VES

ツール導入だけでは成功しない

テスト自動化で陥りやすい失敗パターン

～テスト設計見直しから定着支援まで～

株式会社ヴェス
https://www.ves.co.jp

テスト自動化市場の動向

■テスト自動化市場の動向

市場は年率10～15%超で成長

- グローバルのテスト自動化市場は2025年時点で約250～380億ドル規模
- 2034年には約730～1,380億ドル規模へ拡大予測
- 年平均成長率（CAGR）は約10～15%で推移

POINT 01

開発スピードの加速

Agile・DevOpsの普及
リリース頻度の増加
CI/CDの定着

POINT 02

品質要求の高度化

マルチデバイス対応
セキュリティ要件強化
障害発生時のビジネス影響拡大

POINT 03

人材不足

QAエンジニア不足
テスト工数増加
属人的な検証体制

昨今のトレンド – テスト自動化×生成AIの活用

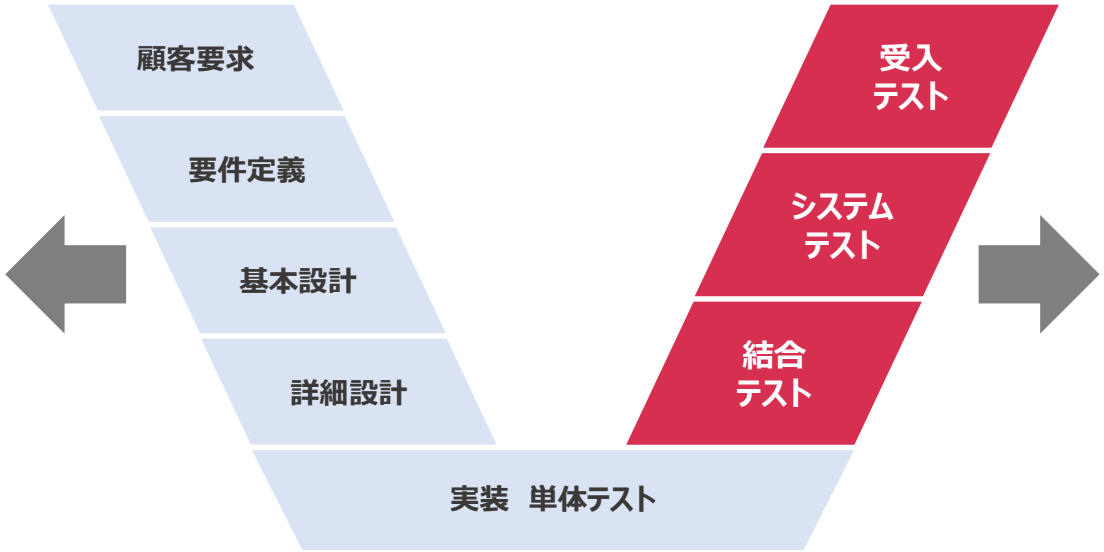
■トレンド
生成AI（ChatGPT等）を活用し、**上流から下流まで生成AIで、品質向上、コスト削減、効率化を図れる**
⇒自然言語（日本語や英語など）で記述されたテスト要件やシナリオから、自動的にテストケースやテストスクリプトを生成が出来るため、誰でもテストスクリプトの作成と実行が可能となる

■ヴェスとしての取り組み
法・規格（金融法など）に則ったテスト観点を抽出 × 生成AI活用したテスト自動化

■AIノーコード開発プラットフォーム（例：Base44）

ユーザーが「予約管理機能のあるアプリを作りたい」といったリクエストをすると、
画面構成/データベース/認証設定などを自動的に設計し、数時間で動作するパイロット版アプリ構築ができるツール

■主な特徴
- 自然言語操作で開発を自動化し、要件定義から試作まで完結
- 短時間でパイロット構築が可能（半日以内で動作試作を実現）
- AI連携機能により、ChatGPT・Claude・Geminiなどの外部APIを簡単に統合できる
- 工程の効率化を支援し、設計・試作・レビューを1つの環境で完結させられる



■生成AIの適性を活かし、複数ツールを組み合わせて自動化を実現

テストケース作成、テスト実行、テスト管理など、それぞれの役割で最適な生成AIを活用し自動化を実現

※詳細は後述にてご紹介

例：
PlayWrite × MCP × Claude

テスト自動化 – As-Is,To-Be

テスト自動化実現に向けた、As-Is,To-Be例を記載いたします。品質向上、コスト削減、効率化を目指すにあたり、テスト自動化による高いROIの導出がカギとなります。

As-Is	• ヒューマンエラーの多発 • テストの網羅性不足 • リリース後の不具合発生	• 手動テストの工数・コスト増大 • メンテナンスコストの増加 • テスト人材のリソース不足	• 開発サイクルの遅延 • CI/CD導入の遅れ • テスト資産の非効率な運用
Action	**テスト設計および実行の自動化の実現**		
To-Be	**品質向上** • **自動化で一貫性のあるテスト** • **テスト網羅性の向上** • **早期バグ検出**	**コスト削減** • **手動テストの負担軽減** • **中長期的な工数削減** • **メンテナンスコストの低減**	**効率化** • **CI/CDとの結合** • **テスト資産の再利用** • **継続的な改善サイクル**

テスト自動化で陥る失敗パターン10選

テスト自動化実現において、陥りやすい失敗パターンを記載いたします。特に見落とされやすいポイントにハイライトを入れさせていただいております。以下の課題に対して弊社としては、テスト設計の見直しからご支援させていただき、貴社での内製運用に向けてプロジェクト中に貴社にレクチャーおよびハンズオン形式で伴走型トレーニングをさせていただきます。

#	要因	内容
1	自動化対象の選定ミス	ROIの低いテストや変更頻度の高いテストを自動化してしまう
2	**テスト設計品質の不足**	**手動テストをそのまま自動化し、保守不能になる**
3	UIテストへの過度な依存	UI変更で大量のテストが壊れる
4	**保守コストの過小評価**	**作ることばかり考え、維持費を見積もらない**
5	テストデータ管理不足	データ準備やクリーンアップができない
6	開発チームとの連携不足	テスト担当だけで自動化を進める
7	**スキル不足**	**プログラミングや設計スキルが不足**
8	CI/CDとの統合不足	自動実行されず利用されない
9	不安定なテスト（Flaky Test）	実際は正常なのに失敗する　例：グラフの確認など
10	経営層・管理層の期待値ミス	「全部自動化できる」と考える

【弊社としてのご提案方針】
高いROIを導出するために、**テスト設計の見直し**からご支援させていただき、弊社支援内容やナレッジについても、貴社内での内製運用に向けて、**レクチャーおよびハンズオン形式で伴走型トレーニングをさせていただきます。**

テスト自動化成功に向けたロードマップ

テスト自動化成功に向けた進め方のポイントとしては、現状分析・要求整理で課題の棚卸を行い、その後PoCを行うことで、効率的・効果的にテスト自動化本格導入を実現いたします。
また、弊社がプロジェクトから脱退後、貴社内での内製運用垂直立ち上げに向けた、プロジェクト伴走型のトレーニングを並行してご支援することも可能です。

フェーズ	現状分析 要件整理	PoC （試験導入）	本格導入	運用定着・改善
目的	現在のテストプロセスを分析し、 自動化の目的・対象・優先度を明確にする	限定された範囲で自動化を試し、 技術的・運用的な課題を洗い出す。	自動化を主要機能全体に展開し、 開発プロセスに組み込む	自動化の運用を定着させ、 品質データを継続的に改善に生かす
主な活動	• テスト工程・工数・担当範囲のヒアリング • 自動化可能な領域の洗い出し（回帰テスト・主要フローなど） • 環境（テストデータ、ブラウザなど）の調査 • テスト設計の見直し	• 自動化対象（例：ログイン機能や検索機能）を限定 • 自動テストコードを作成し、実行確認 • 成果・課題をドキュメント化	• 自動テストスイート構築 • レポート生成・通知設定 • チーム内レビュー・自動テスト文化の浸透	• スクリプト保守 • カバレッジの拡大・追加自動化領域の選定 • 定期的なメトリクス評価 • チーム教育・ナレッジ共有
想定成果物	• テスト自動化戦略書 • 自動化対象リスト • 見直し後のテストケース	• 初期テストスクリプト • 改善提案書 • PoC実施レポート	• テスト運用ガイドライン	• 運用ルール • 品質指標レポート（月次・スプリント別） • 継続改善計画書

貴社メンバーの方にもプロジェクト参画いただき伴走型のレクチャー支援を実施
⇒ 弊社がプロジェクト脱退後、貴社内での内製運用垂直立ち上げの実現

伴走型レクチャー コンテンツ概要

No-Codeツールでのテスト自動化を見据えた教育コンテンツの例を以下に記載いたします。貴社のご状況やご検討方針に合わせて柔軟に教育コンテンツを採択させていただきます。

項目	Step1　基礎理解	Step2　テスト基礎	Step3　No-Codeツール基礎 & 実践	Step4　運用保守
概要	テスト自動化の前提知識を 身につけるステップ	テスト設計の基礎知識を 習得するステップ	No-Codeツールを使った実践的な 自動化スキルを身につけるステップ	自動テストを継続的に運用していく スキルを身につけるステップ
コンテンツおよび内容	**テスト自動化の目的** └「決まったことを何回も正確にやる」という自動化の本質的な価値を理解する **自動化できる/できないもの** └自動化に向いている領域と効果が低い領域の見極め **手動テストとの違い** └手動テストとテスト自動化それぞれの特性を比較 **テストピラミッド** └どのテストレベルでどれくらい自動化を実施すべきかの考え方	**テストケース作成の観点と設計原則** └自動化テストに必要な観点と設計の原則 **正常系/異常系、境界値、同値分割** └代表的なテスト技法の具体的な記載例 **リスクベーステスト** └リスクに基づいたテスト優先度の考え方 **E2E/API/Unitの違い** └各テストレベルの役割と使い分け	**基本操作（Recorder利用）** └No-CodeツールのRecorder機能を使ったテスト作成（初級・中級） **UI要素理解** └Locator概念、 id/data-testid の使い方、XPathの注意点 **待機制御（Wait）** └テスト実行時の待機制御における問題と対策 **検証（Assertion）の重要性** └テスト結果を正しく判定するためのアサーションの考え方 **テストデータ管理の重要性** └テストデータの管理方法とその重要性 **壊れにくい自動化設計** └メンテナンスコストを抑える堅牢な設計手法	**運用保守とは** └自動テストの運用保守の全体像 **テスト独立性** └テスト同士が互いに影響しない設計 **再実行戦略** └失敗時の再実行の考え方と戦略 **エビデンス取得** └テスト結果のエビデンスを適切に取得する方法 **ログ確認** └実行ログの確認・分析方法

ヴェスが対応可能なテスト自動化ツール例一覧/比較表

弊社にてご支援可能なテスト自動化ツール例を以下に記載いたします。実際に貴社向けに導入する際のコスト感などは、別途お打合せ後に御見積差し上げます。

区分	ツール名	実行可能なテスト				コスト					特徴
		Webアプリ	モバイルアプリ	APIテスト	デスクトップアプリ	メンテナンス	学習コスト	実行速度	費用	環境構築	
No-Code	MagicPod	○	○			High	Low	Slow	Middle	無	• キャッチアップの難易度が比較的易しい • **Web・モバイルアプリの自動化のトライアル向き**
	Autify NoCode	○	○			High	Low	Slow	Middle	無	
Low-Code	mable	○	○	○		Middle	Middle	Middle	High	有	• No-Codeツールと比較すると、キャッチアップ、メンテナンス、ランニングコストがかかる • **自動化実現のカバレッジもある程度確保できるため、大規模な自動化や中長期的な自動化運用向き**
	UFT One	○	○	○	○	Middle	Middle	Middle	High	有	
	Ranorex	○	○		○	Middle	Middle	Middle	High	有	
Pro-Code	Selenium	○		○		Low	High	Quick	Low	有	• キャッチアップやメンテナンスのコストが最もかかる • **自動化にて実現のカバレッジが広い**
	Appium		○	○		Low	High	Quick	Low	有	
	Playwrite	○				Low	High	Quick	Low	無	

生成AIを活用したテスト自動化のイメージ

昨今のトレンドはテスト自動化のみならず、生成AIを活用した自動化になります。
実際のツール（Playwrite×MCP×ChatGPT/Claude）をもとに以下のイメージを実現し、品質向上、コスト削減、効率化を実現しております。

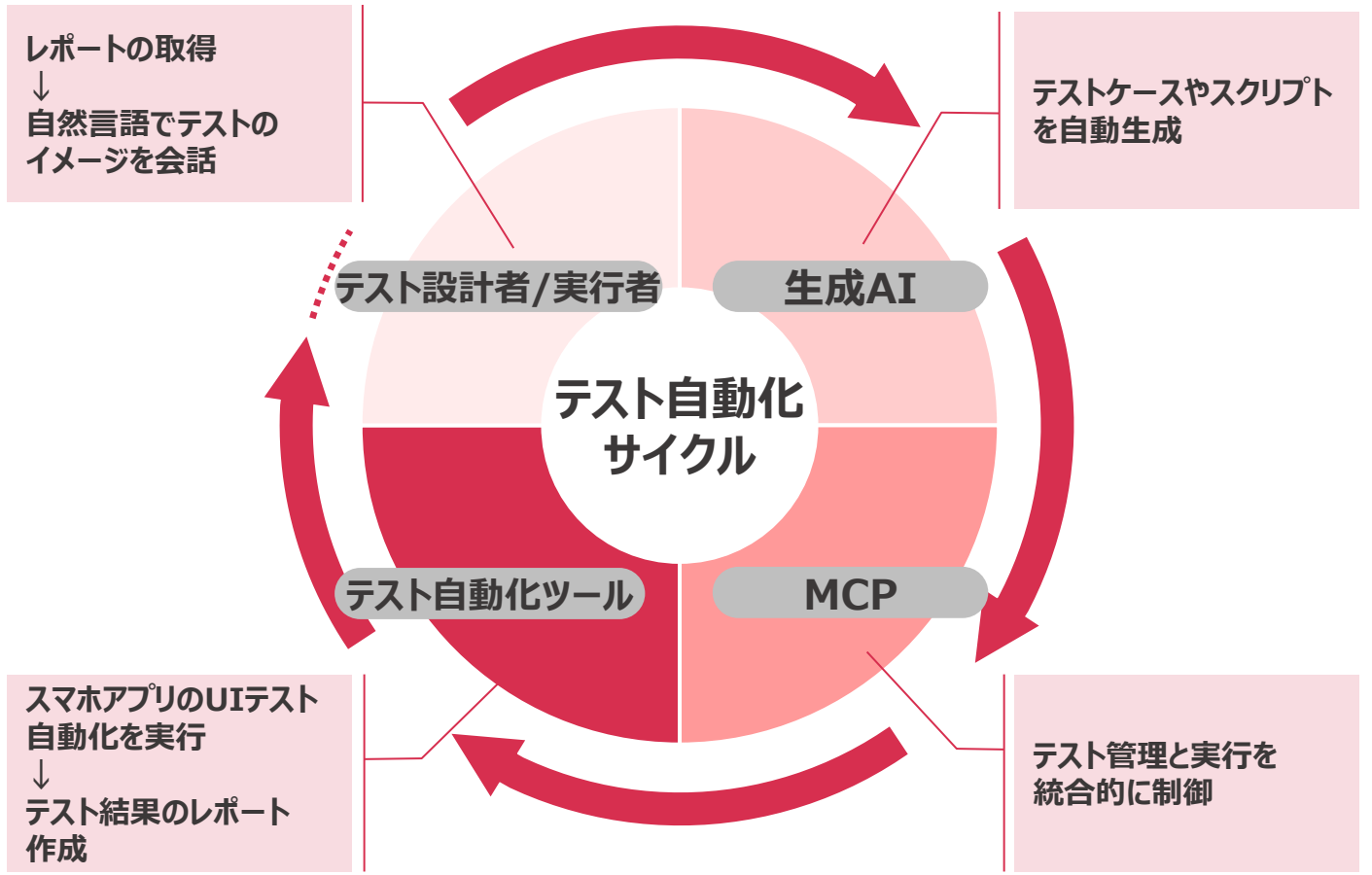
目的

Webアプリの品質向上と
リリースサイクル短縮のためのテスト自動化推進

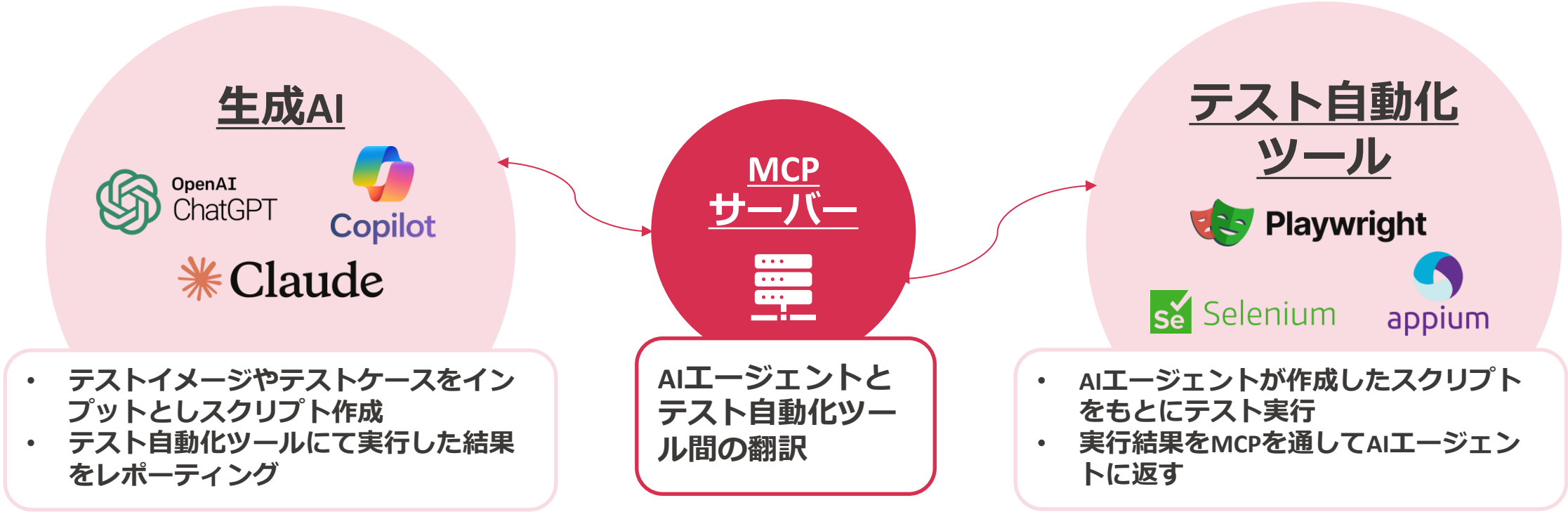
- 生成AI（ChatGPT等）を活用し、
  テスト設計・実行・運用保守の効率化実現
- 対話型のテスト自動化実装により、**誰でも対応可能**

ツールおよび役割

- **生成AI（ChatGPT / Claude）**
  テストケース生成・スクリプト自動化支援
  自然言語からPlaywright用テストコード（TypeScript/JavaScript/Python等）を自動生成可能
- **MCP（Mobile Center Platform）**
  テスト管理・実行オーケストレーション
  CI/CDと連携し、**テストの自動実行・管理も容易**
- **テスト自動化ツール（Playwrite）**
  Webアプリの**UI自動テスト実行に特化**したツール
  ※Appiumのようなモバイルネイティブアプリの自動化は対象外



生成AI × テスト自動化ツール – システム構成



MCPの仕組みと構成

コンポーネント	役割
MCP Client（ホスト）	AIを動かす環境。ChatGPT、VSCode拡張、CLIなどが該当
MCP Server（ツール側）	実際に操作する対象。Playwright、Git、DB、ファイルなど

あとがき

弊社はソフトウェア第三者検証専門会社として、お客様の製品の品質向上の為に必要な検証ソリューションを
今後も提供して参ります。

本資料をご閲覧いただいた上で、お客様のより良い製品・サービス開発の一助として、
弊社の検証ソリューションをご活用いただけますよう、ご検討のほど、よろしくお願い申し上げます。

【お問い合わせ】　株式会社ヴェス　営業本部

Mail : **e-sales@ves.co.jp**

TEL : 03-6277-0440（代表）

FAX : 03-5794-3742

ヴェスのホームページは
こちらから
https://www.ves.co.jp