

組込み系版 現場の失敗から学ぶ自動テストの設計プロセス

株式会社AGEST QA事業本部
アドバンスドテストソリューション部
アドバンスドソリューショングループ
○ 林 尚平

- 概要
- 組込み系製品の自動テストの必要性
- 組込み系製品のテストの特徴
- 自動テストの導入を成功させる4つのプロセス
- TestArchitectを使用した組込み系自動テスト
- まとめ

組込み系の自動テストは評価対象に合わせた自動化環境を構築する必要があるため、業務系に比べて難易度が上がります。

自動テストの知見がなく中で進めた結果、導入に失敗し時間と費用が無駄になることがあります。

実際に現場で起きた問題を踏まえ、成功するための組込み系の自動テストの設計プロセスを説明します。

■組み系製品の自動テストの必要性

業務系システムと同じく組み系製品も品質確保とリリース期間短縮などを目的とした自動テストの導入検討が行われています。

組み系は派生機種が多い

製品A1 → 製品A2 → 製品A3

製品A1 → 製品B1 → 製品B2

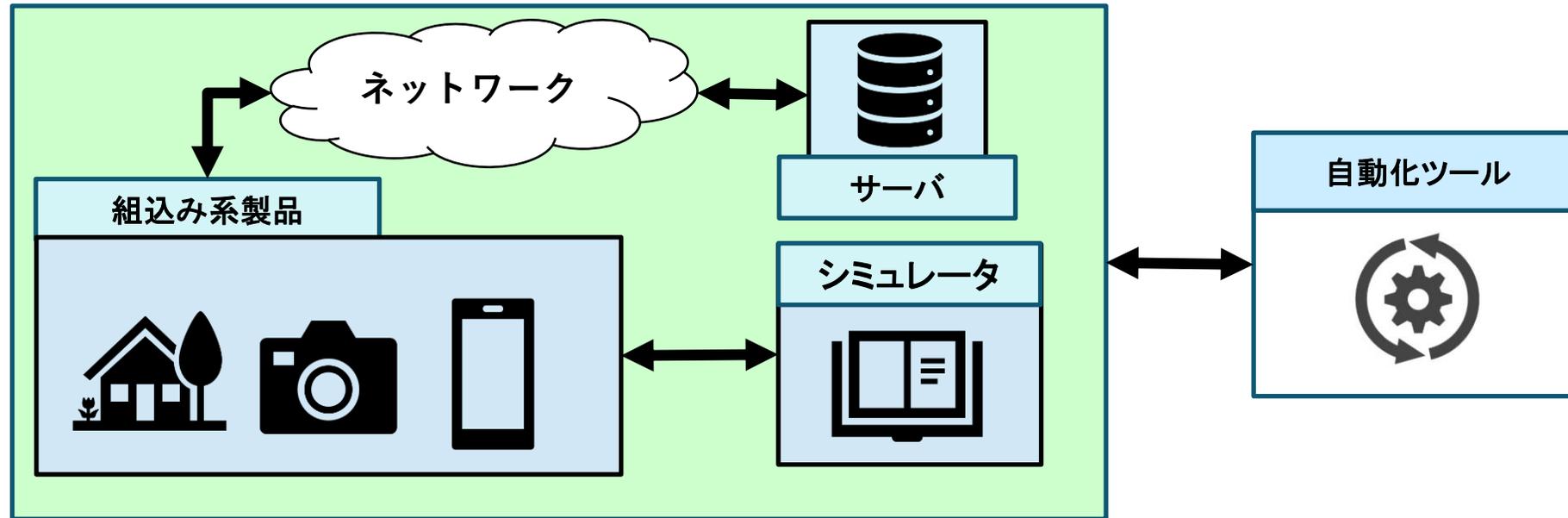
製品B1 → 製品C1

派生機種が多くソフトリリースとなった際は評価工数が膨大になる

安心・安全にかかわる機能もあるため、評価は手が抜けない

製品サイクルが比べて長い

■組込み系製品の自動テストの特徴



組込み系のテストは実製品を使わずシミュレータ機器を使用することがあります。
 自動テストを行う際には業務系システムと異なりPCとテスト機材との連携が必要です。

組込み系自動テストの導入プロセスを紹介します

- ①計画 : 方針検討、テスト分析、
自動化ツールの構成検討、自動テストの計画作成
- ②設計 : 自動化機能の実装、自動化範囲の検討
- ③実施 : スクリプト作成、実行確認
- ④振り返り : 実績検証と改善

このプロセスを実際にあつた現場の失敗事例を踏まえ紹介します。
※全ては説明できないのでポイントを絞って説明します。

自動テストの目的、方向性を決め成功基準を作ります。
その方向性で自動化した場合の成功基準を設けることで
やるべきことを明確にします。

以下の成功基準は数十機種種の製品のリリース確認の効率化を想定した
「**長期運用できる自動テスト**」を目標とした成功基準になります。

| No | 成功基準 |
|----|---------------------------------|
| ① | 1回の実行で手動に比べ最低30%以上の効率化/工数削減ができる |
| ② | 必要な自動化するテストが検討できており自動化できている |
| ③ | シナリオが連続で正しく実行出来ること |
| ④ | 結果の誤判定などなく全て正しく実行出来ること |
| ⑤ | 実行から結果確認まで自動化できること |
| ⑥ | メンテナンスを考慮した設計ができていること |

【①計画】テスト分析

組込み系の自動テストでは業務系と異なり、自動化ツールにテストに必要な機能が揃っていないことが多い。必要な機能を作成するため、テスト内容を理解しテストに必要な機能や手順などを把握することが重要です。

テストでの実運用での使われ方などを把握し、運用を簡素化するための自動化環境にする必要があります。

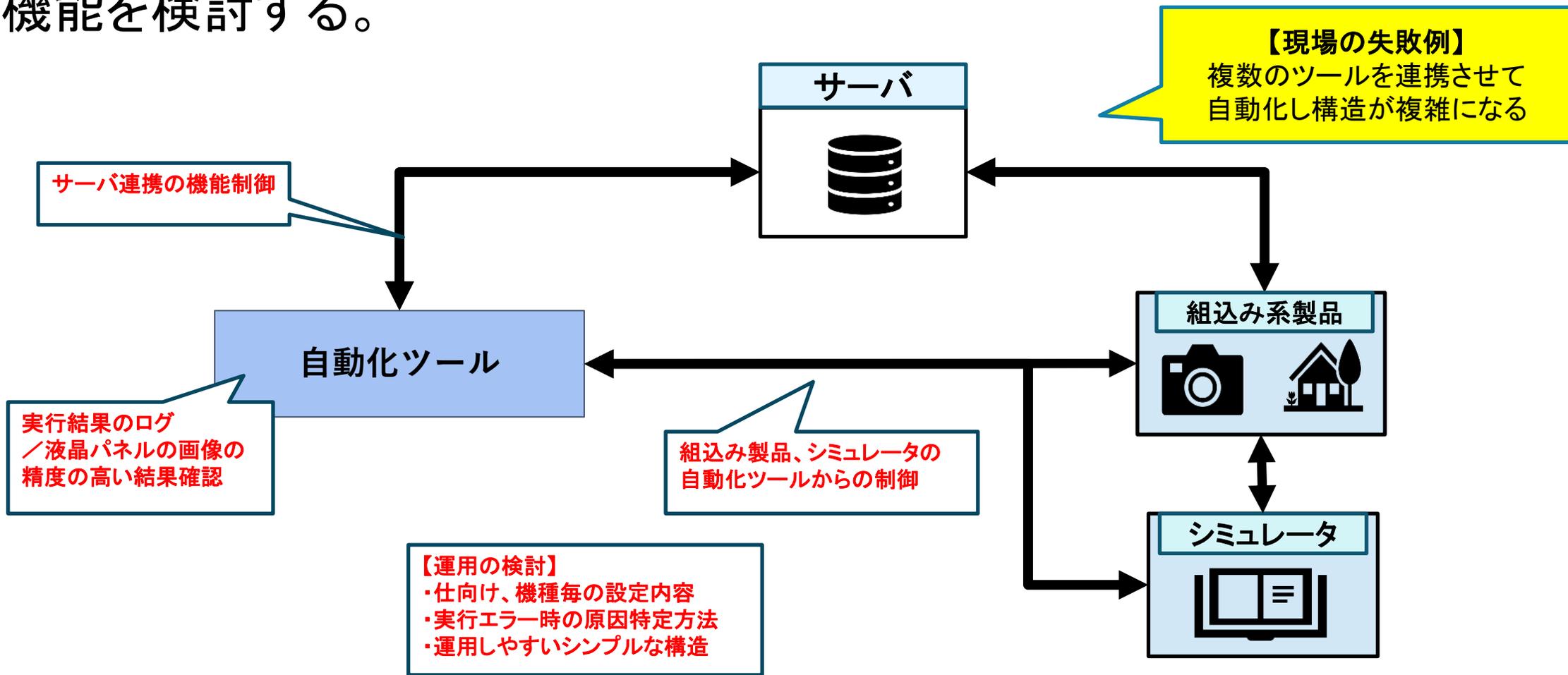
【現場の失敗例】
運用する機種数が
実運用で想定を超えた

【現場の失敗例】
テストを分析しない

| No | 内容 |
|----|-------------------------------------|
| 1 | テスト内容の調査(条件設定、手順実施、結果確認やテスト機能の洗い出し) |
| 2 | テスト機材の調査(テストツール、シミュレータなど) |
| 3 | 実運用のイメージ(実行タイミング、派生機種数など) |
| 4 | 設定内容の調査(機種の区別、機材の連携) |

【①計画】自動化ツールの構成検討

分析した内容から自動テストの構成検討を行います。
 必要なテストを自動化を実現することができる自動化ツールと
 必要な機能を検討する。

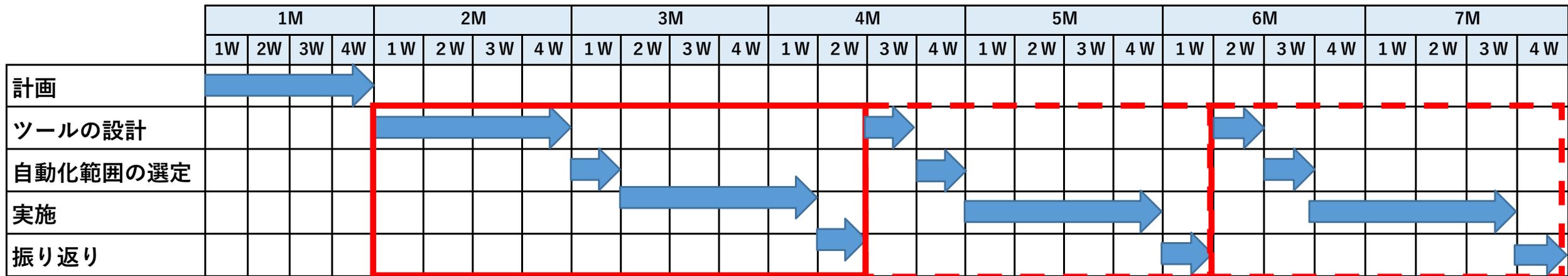


組込み系の自動テストでは実際にスクリプト実行を行った後になってから、数多くの問題が判明することがあります。

【現場の失敗例】
 一度にスクリプトをすべて作成した後、発覚したミスの修正工数が膨大になり、修正できなくなった

- ・シミュレータなどテスト機材との連携ミス
- ・結果確認の精度の不安定さ
- ・運用時のエラー発生時の原因特定の困難さ

一度に広範囲を自動化する計画ではなく、**スモールスタートで問題を潰すことが重要**



分析した内容から自動テストの構成検討を行います。
 必要なテストを自動化を実現することができる自動化ツールと
 必要な機能を検討する。

【現場の失敗例】
 実施エラー時に
 ログ出力がされず
 原因特定できない

| No | 内容 | 設計内容 |
|-----|------------------------|--------------------------|
| (1) | 機能の追加：製品制御、シミュレータ制御 | ラズパイを使った機能設計(試験) |
| (2) | 機能の追加：前提条件の設定 | ラズパイを使った機能設計、共通 |
| (3) | 期待結果の確認方法：ログ | ファイル操作の機能設計 |
| (4) | 期待結果の確認方法：画像判定 | デジカメ画像の液晶パネルの画像判定の機能設計 |
| (5) | 運用に必要な設定内容：機種別の判定パラメータ | iniファイルのパラメータ設計 |
| (6) | 運用に必要な設定内容：エラー発生時の原因特定 | エラーログの設計、1つの自動化ツールでの一括制御 |

【現場の失敗例】
 自動化できる機能だけ
 自動化する

実施のプロセスではこれまでのプロセスで決定した自動化方針や詳細内容を元にスクリプトを作成する。

- ・スクリプトを作成する
- ・スクリプトを実行して確認する

1本1本のスクリプトでは正常に動いても、纏めて実行するとNGとなる場合がたびたびあります。

【現場の失敗例】
機材の連携でトラブル続発

【現場の失敗例】
期待結果の精度の低さ

導入前に定めた成功基準の考察

| No | 成功基準 | 結果 | 結果詳細 |
|----|---------------------------|----|------------------|
| ① | 手動に比べ30%以上の効率化/工数削減ができること | ◎ | 40%～80%の効率化が可能 |
| ② | 必要なテストが自動化できていること | ○ | 最大70%の機能は自動化可能 |
| ③ | 実行途中で処理が止まらないこと | △ | 設定ミスで実行エラーが頻発 |
| ④ | 実行結果の誤判定がないこと | △ | 設定ミスで実行エラーが頻 |
| ⑤ | 実行から結果確認まで自動化できること | ◎ | 一連の動作は全て自動化できている |
| ⑥ | メンテナンスを考慮した設計ができていること | ◎ | 自動化に必要な対策は完了 |

【現場の失敗例】
 振り返りをせず
 トラブルを抱えながら運用を続ける

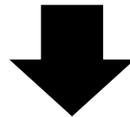
定めた成功基準と実績との比較を行い、
 基準をクリアための問題を特定する

条件①1つの自動化ツールで必要な機能が揃っていること

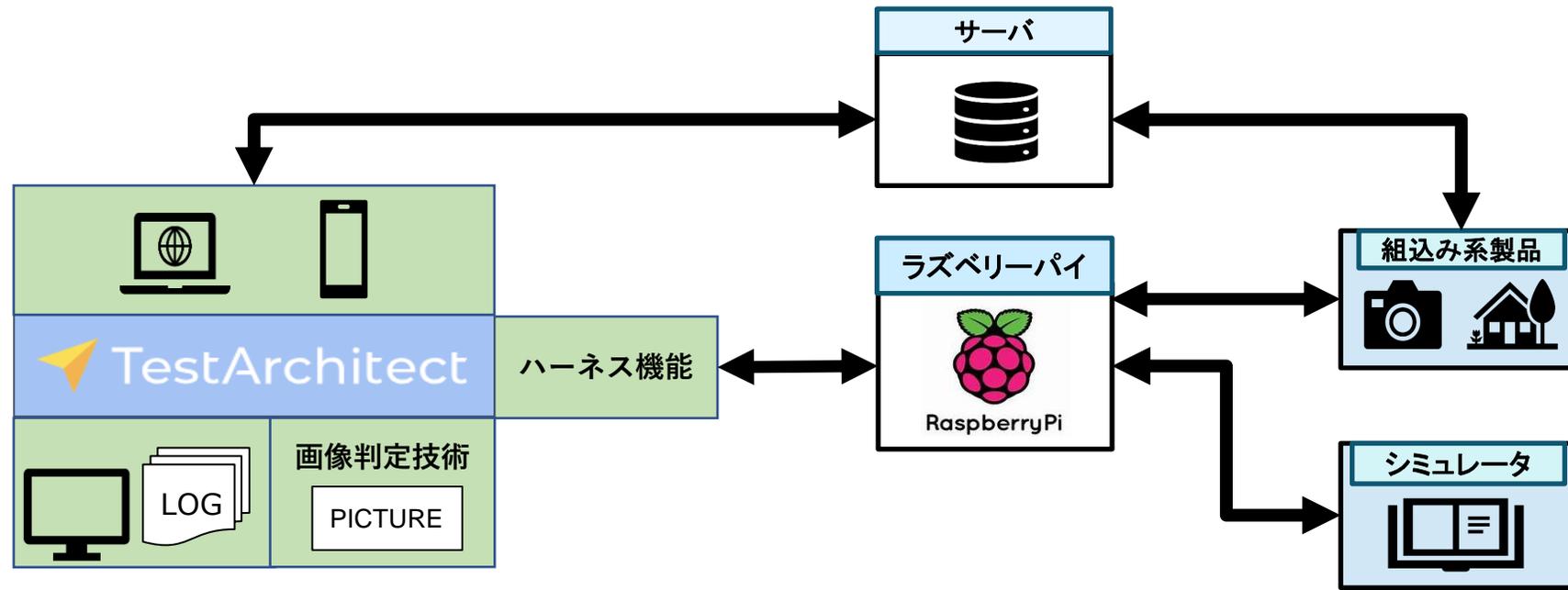
- ・組込み製品やシミュレータ機器を制御できること
- ・Webブラウザ、モバイルなどサーバとの連携ができる機能があること
- ・テスト項目の前提条件設定、手順実施、期待結果確認が自動化ができること

条件②運用を考えた自動化環境であること

- ・実行担当者にも理解しやすいシンプルな構造であること
- ・自動テストを1つのツールで一括制御できること



AGESTの製品であるTestArchitectを使うことで、上記の条件を満たせることが確認できました。

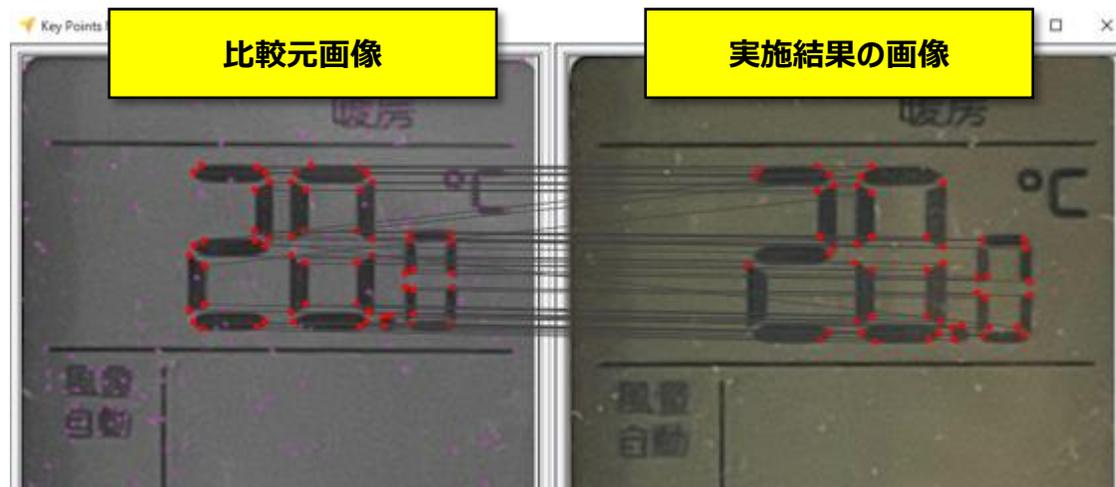


| TestArchitectの機能 | 特徴 |
|------------------|--------------------------------------|
| ハーネス機能 | ラズパイで制御する機能を取り込み、TestArchitectで使用できる |
| 豊富な自動化機能 | モバイルやブラウザでサーバと連携が可能、Windowsアプリなども自動化 |
| 豊富な画像判定技術 | 期待結果の画像の正確な判定が可能 |
| シンプルな自動化システム | 1つのツールで一括管理できるため、設定ミスの減るスマートな構造 |

ピクセル単位的一致ではNGになる画像も、KeyPointを使えば特異点の一致確認をするため画像比較が可能。

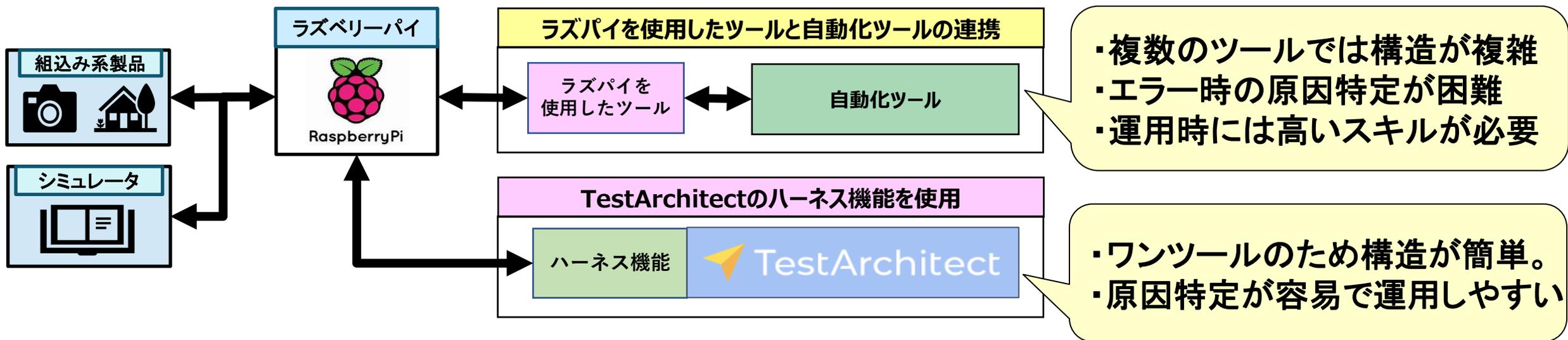
| 確認方法 | 判定方法 | 精度 |
|----------|-------------|----|
| 画像判定 | ピクセル単位的一致 | 低い |
| KeyPoint | 抜き出した特異点の一致 | 高い |

デジカメで取得した画像は位置ずれ、光の加減などより一致は難しい



KeyPointを使うと特異点の一致で判定を行うため確認精度が高い

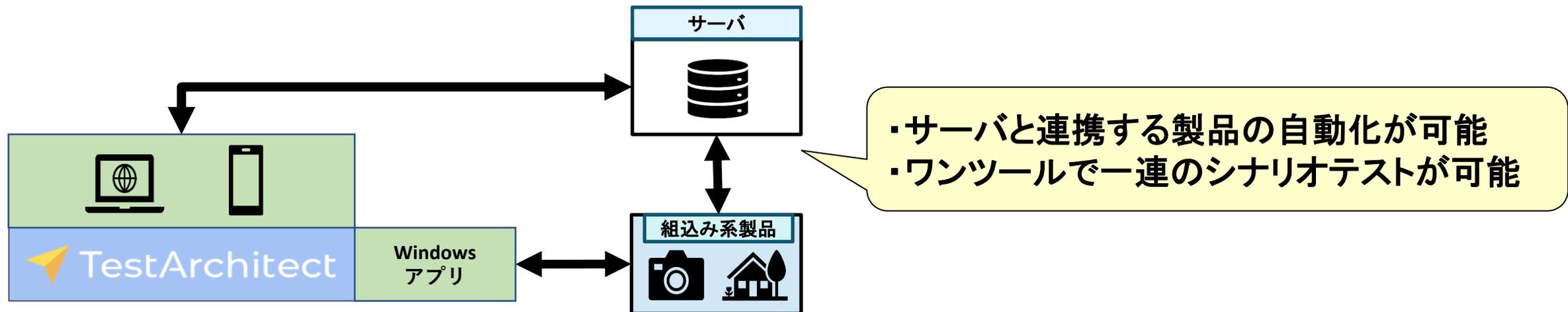
ハーネス機能を使えば自動化機能を作成することができます。
ラズパイを使用した機能を作成することで製品の制御も可能になります。



ハーネス機能で製品制御する機能を作れば、
長期間の自動テストの運用が可能になります。

組込み系では製品の操作だけでなく、サーバとの連携も必要になります。

TestArchitectではWindowsアプリだけでなく、Webブラウザ、モバイルでの制御も自動化することができます。そのためサーバとの連携が必要な組込み系製品も自動化することができます。



組込み系製品の自動化テストの注意点は以下になります。

| No | 組込み系の自動テストの注意点 |
|----|--|
| ① | 運用を考えた自動テストの設計を行うこと |
| ② | 1つのツールで自動テストを一括管理し、構造をシンプルにすること |
| ③ | 前提条件設定、実施手順、期待結果の確認などテストに必要なものは自動化すること |
| ④ | コスト削減だけに着目せず必要なコストは掛けること |
| ⑤ | 自動テストの知識や経験を持った技術者をツール開発に参加させること |

組込み系製品の自動テスト成立には自動テストの知識と運用を考えた自動化設計とそれを実現する自動化ツールの選定が重要です。