次年度の各部門の品質目標について合意しました。品質計画の初年度の後、コンサルタントとのフォローアップを実施し、プロセスの見直しと必要な調整を実施しました。最適な品質プロセスを開発する上で、外部の専門家の存在は非常に有用です。現在、我々のプロセスは非常にアジャイルでリーンです。

現在 Ranorex では、開発チームが、製品と自分たちが作成する機能に対して全責任を負います。Stoiser は次のように説明しています。

> 我々はあらゆることを考えなければなりません。顧客が望むものや我々自身のスケジューリング、そしてそのすべてに自由と責任があることが開発チームを幸福にします。チームには、主にテストを実施する、テストのバックグラウンドを持つスタッフがいますが、開発者自身もテスト プロセスに深く関わっています。我々は、コードに何かがあれば、すぐにテストを試みます。これは非常にアジャイルで短いフィードバック ループであり、これが現在のプロセスです。

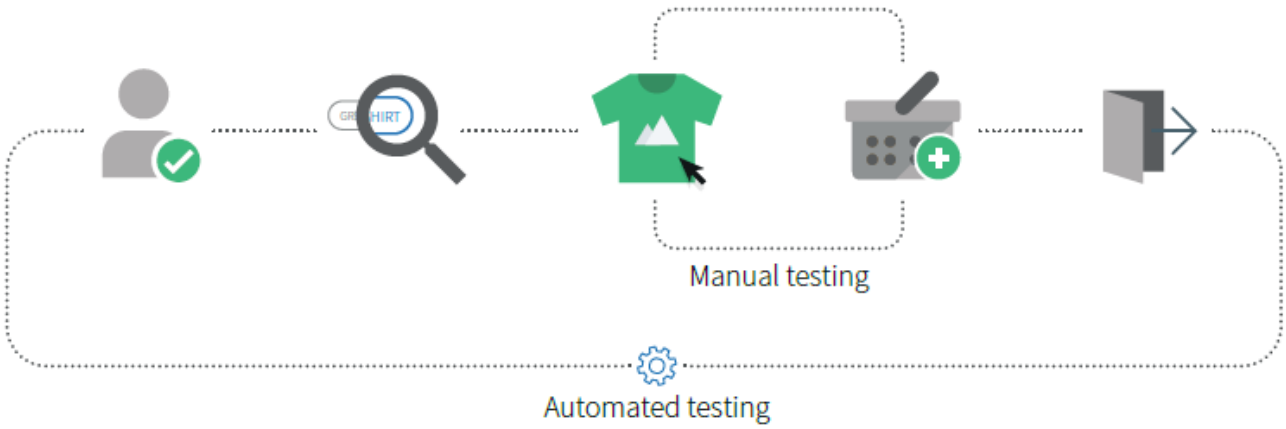
開発者が品質に対して高い意識を持ち、テスト可能なコード (たとえば、モジュール化されていること、UI 要素が静的な ID を持つことなど) を作成することを意識している場合、テストの自動化を実装することは簡単です。この意識を高めるために、自動化プロジェクトおよび自動テストの保守について、その役割と責任をチーム内で自由に定義させましょう。たとえば、誰がテストとオブジェクト リポジトリ情報を作成して保守する責任を負うのかを決定します。テストを実行するのは誰ですか？ツールの社内サポートを担当するのは誰ですか？機能横断型の実装チームを構築する主な利点のひとつは、品質に対するコミットメントを、開発者やテスターを含むチーム全員で共有できることです。

4. 自動化のためのサンプル テストケースを準備する



ツール評価の一環として、自動化するテスト ケースの優先リストを作成します。このリストに適した候補には、スモーク テスト、ビルド受け入れテスト、およびハイ リスクな回帰テストが含まれます。選択したテスト自動化ツールが、AUT (テスト対象アプリケーション) の主要機能と確実に連動するように、少なくとも 1 つのエンドツーエンドのテスト ケースを必ず含めてください。

試験的なテスト ケースをいくつか選択したら、それらが十分に文書化され、適切に設計されていることを確認します。手動テスト ケースを自動化する際には、追加情報が必要になることが一般的です。たとえば、Web ショッピング サイトにおいて、ユーザーがショッピング カートに商品を入れた際のアプリケーション動作を記述したテスト ケースがあるとします。このテスト ケースには、アプリケーションがこの時点に到達するための前提条件 (ログオンや商品の検索など) が含まれていないかもしれません。手動テスト ケースであれば、テスト担当者が指示を解釈して足りない手順を補完できるため、前提条件が含まれていなくても問題ないでしょう。しかし、自動テスト ケースでは手順を明示的に記述する必要があります。自動化するテスト ケースの前提条件、テスト ケースを実行するための操作手順、テスト データの値、実行する検証の内容、テスト ケースの実行後に必要なクリーンアップの手順、およびテストケースが成功したかどうかを判断する条件が、すべて文書化されていることが重要です。



まだテスト ケースを文書化する方法がない場合は、必ずテンプレートを作成してください。テンプレートは、スプレッドシート ファイルや Google フォームの場合もあれば、TestRail などのテスト管理ツールを利用する場合もあるでしょう。テスト管理ツールを使用すると、前提条件、テストデータ、予想される結果、推定作業量によってテスト ケースのリポジトリを管理できます。

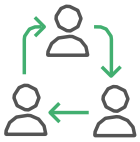
自動テスト ケースでは、前提条件とテスト データに加えて、ポップアップ ウィンドウなどの予期しないイベントへの対応方法など、エラー処理に関する規則が必要になります。これは欠陥検出機能とは別のステップです。メンテナンスの手間を省くために、必ず自動テスト ケースをモジュール化するようにしてください。これは手動テスト ケースの計画ではあまり考慮されないことです。 たとえば、ショッピング カートの手動テストにはアプリケーションの起動とログインの手順が含まれていない場合がありますが、自動テスト ケースにはこの手順が必要です。ただし、自動化した各テスト ケースにログイン手順を含めるのではなく、単一のログイン テスト ケースを作成し、それを前提条件とする他のテスト ケースから呼び出します。 このようにモジュール設計を行うことによって、テスト ケースのメンテナンスの負荷を軽減できます。

5. 候補となる自動化ツールを調査して選択する

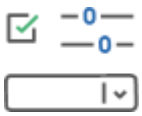
自動化ツールを検討する際、主な基準となるのは、対象のデスクトップ、Web、またはモバイル テクノロジーへのツールの対応能力です。たとえば、Web ベースのアプリケーションをテストしている場合は、クロス ブラウザー テストを実行する機能が重要かもしれません。モバイル アプリケーションの場合は、実際のデバイスとシミュレーター/エミュレーターの両方で、Android および iOS アプリケーションのテストを自動化できるソリューションが重要になるでしょう。開発テクノロジーに依存しない追加の評価基準は以下のとおりです。

標準のプログラミング言語を使用するか、ツール専用のスクリプト言語を使用するか

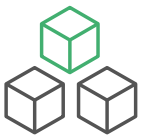
デバッグとリファクタリングをサポートする IDE



チームの共同作業を容易にするためのツール (再利用可能なオブジェクトなど)



非プログラマのためのコーディング不要な自動化サポート



フレームワークのサポート – モジュール化、データ駆動型、BDD、キーワード駆動型、ハイブリッド

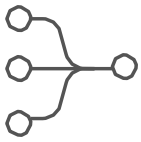
ジェイルブレイクすることなく、オートメーションを AUT から独立させることが可能

強力な UI オブジェクト識別

GUI および/またはバックエンドプロセスを介した操作処理

柔軟なライセンス オプション (マルチ ユーザー ライセンスやランタイム ライセンスなど)

他のツールとの統合



バージョン管理のサポート

ユーザー フレンドリーなインターフェイス

経験豊富な自動化エンジニアがチームにいる場合は、彼らの自動化プロジェクトに関する経験や、その分野の他の同僚の経験を当然生かすことができます。また、業界出版物、Gartner や Forrester などのアナリストによる研究、あるいは G2Crowd や Capterra などのサイトで、製品の独立したレビューを読むことも役立つかもしれません。

6. PoC を実施する

候補となる 2、3 のツールを選択したら、PoC (概念実証) を実行します。これは、通常は数週間の短期間のプロジェクトであり、実装チームが実際の環境で主要ツールを評価し、評価基準に照らしてツールを測定するものです。PoC の間、各ツールを使用して、前述の推奨事項 4 に記載されているサンプル テストケースの自動化を試みます。複数の有料ツールを検討している場合は、ベンダーの営業部門に連絡して PoC の支援を依頼してください。ベンダーのプリセールス エンジニアは通常、無料評価ライセンスの取得、使用マシンへのソフトウェアのインストール、およびアプリケーションのテスト自動化における技術的な障害の克服を支援します。

Jason Branham は、PoC を支援するための Ranorex のアプローチについて次のように説明しています。

> 私たちは単なる製品サポートではありません。顧客のインフラストラクチャ、プロセス、チームに Ranorex が適するように最善を尽くします。そのために我々は多面的な分析を行います。単に Ranorex が機能することだけでなく、Ranorex が本当に顧客のニーズに最適であることを証明するために、すべての面をチェックするのです。そのため、デモを通じてトレーニングを実施したり、環境の問題によって一部が機能していない製品をサポートしたりする場合もあります。

PoC の後、あなたのチームは採用候補のツールを推薦する準備ができているかもしれませんし、あるいは別のツールを検討するべきであるという結論に達しているかもしれません。

7. 選択した自動化ツールの環境を構築する

自動化ツールを選択したら、自動テストの実装プロセスを開始できます。テスト環境をセットアップし、バグ管理システム、テスト管理ツール、CI / CD サーバーなどとの必要な統合を行います。有料のツールを選択した場合は、ローカルマシンでテストを開発し、リモート サーバーでそれらを実行するために必要なライセンスを持っていることを確認してください。

テストの自動化はそれ自体がソフトウェア開発プロジェクトなので、テストケースに Git などのソース管理システムを使用することを検討してください。Larissa Stoiser は、Ranorex の内部的なアプローチについて以下のように述べています。

> 私たちの継続的インテグレーション開発プロセスでは、作業項目の管理のために Microsoft の Team Foundation Server (TFS) を使用しています。私たちのバージョン管理は TFS の Git によって行っています。 Ranorex から生成されたテスト コードもそこに格納します。 Git を使えば、個々のバグ修正を別々のブランチで実施できるため、テストでもアジャイルです。テスト ソースと Ranorex ソースコードを同じリポジトリに保持することは、開発の効率を上げ、適切なブランチに対して適切なテストを実行するためのアプローチの 1 つです。

さらに、テスト環境自体がテストの自動化をサポートするのに十分安定していることを確認してください。 Simon Knight は、そうでない場合に起こり得ることについて次のように説明しています。

> 私が過去に関わった あるクライアントは、サード パーティ製の分析プラットフォーム上にアプリケーションを構築していました。そこには、さまざまな既製の分析サーバーに加えて、構築中のアプリケーション、そのアプリケーションに関連するさまざまなデータベース、および他のサブシステムなどが存在していました。ただし、我々はデータベースや分析プラットフォームをテストするのではなく、構築中のアプリケーションだけをテストしていました。

> しかし、自動テストを実装しようとしていた環境では、信頼性の高い自動化を進められるだけの十分なパフォーマンスは得られませんでした。たとえば、よくタイムアウトが発生しました。そこには複数の問題が存在していましたが、いずれもアプリケーション自体に関するものではなく、完全にこれらのシステムがホストされている環境に依存したものでした。これは完全な悪夢です。自動化や環境に問題があるのか、それともアプリケーションに実際の欠陥があるのかを解明するために全時間を費やし、浪費することになります。
>
> ”

このような状況を回避するために、AUT のコンポーネントを分離し、インフラストラクチャがそのテストに対して十分に安定していて信頼性があることを確認してください。そうでなければ、あなたのチームはテストの自動化において大きな課題に直面するかもしれません。

8. トレーニングと学習曲線のための時間を取る

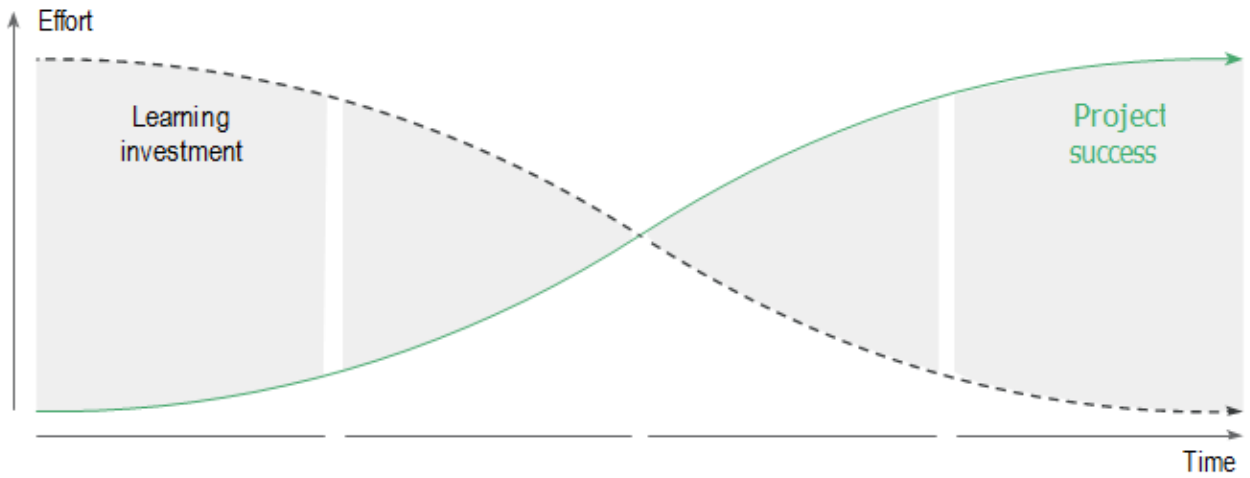
時間的なプレッシャーに直面すると、新しいツールのトレーニングに必要な時間を、短縮あるいは過小に見積もりたくなるかもしれません。しかし、自動化スタッフがツールにすでに精通しているのでない限り、十分な時間をトレーニングに割くことが重要です。トレーニングを実施する理想的な時期は、テスターが自動テスト ケースの作成を開始すると予想される直前です。なぜなら、自分の仕事とトレーニングの関連性を自動化スタッフが理解するため、モチベーションが高まるからです。トレーニング プランには、ベンダーによる対人トレーニング、オンライン チュートリアル、またはその両方の組み合わせ などが考えられるでしょう。

Ranorex のシニア セールス エンジニアである Hubert Gasparitz は、テスト自動化トレーニングに関する自身の経験を次のように述べています。

> “トレーニングが必要かどうかはチームによって異なります。新しいチームが自動テストに慣れていない場合、あるいはツールを新規に導入した場合には、トレーニングによって、大きな問題を回避し、プロジェクトをより容易かつ迅速に進めることができます。仮に、チームが過去に他の自動化ツールを使った経験があったとしても、トレーニングは有益です。過去に別のツールを経験した自動化エンジニアは、新しいツールが過去のツールと同じように機能することを期待してしまう場合があります。自動化ツールはそれぞれに特徴があるため、過去のツールの知識がかえって足を引っ張る場合もあるのです。”

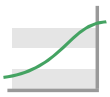
また、プロジェクトの開始時には、学習曲線についても計画を立てる必要があります。 次の図が示すように、Ranorex Studio などの使いやすいツールを使用しても、自動化プロジェクトの開始時には、テスターの生産性が低下する可能性があります。長期的に見れば、学習曲線への時間投資は、相応の効果をもたらします。

Ranorex の評価プロセスは 4 つの段階を経ます。



基本を学ぶ

知識を深める



テスト ケースを作成する

統合する

9. 文書化されたテスト ケースの自動化を開始する

テスト自動化ツールを環境内にセットアップし、ツールの使用方法についてのトレーニングを完了したら、文書化されたテスト ケースの自動化と実行を開始できます。自動テストを作成して実行する最初のサイクルは、手動テストを実行するよりも時間がかかりますが、それ以降は、リリース サイクルごとに時間が短縮されていくでしょう。以下は、テストケースの自動化を成功させるための追加の推奨事項です。

成功する可能性が高い場所を自動化する

実装の初期段階では、テスト スタッフが達成感を得て自動化ツールに対して自信を持てるように、有効で、安定した、価値の高い自動テストを作成することに注力してください。価値が高いテストに該当するのは、手動で行うと大変であったり失敗しがちであったりするテスト、繰り返しが多いテスト、比較的自動化が容易なテスト (他のシステム、外部のサービスまたはデータベースに依存しないテストなど) です。

最大の潜在的な ROI がある場所を自動化する

次に自動化の優先順位が高いのは、スモーク テスト、ビルド受け入れテスト、ハイ リスクな回帰テストです。自動化の利点を判断する簡単な方法は、どのテストが自動化に対して最も高い ROI をもたらすか、そしてどのような欠陥が本番環境に入り込んだ場合に最も大きな損害を引き起こすかを、見極めることです。リスクに応じたテスト ケースの優先順位付けの詳細については、「Ranorex Guide to User Interface Testing」の Prioritizing test cases for risk-based testing セクションを参照してください。自動化に適したその他の候補としては、データ入力を必要とするテスト、クロス プラットフォームテスト、複数フィールドの検証を伴うテスト、および CRUD (作成、取得、更新、削除) アクションがあります。より難易度が高く自動化が困難なシナリオの手動テストを実施できるよう、繰り返しの多い退屈なシナリオを自動化してそのようなテストからテスターを解放しましょう。データ駆動型にするなどによって、既存の手動テストをよりスマートにできる箇所がないか探してください。

保守できるよう自動化する

自動化するときは、再利用可能なユニット、モジュール、またはメソッドを使用して、テストを互いに独立させます。テスト ケース間でテスト ステップのコピー & ペーストを行ってはいけません。各テストの後にテスト対象アプリケーションをクリーンアップするステップが自動化に含まれていることを確認してください。

現実的になる

すべてを自動化することはできません。テストによっては、自動化に非常な労力を必要とするため、手動でテストする方が現実的なものもあります。Jason Branham がその一例を紹介します。「最近の話ですが、POS の精算処理機能をテストした例がありました。テストの一部は手動であり、その中にはペイメント カードをスワイプするテストもありました。このケースでは、その機能を自動化しようとするよりも、物理的にカードをスワイプする方が簡単でした。たとえば Captcha への応答のような、ユーザーが手動で何かをしなければならない操作は、ロボットではないことを証明するためのものです。つまり、それらは自動化をブロックするために配置されているのです。」ケースごとに適切な自動化のレベルを選択することは、「現実的になる」ことの一部です。たとえば、単体テストまたは統合テストによってインストール テストをより効率的に実施できる場合、GUI テストでインストールをテストしてはいけません。

キャプチャー/再生機能を使用して自動テストを迅速に作成し、時間を節約する

Jason Branham は、次のように説明しています。「Ranorex の顧客の中には、IDE 上ですべてを手作業でコーディングする人もいます。しかし、Ranorex Studio のレコーディング/再生機能は、実際はとても有能です。手動テスト ケースを手元に用意し、通常操作する際と同じようにテストをレコーディングします。それが終わると、Ranorex による自動化が完了しています。それは完璧に動作するかもしれませんし、あるいはユーザー コードを追加してオートメーションを強化する必要があるかもしれません。しかし、すべてを一からコーディングするよりも早く自動テストを開始することができます。」

テスト可能なアプリケーション コードを作成するよう開発者に奨励する

UI テストの場合、これは自動化 ID を設定することを意味します。ソフトウェアの構造化とモジュール化が進んでいればいるほど、自動化チームがテストを作成することが容易になります。

テストの失敗を処理する

テストの実行で 1 つのテストが失敗したからといって、他のすべてのテストも失敗して欲しいわけではありません。選択したツールに、テスト ケースが失敗した後の動作をカスタマイズできる設定があれば、利用しましょう。たとえば、テスト ケースの次の繰り返しを続行する、次の兄弟テスト ケースにジャンプする、あるいは失敗メッセージを出してテスト ケースを中止する、といった設定です。また、テストが失敗したときに、問題がアプリケーションにあるとすぐに推測しないでください。個々の失敗を分析して、自動テスト ケース、環境、またはテスト対象アプリケーションのどこに問題があるのかを判断しましょう。

テスト ケースのメンテナンスを計画する

テスト対象アプリケーションに変更があった際は、メンテナンス期間中に (たとえば次のスプリントの初期段階などに) 変更への対応をテスト計画に組み込んで、新しいテストケースを自動化します。

10. 定期的にプロセスを見直し、必要に応じて調整する

実装の初期段階が完了したら、プロジェクト チームはプロジェクト メトリクスをレビューし、チームのパフォーマンスを分析します。プロジェクト レビューの一環として、組織化とコミュニケーションの効果を評価しましょう。テスト自動化プロジェクトがチームのテスターの役割にどのような影響を与えたか、そして継続的な自動化が、さらにどのような変化をもたらす可能性があるかを検討してください。

機能横断型チームの利点の 1 つは、開発者にならなくてもテスターがソフトウェア開発について理解を深めることができることです。Larissa Stoiser は、機能横断型チームが Ranorex にもたらした変化について次のように述べています。

> “私たちの主な目標は、QA と開発をより緊密にし、開発プロセスの最初から QA を巻き込むことでした。QA スタッフには、たとえ開発者にならなくても、少なくともテストコードを記述し、単体テストを実行し、開発者をより理解するよう奨励しています。”

2017 年に Ranorex が実施した、アジャイル チームにおけるソフトウェア テスターの役割の変化に関する調査結果を以下に示します。最初のグラフは、アジャイル チームの約 37% のテスターが、ユーザー要件と受け入れ基準の定義を支援していることを示しています。これらは一般的に、ウォーターフォール開発でビジネス アナリストが担当する業務です。

2 番目のグラフは、アジャイル チームにおけるテスターの責任範囲を示しています。50% 以上が自動テスト ケースの作成を担当し、22% が開発者と協力して単体テストおよびインターフェイス/ API テストを実施しています。これらの調査結果は、機能横断型のチームと組み合わされた自動化プロジェクトが、テストについてのみ深いスキルを持っていた「I 字型」チームメンバーから、テストスキルと開発に関する幅広いスキルの両方を併せ持った、さらに価値のある「T 字型」チームメンバーへと成長させたことを示しています。

Ranorex による調査: QA/テスターの計画責任

アジャイル チームのテスターは定期的にどのような計画作業を行いますか?

- 78% シナリオ、ケース、スクリプトといったテスト成果物を作成する
- 67% テスト計画の作成に関わる
- 65% 機能テスト/非機能テストをするべき箇所を特定する
- 51% テスト自動化フレームワークをセットアップする
- 36% テスト可能なユーザー要件の定義を支援する
- 37% 受け入れ基準を定義する
- 22% リスク分析に関わる

Ranorex による調査: QA/テスターのプロジェクト責任

アジャイル チームのテスターは定期的にどのようなプロジェクト タスクを行いますか?

- 54% 自動化 GUI テスト ケースを作成する
- 53% 完了したテストを分析する
- 52% 探索的 GUI テストを実施する
- 50% 回帰テスト用のテストを文書化/アーカイブする
- 46% スクリプト化された GUI テストを実施する
- 34% カバレッジなど、テスト後の成果物を作成する
- 22% 開発者と連携して単体テスト、インターフェイス/API テストを実施する
- 11% テスト技法について開発者をコーチする

最初の実装段階の後、自動化の品質をレビューすることも有益でしょう。たとえば、Ranorex は、契約の最初の 3 か月間、Enterprise の顧客に対してプロジェクト レビュー サービスを提供しています。

Ranorex のカスタマーサポート担当ディレクターである Bernhard Seunig-Karner は、レビューの仕組みを次のように説明しています。

> 顧客から Ranorex ソリューションが送られ、希望する解決策に関する情報が提供されます。これには、テスト対象アプリケーションについての説明、テスト アプローチは何か、Ranorex から何を得たいか、どのような利益をもたらすべきか、といった情報が含まれます。我々は、顧客の自動化プロジェクトにある非効率性を調査します。これは人間によるレビューと自動化されたレビューの両方で実施します。自動レビューによって潜在的な問題のリストが作成されると、このリストをエンジニアが分析して、問題が本物であるか確認します。レビューが完了すると、より速く、安定して、良いテスト結果を提供し、そしてより保守しやすくするために何を改善できるかについて、顧客にフィードバックします。

Seunig-Karner は、プロジェクトの非効率性のいくつかの原因についても説明しています。

> ほとんどの場合、問題はプロジェクトの構造にあります。Ranorex 固有の問題ではありません。そもそも、誰でも簡単にその構造をメンテナンスできるかという問題です。たとえば、同じタスクを実行する複数のテスト モジュールは、メンテナンスの問題を引き起こします。Ranorex はメンテナンスを最小にするために再利用可能なモジュールを提供していますが、ユーザーがこの機能を利用する必要があります。我々が時折見かけるもう 1 つの問題は、オブジェクト リポジトリの構造にあります。冗長性を排除するには、オブジェクトの識別を適切にメンテナンスすることが重要です。レコーディング モジュールとオブジェクト リポジトリをチェックし、UI 要素が適切に構造化されて正しいこと、および冗長性がないことを確認しましょう。同様に、実質は単一のアクションであるものを、レコーディング モジュール上で複数のステップに分けることが可能です。最大の効率を得るためには、アプリケーション コードと同じように、自動テストもリファクタリングする必要がある場合があります。

Seunig-Karner による最後の推奨事項は、自動テスト レポートへ出力される詳細情報の量に関するものです。「夜間にテストを実行する場合は、すべてのアクションをレポートに記録する必要はないかもしれません。レポートが膨大になり、大量のデータがあるためにレポートを開くことが難しくなる可能性があるためです。分析が難しくなるのはもちろんのことです。Ranorex には、失敗したアクションのみをテスト実行レポートへ出力する設定があります。モジュールのテストが成功した場合、そのモジュールはレポートに記録されません。これは、テスト実行レポートをより便利にすることができるカスタマイズ設定のほんの一例です。」

最後に

成功の可能性を最大化するために、テスト自動化プロジェクトを、開発プロジェクトとは別の取り組みと見なすことを推奨します。機能横断的なチーム コラボレーションに重点を置きつつ、テスト自動化プロジェクトを管理するために段階的で反復的なアプローチを行ってください。PoC を実施して、実際のテスト環境およびテスト プラットフォーム上で、選択した自動化ツールが機能することを確認します。最も簡単に自動化でき、安定したテストを行えて、最大の ROI を得られるテスト ケースの自動化に集中しましょう。最初の実装サイクルが終わったら、プロジェクトとプロセスをレビューして、改善の余地がある箇所を特定します。そして、最後のヒントになりますが、自動化のメリットを最大化するために、手動テスト用のテスト データのロードや運用環境のモニタリングなど、他のタスクでも自動化ツールを活用することを検討しましょう。

テスト自動化プロジェクトを開始する準備はできましたか？ Ranorex が皆さんの作業をサポートします

Ranorex のテスト自動化エンジニアは、自動化プロジェクトを成功させるためのお手伝いをします。詳細については、セールス チームにお問い合わせください。 または、独自の PoC を社内で実施する準備ができている場合は、今すぐ Ranorex の無料試用版をダウンロードして、そのテスト自動化のパワーを体験してください。

寄稿者:

Jason Branham, セールス エンジニアリング マネージャー, Ranorex Inc. (米国)

Jason は 2014 年から Ranorex に勤務していますが、ソフトウェア自動化とリレー、スイッチ、ロボットなどのハードウェア自動化の両方の経験を有し、15 年以上にわたり自動化に携わってきました。彼は顧客と協力して自動化の夢を実現することを楽しんでいます。

Hubert Gasparitz, シニア セールス エンジニア/チーム リーダー, Ranorex GmbH (オーストリア)

Ranorex のプリセールス エンジニアとして、Hubert は自動化ツールの選択、テスト自動化プロジェクト、テスト環境のセットアップ、自動化の効率最大化を支援します。電気工学、ソフトウェア開発、および経済学のバックグラウンドを有しています。

Simon Knight, TestRail 製品マネージャー, Gurock Software GmbH

Simon はこの 10 年間、テスター、テスト リード、テスト マネージャー、テスト自動化エンジニア、そしてテスト コンサルタントとして活躍してきました。現在、彼は Gurock の TestRail 製品マネージャーとして、TestRail 製品のロードマップを策定し、開発チームと連携して高品質な製品を提供する責任を負っています。

Larissa Stoiser, シニア QA エンジニア/QA チーム リーダー, Ranorex, GmbH (オーストリア)

グラーツ工科大学を卒業し、計算数学の修士号を取得した Larissa は、Ranorex 品質保証チームを率いています。彼女は ISTQB 認定テスターであり、プロの写真家でもあります。

Bernhard Seunig-Karner, Ranorex, カスタマー サポート ディレクター

Bernhard は、C# と PHP の両方でソフトウェア開発者としての経歴を持ち、ファームウェア開発と学士号取得のプロジェクトにおいてこれらの言語を使用してきました。Ranorex に入社する前は、複数の企業や病院でソフトウェア インテグレーションを担当していました。現在、彼はカスタマー サポートチームを率いて、顧客が質の高いサポートを受けられるようにし、Ranorex 開発チームに顧客のニーズに関するガイダンスを提供しています。