2025/3/27 JaSST' 25 Tokyo

組込みソフトウェアテスト自動化ツールの開発者が考えるテスト自動化の課題と対策

HEARTLAND.DATA

はじめに

アジェンダ

- 自己紹介と昨年のサマリー
- テスト自動化SmallStart事例
- SmallStartの課題
- 改めて考える自動化のポイント
- 今後のアプローチ

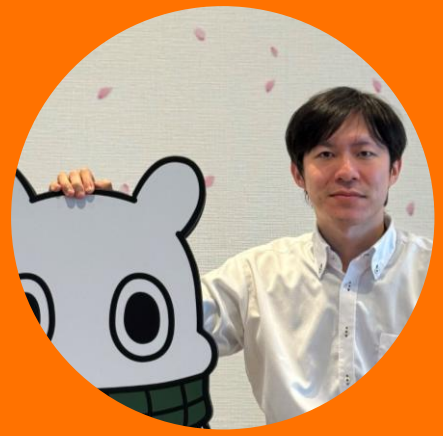
2020年12月
ソフトウェアデザインセンター
を新たに開設しました。

ワークスタイルの多様化に合わせ、
高集中、対話、情報整理、知識共有、
アイデア出し、リチャージなど
用途に合わせて様々な活用が可能な
リラックスできる
コミュニケーションスペースです。

社名	ハートランド・データ株式会社
代表取締役社長	落合 亮
設立	1982年1月18日
資本金	3,200万円
従業員数	70名（2024年7月期）
事業内容	開発サポートツール開発/販売、SW/HW受託開発、テスト自動化関連サービス、機能安全関連サービス、テクニカルソーシング等
国内拠点	栃木県足利市
海外パートナー	ドイツ　：Method Park by UL Solutions アメリカ　：Beyond Security 韓国　：MDS Tech、MOASOFT 中国　：Trinity Technologies イスラエル：Cybellum Technologies インド　：WINIX Technologies

自己紹介

塚原 智也



テストソリューションズ課
主任

約7年間、テストの効率改善・
テストツール開発に従事。
自動化ツール開発Pjリーダー。

趣味：謎解き（リアル脱出）

山﨑 陽平

テストソリューションズ課
技師

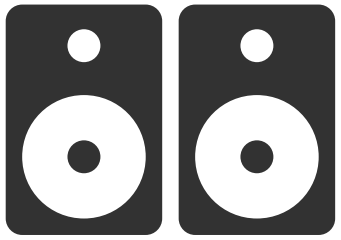
組込み向けテスト自動化ツールの開発、
検証に従事。主にスクリプト変換やシリア
ル通信周りの開発を担当。

趣味：デスク周り、ゲーム（FPS）

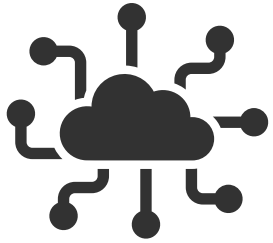
昨年講演させていただいた話

HLDC自動化の取り組み

受託開発プロジェクトでのテスト自動化



ネットワーク
オーディオ機器開発

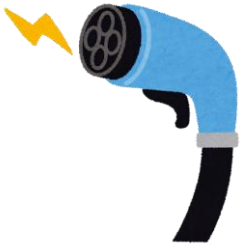


IoT機器開発

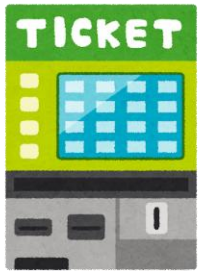
受託開発にて自動テストを実施し、
テストを効率化

テスト自動化環境の受託開発

家電製品の
テスト自動化環境
開発



充電プラグの
テスト手順の
半自動化



チケット券売機の
不具合再現の
効率化

開発製品や実施しているテストに合わせて
テスト自動化環境を開発

自動化環境構築での苦労話

これまで自動化で失敗したこと、苦労したことは...

検討と準備に時間がかかる

さらに...

想定とのギャップが存在

検討に時間を掛けるほど

やりたいことが発散していく

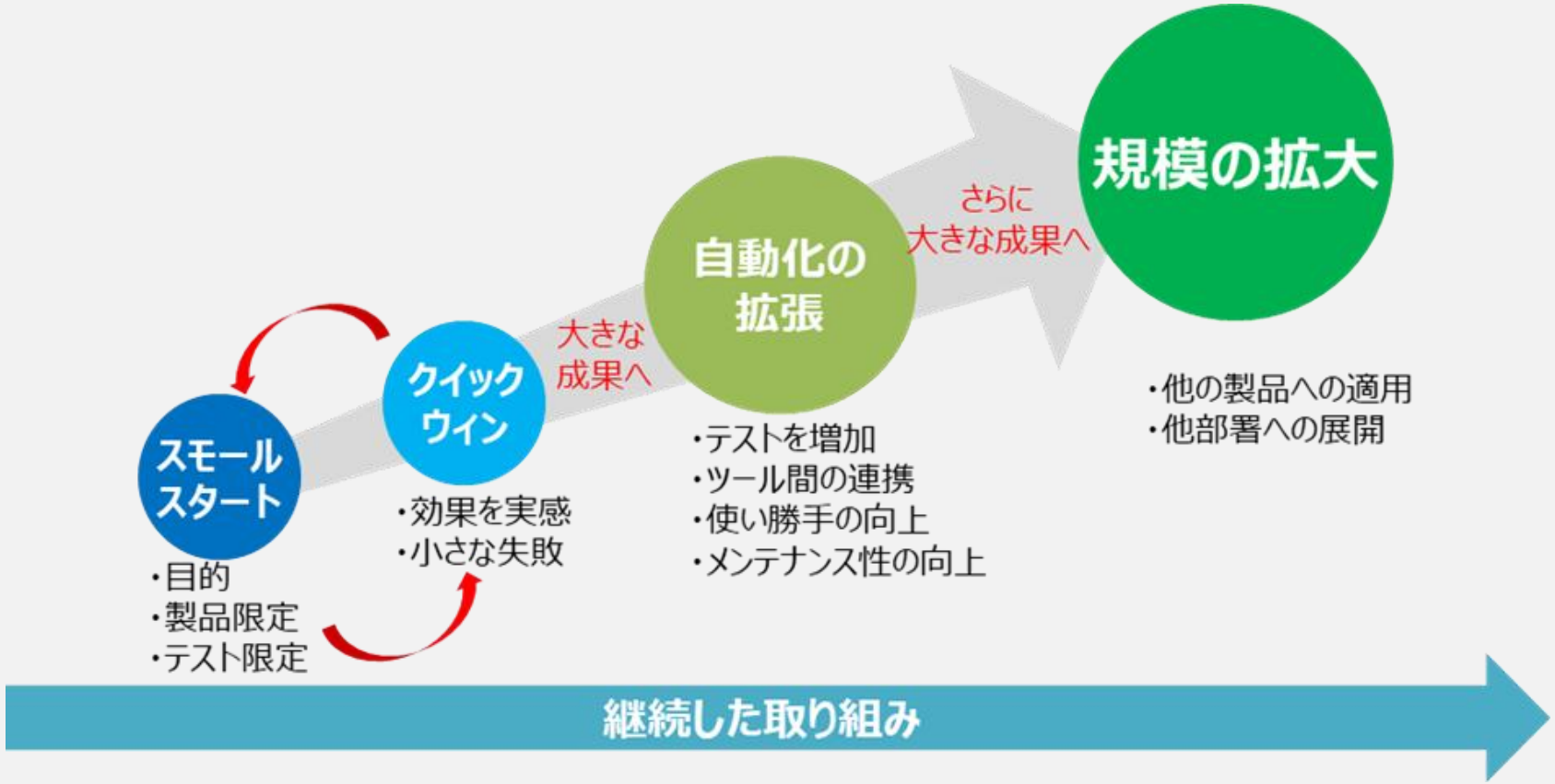
組込みテスト自動化の課題

自動化自体の課題	自動化の方法、費用対効果など検討事項が多い
	長期運用しないと費用対効果を十分に得られない
テスト運用自体の課題	テスト自動化環境自体の品質
	自動化環境にも定期的にメンテナンスが必要
組込み特有の課題	HW操作/物理的な入力が必要
	SWのテスト環境と比べテスト環境の修正・変更が加えづらい
	ターゲットのI/Fに合わせこむため他ターゲットに流用しづらい

- 費用対効果や長期運用前提を考慮して一度にシステム全体のテスト自動化を目指しがち
- 仕様追加・変更に柔軟に対応できず機能を絞った不十分なテスト環境になりがち

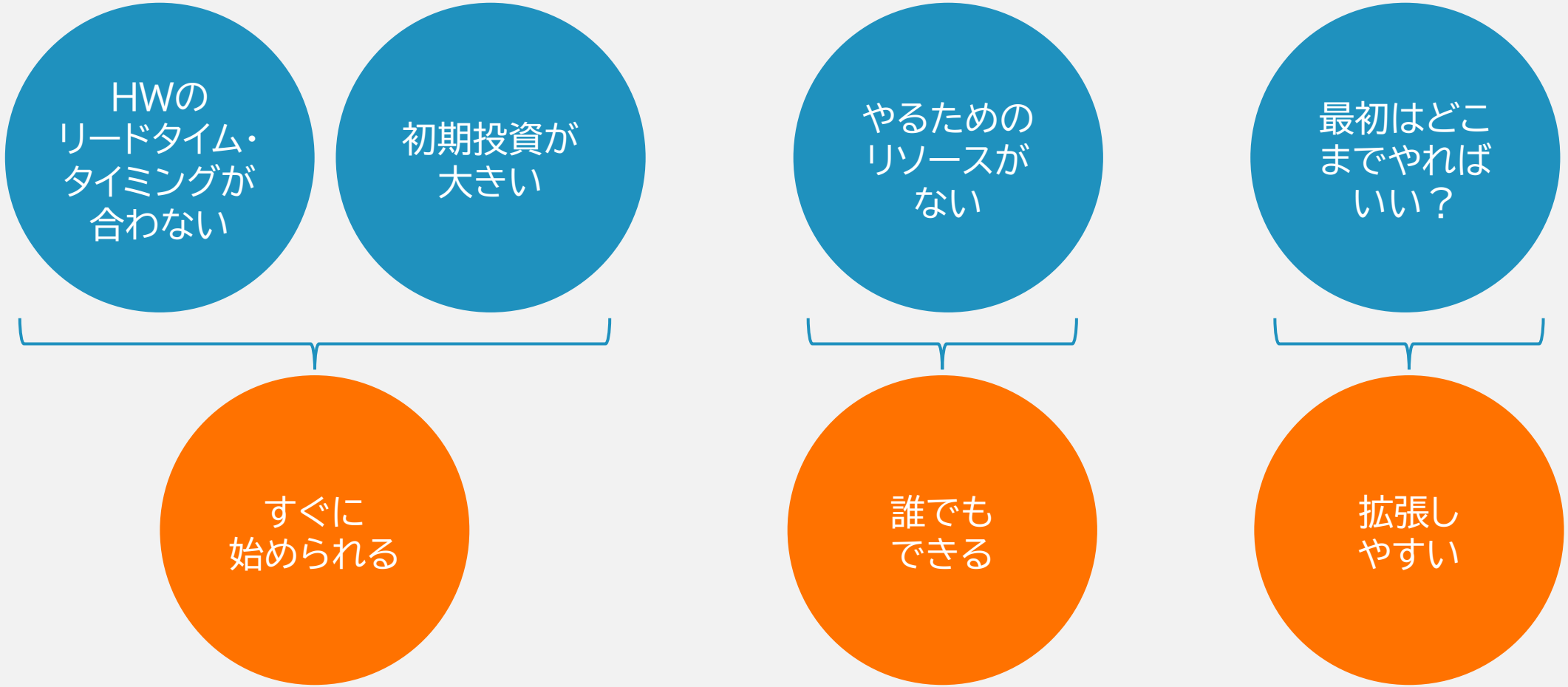
課題に対するHLDCのアプローチ

- まずはやってみるべき



SmallStartするには

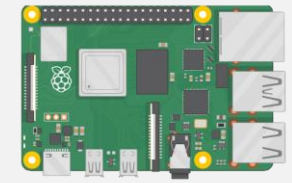
- 組込み開発で“自動テスト”を “SmallStart” するために求められる3つの要素



SmallStartするには

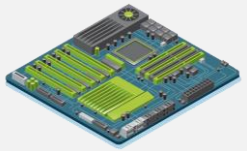
デバッグ・テスト時の

組込み機器の
煩わしい操作を伴う **「制御」「計測」を自動化します**



Raspberry Pi

×

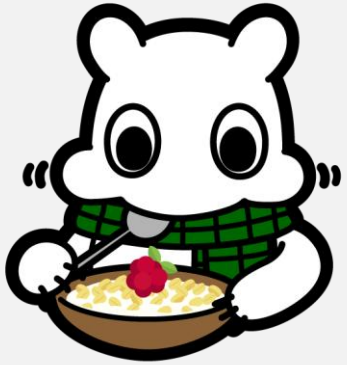


拡張I/Fボード

＋

テスト自動化
アプリケーション

＝



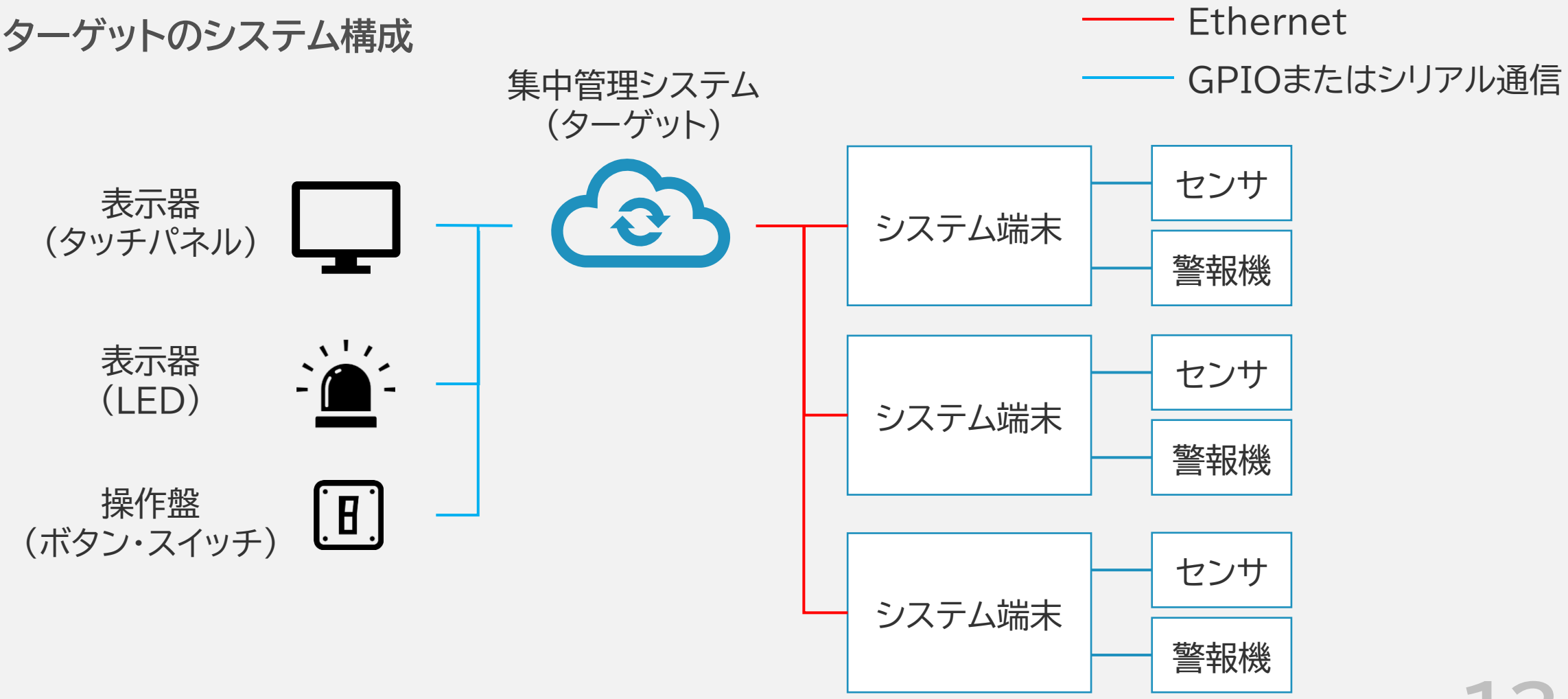
AUTOmeal

組込みSWテスト自動化事例

（10分）

最新事例①（1/13）

- 警報器集中管理システムの自動化
- ターゲットのシステム構成



最新事例①（2/13）

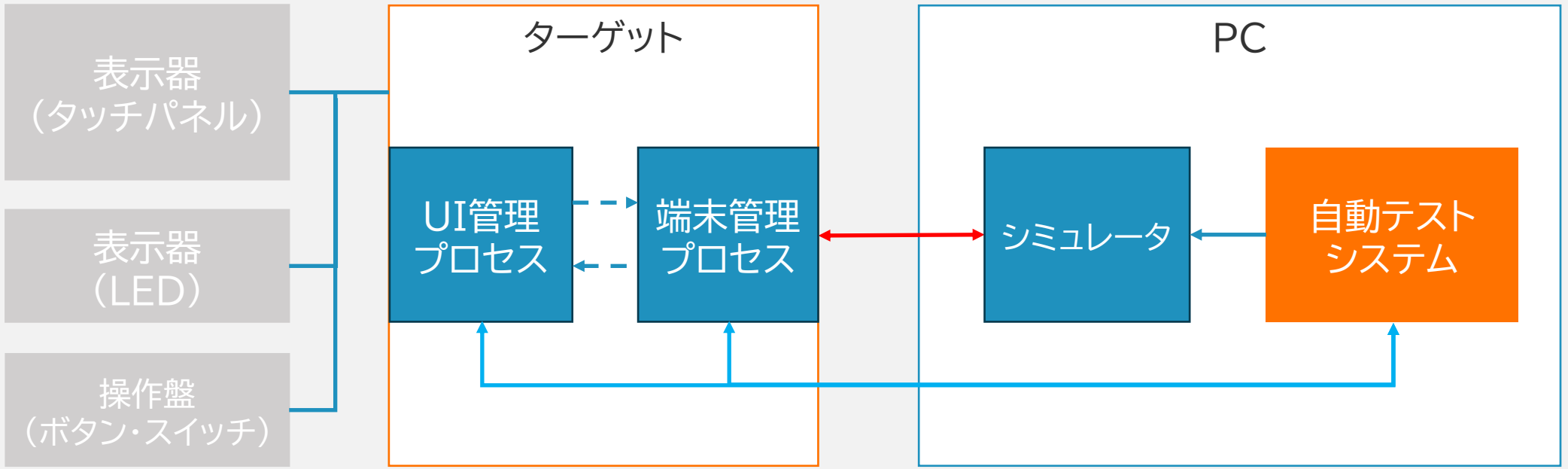
- スモールスタートの目的

テスト自動化の効果や
今後必要になることを明確にしたい

わずかでも自動化して実施負荷を減らしたい

最新事例①（3/13）

- **自動テストの環境**
  - プロセス間の通信(シリアル通信)を代替
    - » デバッグ用のコマンドを使用し、プロセス間の通信や表示器等の操作を再現
  - シミュレータ＋シリアル通信の代替でできるテストのみを対象
    - » デバッグ用途で使用していたシミュレータをGUIの操作自動化と組み合わせることで自動操作し通信を再現



最新事例①（4/13）

- スモールスタートしたポイント

自動化対象I/Fを通信に限定

実現方法が最も明確かつ
テストの主軸になる通信代替に限定

デバッグ用のコマンドを新たに仕込むなど
ターゲットのテスタビリティ向上との
合わせ技でテスト範囲を広げる

既存のテスト環境を活用

シミュレータを利用することで、
テストシステムにターゲットの
複雑な通信仕様を組み込まずに構築

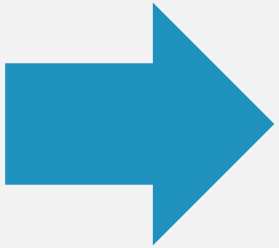
システム自体の開発工数を削減
テストシナリオも軽量化

最新事例①（5/13）

- スモールスタートによるメリット

フィードバックの早さ

およそ2ヵ月で
数件のテストケースを自動化



課題が明確になり
より効果の高いタスクを
判断しやすくなった

- 具体的には

自動テストのインプットとなる
テスト仕様書の不足情報が明確化
テスト仕様書の改善のきっかけにも

スクリプト作成の負荷軽減を早期に対策を開始
自動化システム開発に並行して
独自仕様ScpritGeneratorの仕様検討を進めた

最新事例①（6/13）

- スモールスタートの取り組みその2

ScriptGeneratorの開発

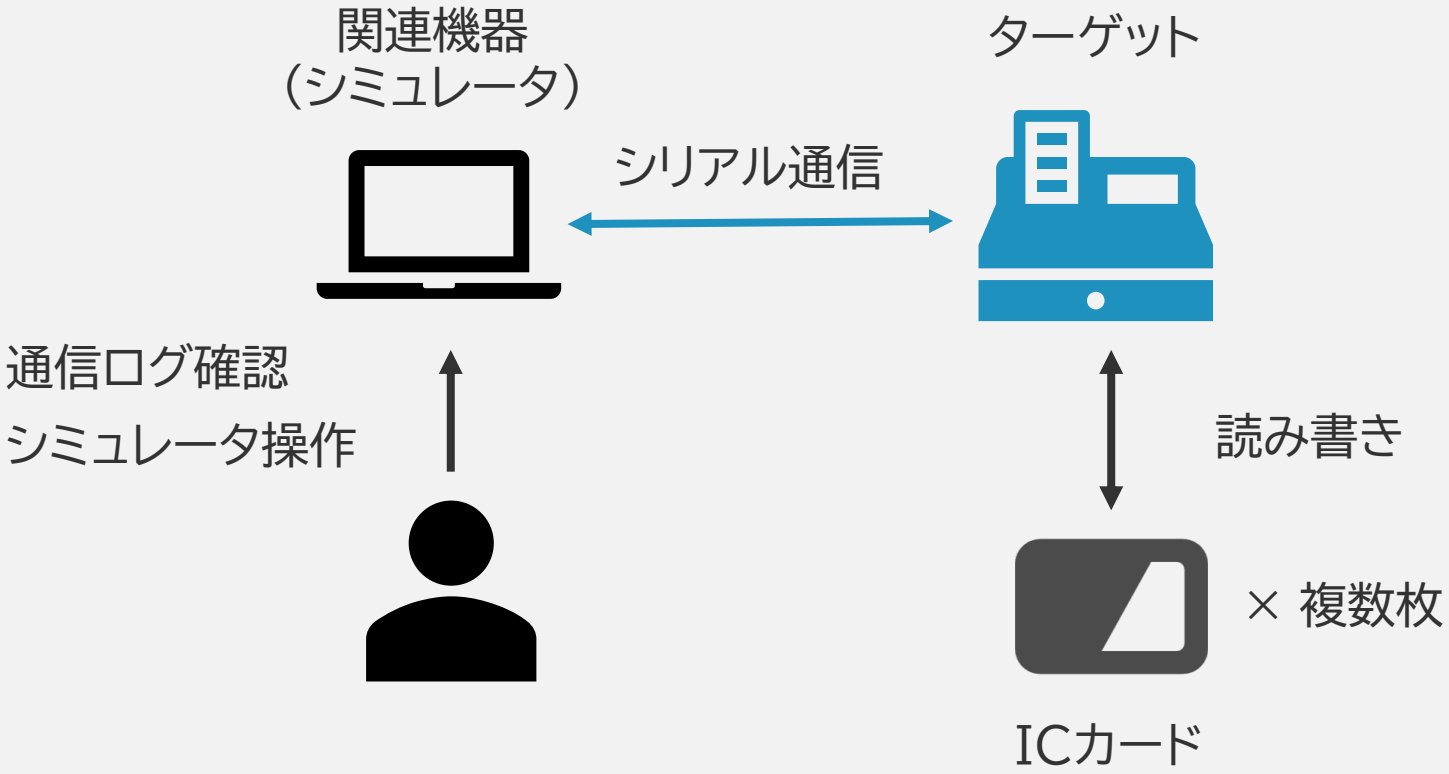
Excelファイルでシナリオを定義し、
ExcelファイルからPythonスクリプトへ自動変換するツールを開発

テスト仕様書からスクリプトを書き起こす手順がスムーズになった
Excelで記述できるためテストしているか直感的にわかりやすくなった

出力後にPythonも編集できるため、
プログラミングできるメンバーはより柔軟なテストスクリプトも作成可能

最新事例②（7/13）

- ICカードの読み込み機能がある機器のテスト
- ターゲットのテスト構成



最新事例②（8/13）

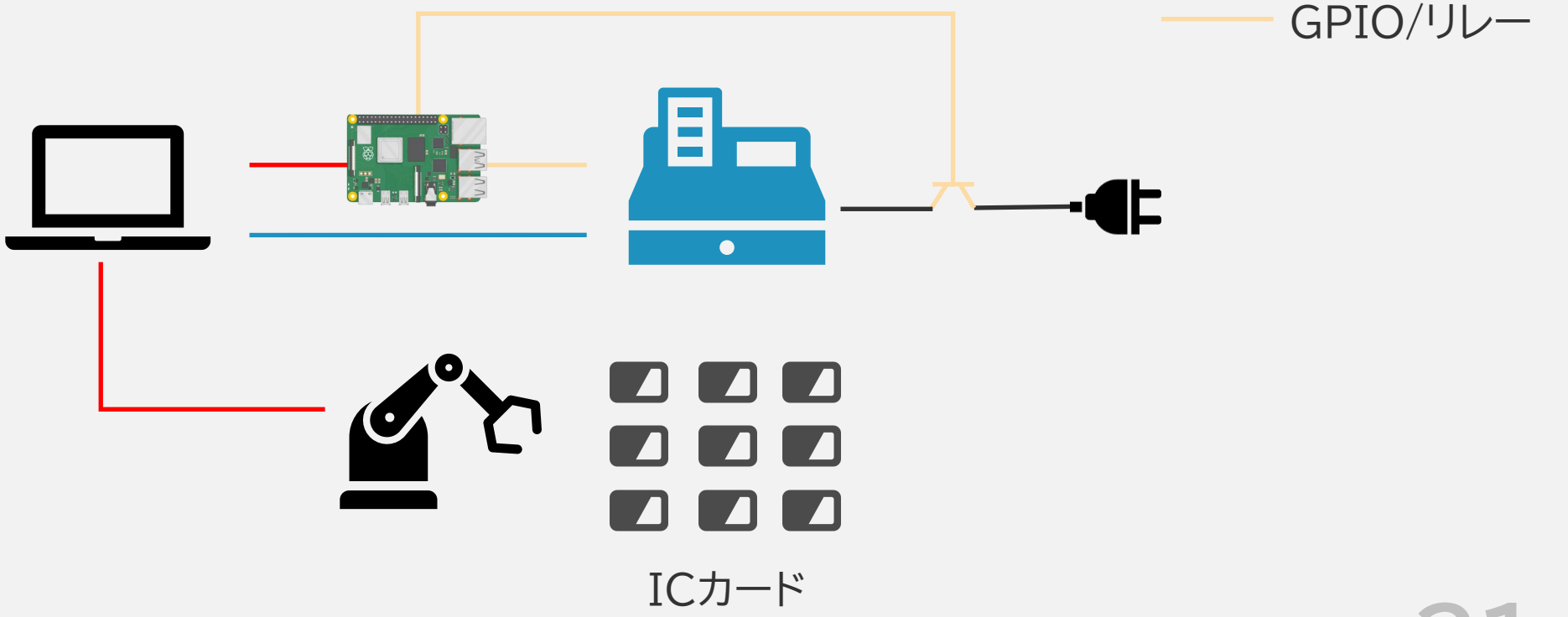
- スモールスタートの目的

今後の開発への自動化導入を考え、
既存製品で自動化で課題を洗い出したい

自動化できるイメージはあったが
自動化できたという実績を作りたい

最新事例②（9/13）

- **自動化構成**
  - ロボットアームでカードのタッチ・持ち替え
  - Raspiでターゲットのボタン押下を再現
  - Raspiで電源をリレー制御しOFF/ONを制御



最新事例②（11/13）

- スモールスタートしたポイント

最小機能の機種を対象

ほぼコア機能だけの機種

他機種と比べてテストケース数も少ないが
対応カードが多く繰返し行うテストが多い

「自動化できる内容」から範囲を決める

どのI/Fを自動化できれば
全体の何%を自動化できるか算出

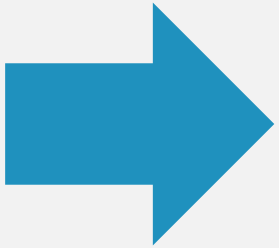
懸念や障害のあるテストケースは
自動化の対象にせず

最新事例②（12/13）

- スモールスタートによるメリット

予算・時間の確保がしやすかった

安価なロボットアームと
Raspberry Piで
最小構成で環境構築



期間や金額が莫大ではないので
比較的容易に計画・執行できた
短期間なので具体的な
スケジュールを組めた

- 具体的には

担当者が他の作業に邪魔されることなく
自動化業務に集中でき
レスポンスも早くコミュニケーションがとれた

期間が短いので必然的に
進捗管理や課題の共有もこまめになり
認識のズレが起こりにくかった

最新事例②（13/13）

- 自動化の効果

約 800 / 1000 件の自動化ができた
1.5人月（30人日）かかっていたテスト実行が
自動化できなかったところを手動でやっても10人日程度に

手動ではやむを得ず削減していたテストや
デバッガを使ったみなしOKとしていたテストを
テスターの負荷を上げることなく実施できる

1000回以上の
タッチ試験
を自動化

連続稼働や
長時間の待機確認
も可能に

組合せを
網羅的に実施

SmallStartの課題

（15分）

SmallStartをするときの課題（1/18）

どこまで拡張性・汎用性を持たせるべきか

小規模に始めても効果ないのでは？

最終的な効果を示さないといけない

SmallStartをするときの課題（2/18）

- どこまで拡張性・汎用性を持たせるべきか

- 組込みは検討要素が多い

ハードウェアの存在が大きく影響

物理的な操作

HW特性の差異

変更の容易性

SmallStartをするときの課題（3/18）

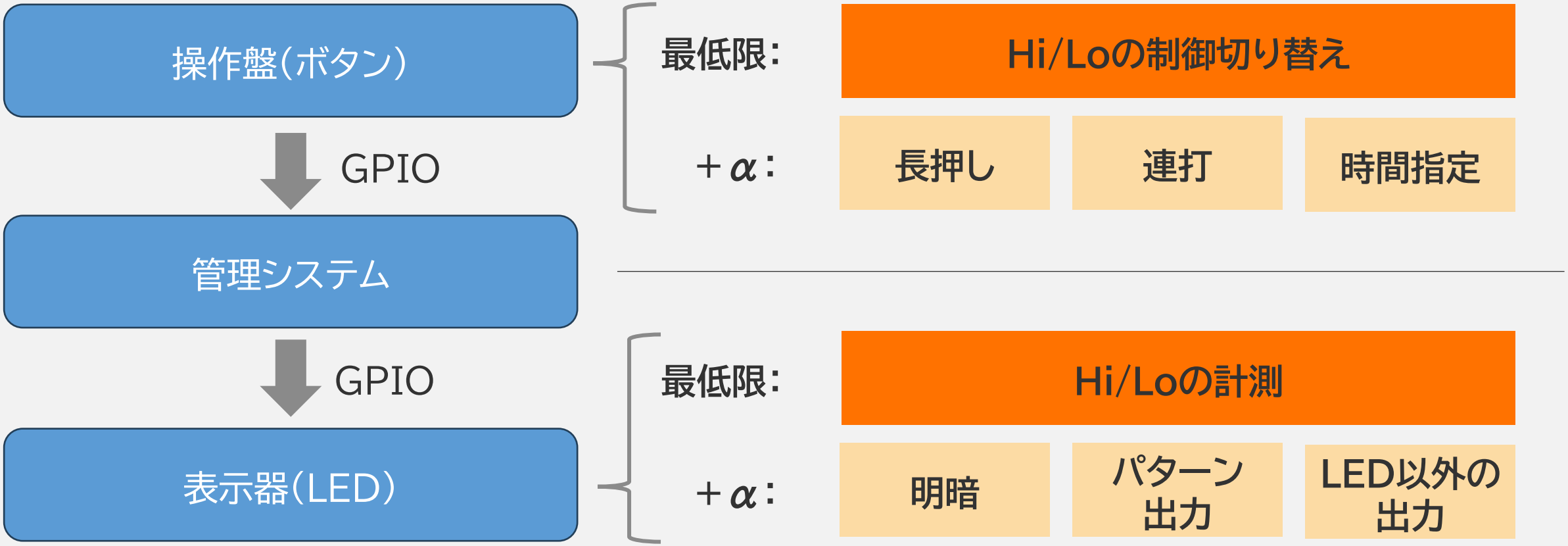
- テストの検討要素

SmallStartをするときの課題（4/18）

- 単純なI/Fに絞る

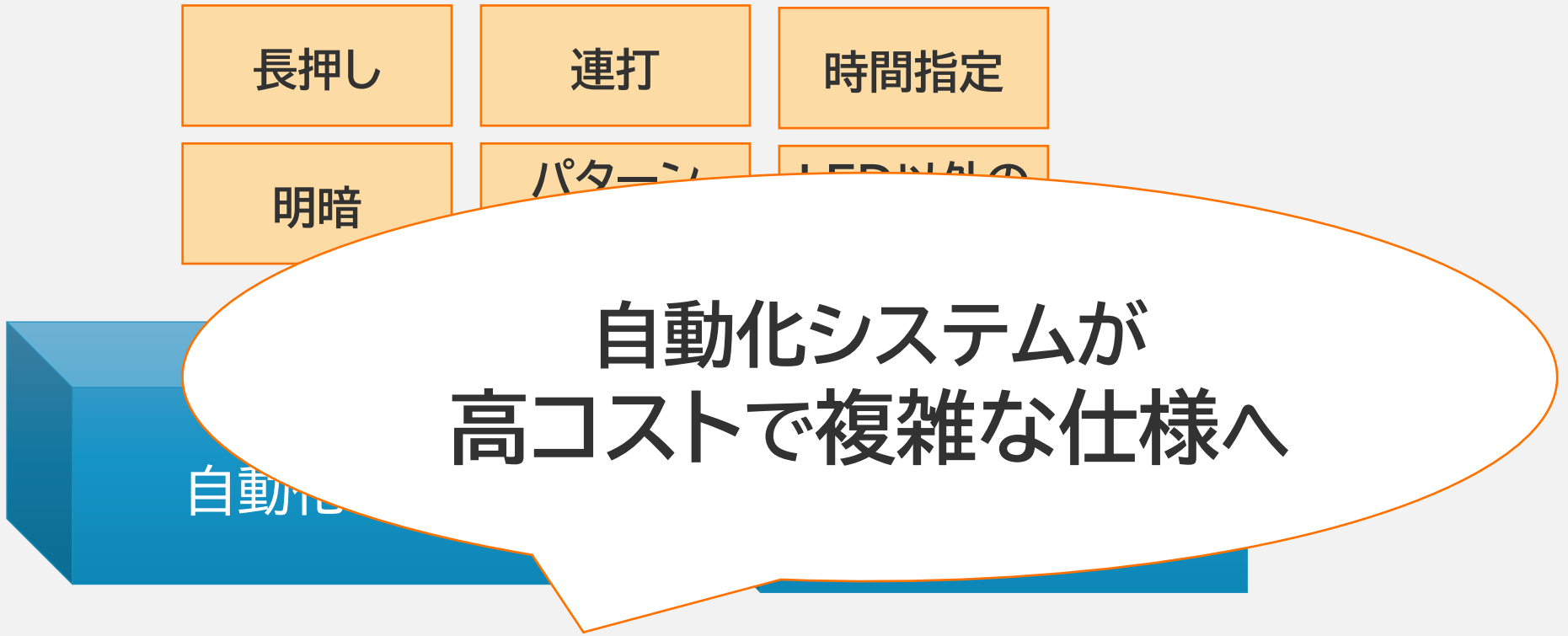
SmallStartをするときの課題（5/18）

- 拡張性を考えると



SmallStartをするときの課題（6/18）

- あれもこれも自動化しようとすると



なんでもできるツールが必要になってしまう

SmallStartをするときの課題（7/18）

- どこまで拡張性・汎用性を持たせるべきか

基本は目的を達成できる最小であるべき

共通する仕様や I/Fのみ自動化する	（エージングテスト など） 1つのプロジェクトでも 効果のある範囲に適用する	自動化しない 半自動化を目指す

✓ 何度もテストするテストフェーズ
✓ 手順の繰り返しが多いテスト　…など

SmallStartをするときの課題（8/18）

- 最小限の範囲に絞るリスク

次のステップで拡張できずに作り直しに

ツール導入なら選定の時点で拡張性・汎用性のあるツールを選べばそのリスクを軽減できる

- 自前で用意
  - ターゲットにあわせて用意するため、拡張性が乏しい
  - 拡張性を持たせようとすると、機能が大きくなり開発費が増える
- ツールを使用
  - 汎用性が備わっている
  - 他機種に流用できる

SmallStartをするときの課題（8/18）

■ 最小限の範囲に絞るリスク

次のステップで拡張できずに作り直しに

予算・リソースとリスクのバランスをとる

■ 自前で用意

- ターゲットに併せて作るため拡張性がない

■ ツールを使用

- 汎用性が備わっている
- 他機種に流用できる

SmallStartをするときの課題（9/18）

- 小規模に始めても効果ないのでは？

工数削減/期間短縮

- ✓ 人がテストをしなくてよくなる
- ✓ テスト密度を増やし手戻り工数を削減する
- ✓ テスト頻度を増やし手戻り工数を削減する

品質向上

- ✓ 人為的なミスを減らし検出漏れを減らす
- ✓ テスト密度を増やし検出数を増やす
- ✓ テスト頻度を増やしバグを早期発見する

SmallStartでは早期のフィードバックにより効果が見えやすい

SmallStartをするときの課題（10/18）

- ただし効果が薄い場合も

導入コスト ＞ 自動テスト1回の利益

すべてのテストで即時的な効果が出るわけではない

大規模に始める前の課題抽出やリスク回避が目的

SmallStartをするときの課題（10/18）

- 効果が薄い場合も

導入コスト ＞ 自動テスト1回の利益

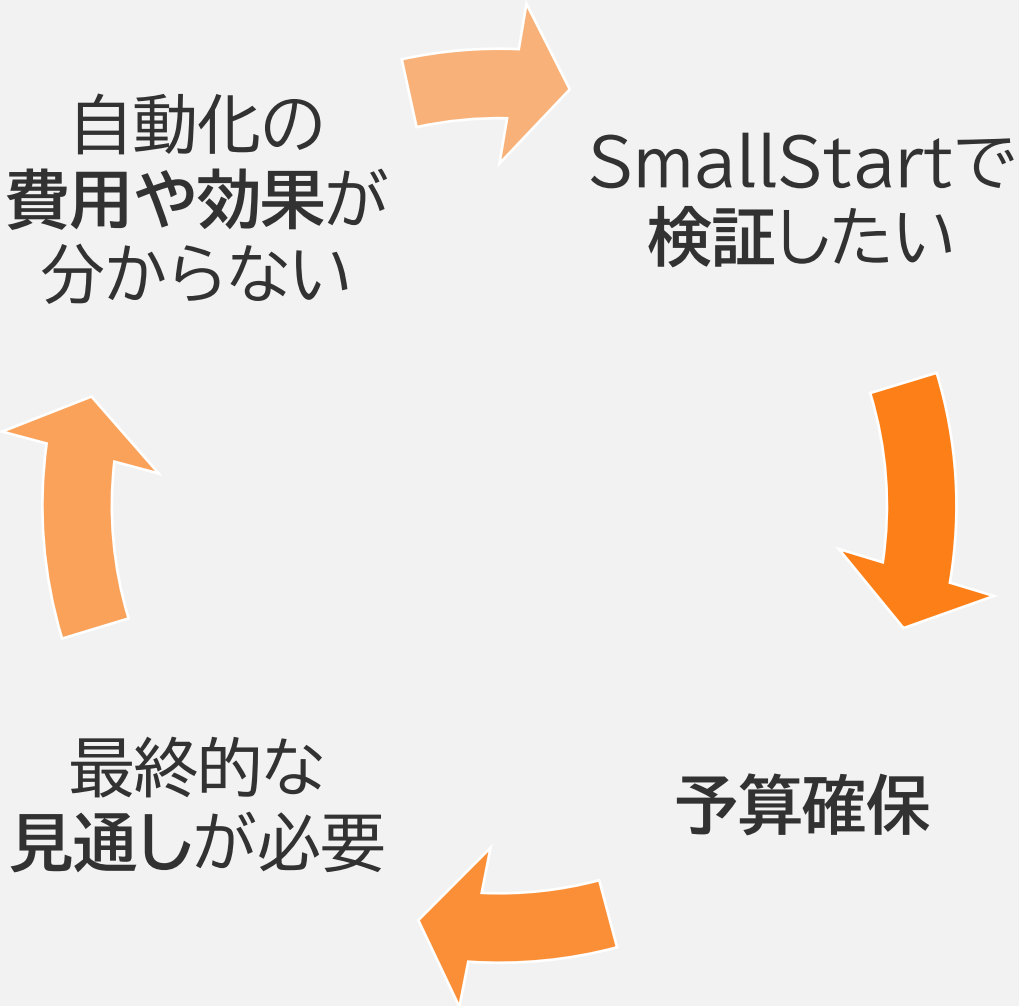
SmallStartでも適切な自動化対象に絞ることで
大きな効果を得られる

何を目的として取り組むのか運用次第で
効果が変わる

大規模に始める前の課題抽出やリスク回避が目的

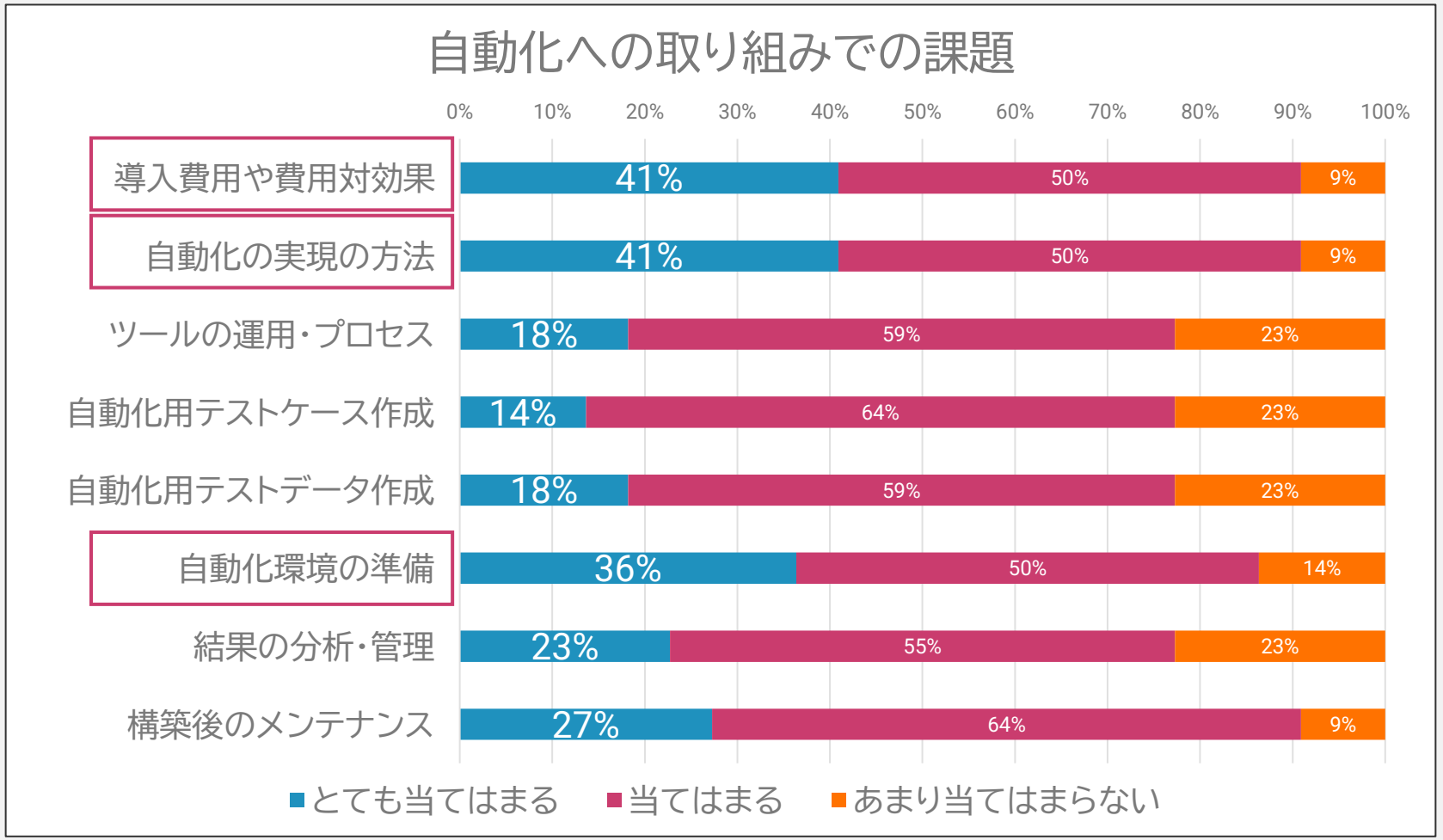
SmallStartをするときの課題（11/18）

- 最終的な効果を示さないといけない



SmallStartをするときの課題（12/18）

- 弊社で実施したアンケート



SmallStartをするときの課題（13/18）

- ポイントを絞って「やってみる」

自動化の導入・運用コストが少ないテスト

- ✓ 手順が単純で、少ない操作
  ⇒ シンプルな仕様のシステムで可能
- ✓ 環境や外部要因の影響を受けにくい
- ✓ 過去に繰り返し実施されている
  ⇒ 修正・メンテの可能性が低い

自動化の効果が大きいテスト

- ✓ パターンが膨大で長時間拘束される
  ⇒ 削減効果が大きい
- ✓ テスターの感覚や判断に依存
  ⇒ 属人化抑制による品質向上が期待
- ✓ 製品の品質向上に直結する主要機能
  ⇒ 品質低下のリスクが大きい部分の保証

Smallな範囲でも効果が出る

SmallStartをするときの課題（14/18）

- 自動化の拡張

成果が見えた部分から、着実に範囲を拡大

SmallStartをするときの課題（15/18）

- 全く目算もできない状態なら

さらに小さく始める

PoC的に行う

✓ 用意できるリソースで、小さく始められる範囲でやる

効果検証のためのシステムとして構築する

✓ プロトタイプ開発と同じくフィードバックを得て
本番環境の構築に生かす

SmallStartをした後の課題　（16/18）

区切りで放置されやすい

SmallStartをした後の課題　（17/18）

- 区切りで放置されやすい

次のStepに進まない

- 要因としては

効果が把握できていない

ルールがなく
運用できていない

自動化の需要がなくなった

担当者が満足した

SmallStartをした後の課題　（18/18）

Smallであっても次のステップまでの計画を行う

- PDCAを回す、プロジェクトとして運用する

次のステップの計画までを
1つのフェーズとして考える

ゴールとステップを設定

ステップ達成の計画と
モニタリング

運用リソースの確保

改めて考えるテスト自動化のポイント

（10分）

改めて考えるテスト自動化のポイント （1/9）

- ポイント

完全な自動化は目指さない

改めて考えるテスト自動化のポイント （2/9）

- 全自動化しない理由

自動化することで
新しい作業が増える

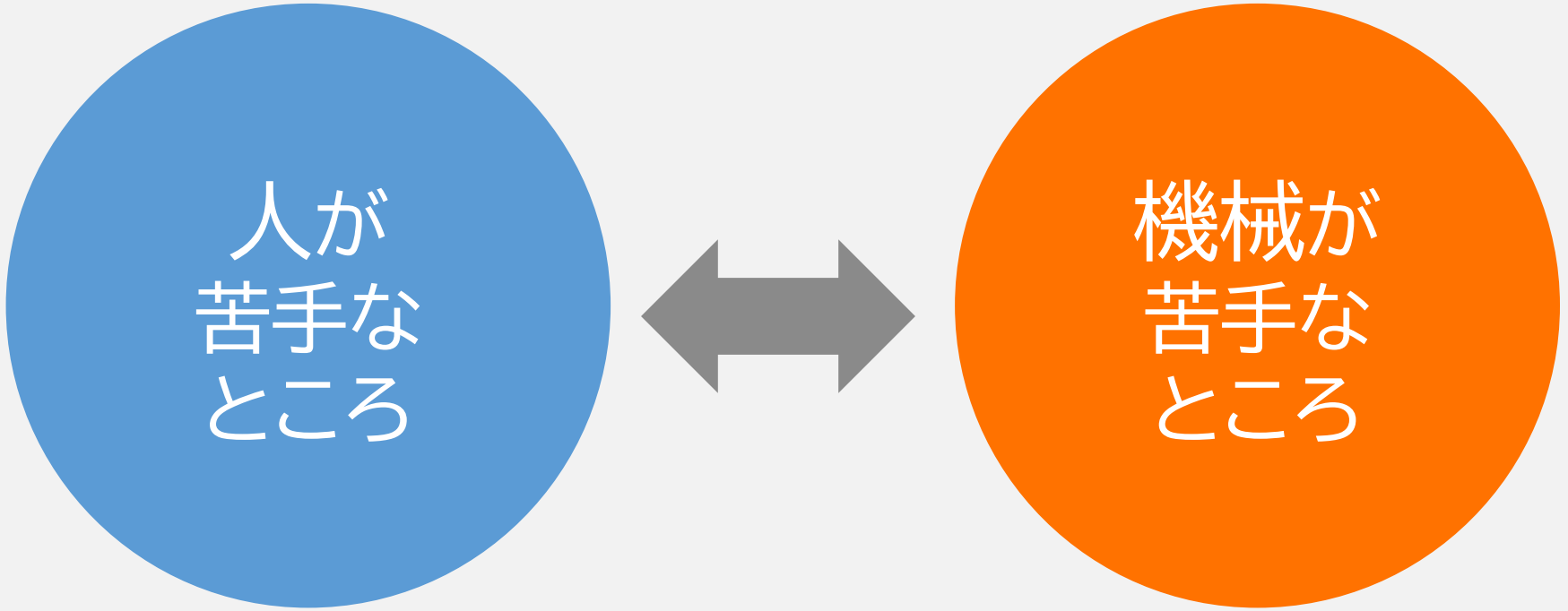
自動化しても
変わらない作業がある

自動テストのコスト ＞ 手動テストのコスト
となるテストは必ず存在する

⚠ 導入コスト、学習コスト、ランニングコスト、メンテナンスコスト…
⚠ 今までにない作業は慣れた作業よりも効率が悪くなる

改めて考えるテスト自動化のポイント （3/9）

- 全自動化しない理由



自動化しようとすると
システムが複雑化

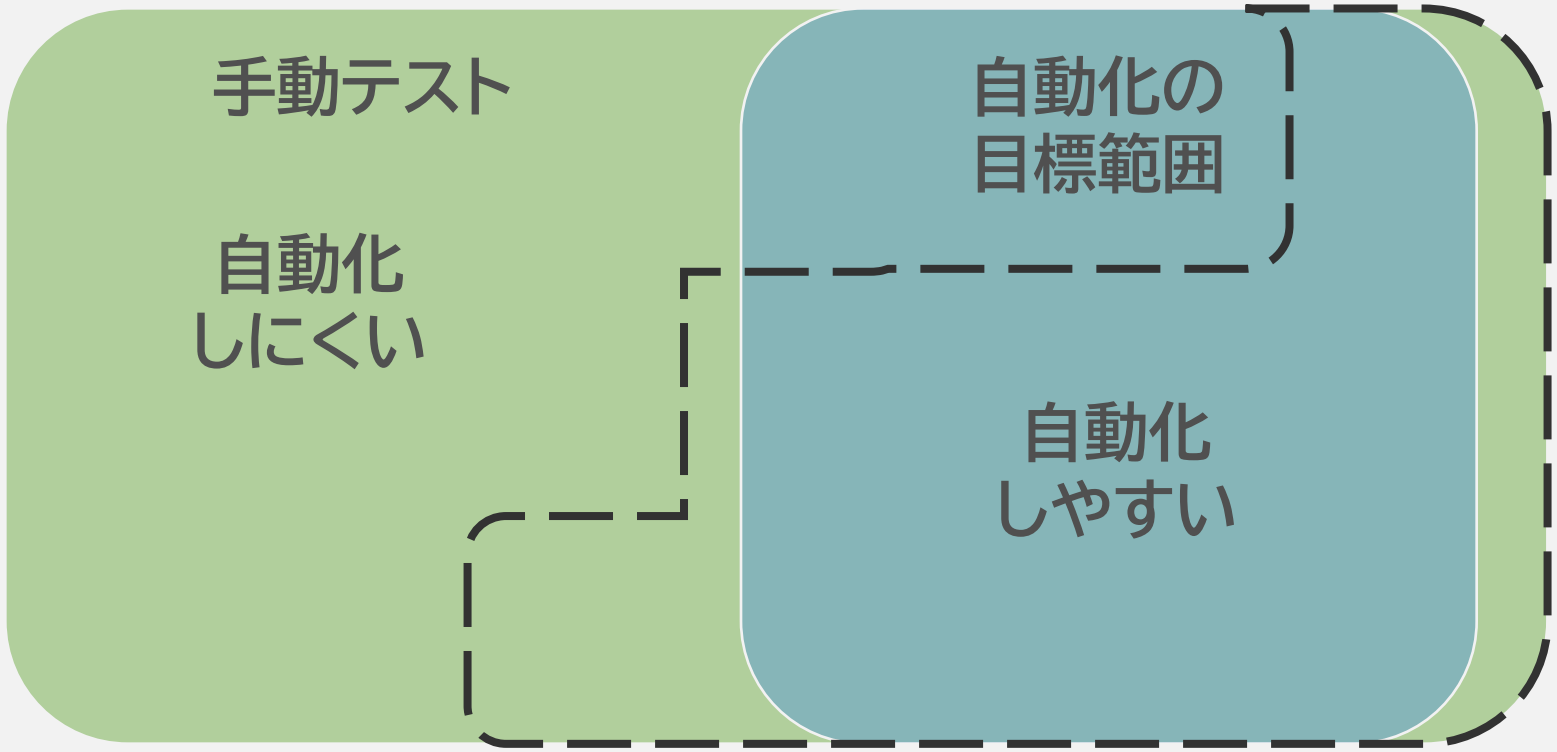
改めて考えるテスト自動化のポイント （4/9）

- 手動と自動は排他である必要はない

**フェーズごと/機能ごとなど
目標を設定することが多い**

改めて考えるテスト自動化のポイント （5/9）

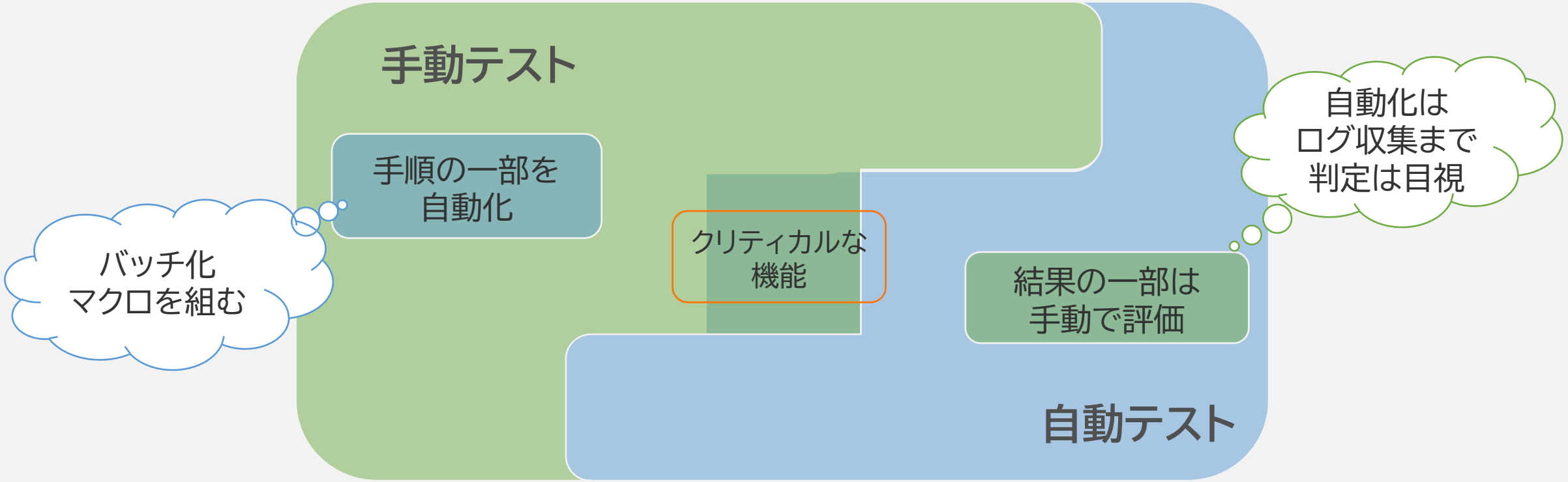
- 実際は自動化しやすい/しにくいテストは混交している



**最初に決めた範囲に固執すると
不適切な範囲で自動化することになる場合もある**

改めて考えるテスト自動化のポイント（6/9）

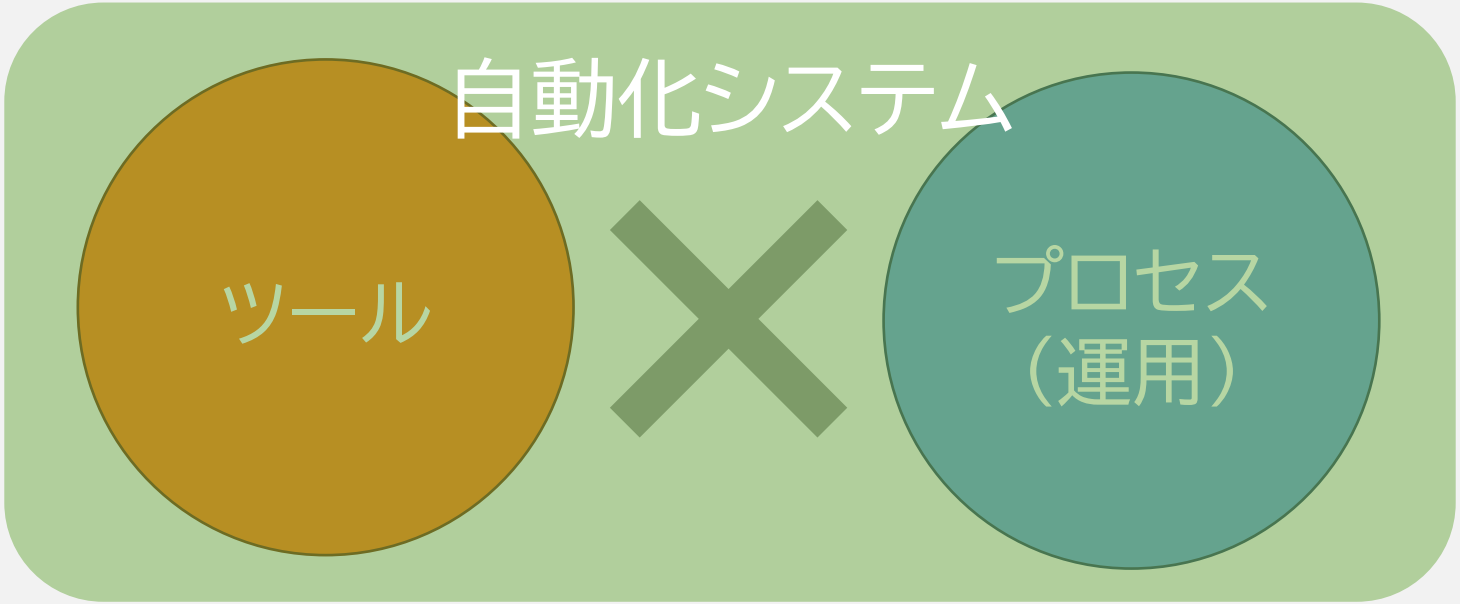
- 一部だけの自動化でも効果はある



**自動と手動が混在していることが
最適な場合もある**

改めて考えるテスト自動化のポイント （7/9）

- ツールの性能だけでは不十分



自動化によって新たに発生する作業の効率化も必要

改めて考えるテスト自動化のポイント （8/9）

- 結局は

コストと効果とリスクのバランス

改めて考えるテスト自動化のポイント （9/9）

- そうはいっても

コストと効果とリスクが不明瞭

あいまいだと気付かずに
スタートしてしまう

あいまいだとわかっているから
スタートできない

明確な範囲から始める
少しずつ明確にする

今後のアプローチ

（10分）

今後のアプローチ（1/8）

- ソフトウェア開発においてテスト自動化は必須

ソフトウェア開発は大規模化複雑化、テストも肥大化
人材不足は進行する一方

次々に新しい商品が出て
製品の開発サイクルは短くなっている

一方で商品のライフサイクルが長期化もしている
＝リグレッションテストの機会も増えている

今後のアプローチ（2/8）

- テスト自動化システムに求められていること

実施以外の自動化

多機能化

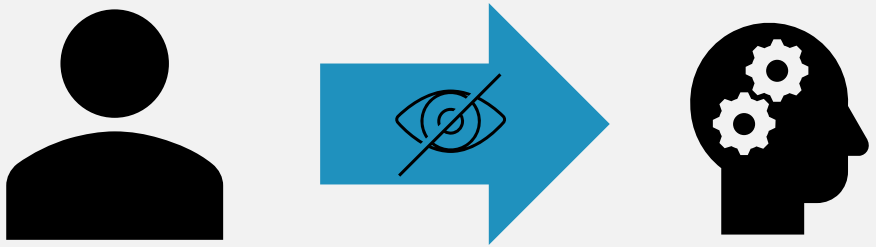
高性能化

大規模化

今後のアプローチ（3/8）

- テスト自動化システムに求められていること

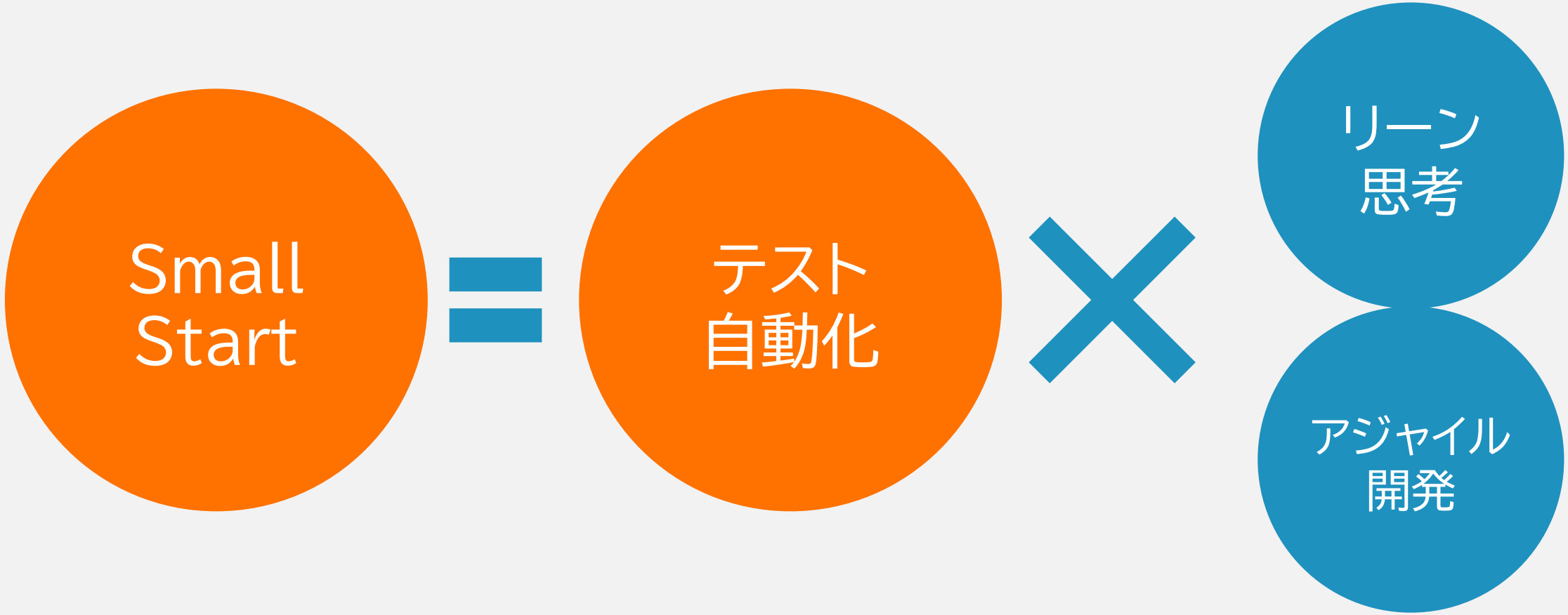
テストシステムの妥当性と透明性



自動化が進むほど、人の評価が入らずブラックボックス化していく
正しい入力がされたこと、正しいプロセスを実施したこと、正しい出力ができたこと
それぞれを担保する妥当性と最終的に人が確認できる透明性が求められる

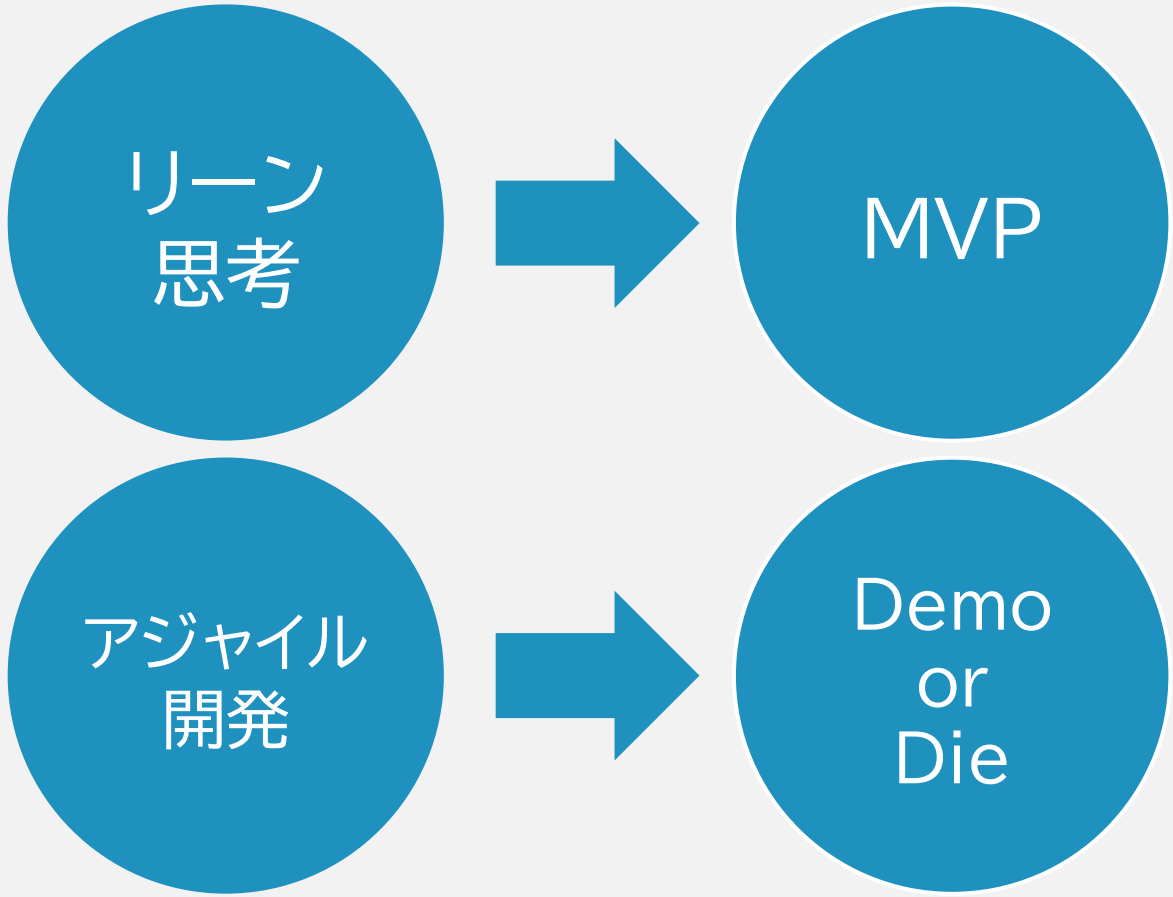
今後のアプローチ（4/8）

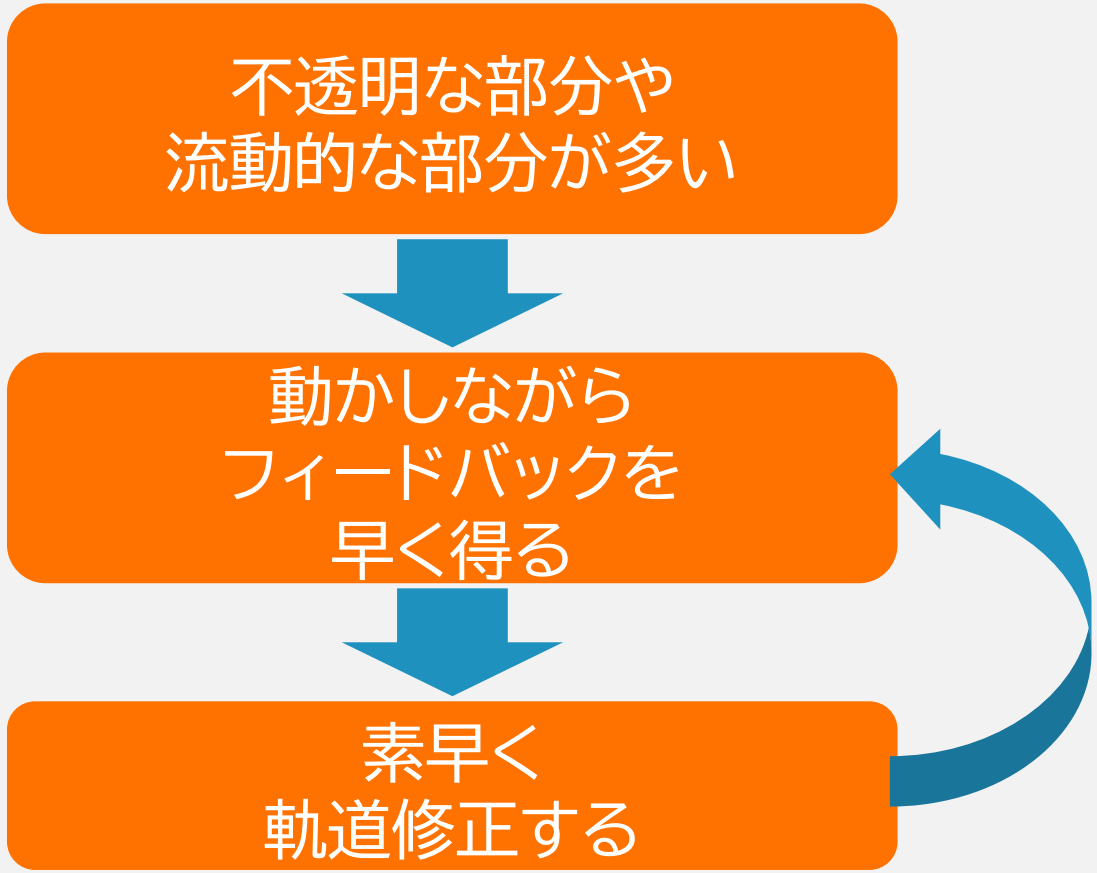
- HLDCのいうスモールスタート



今後のアプローチ（5/8）

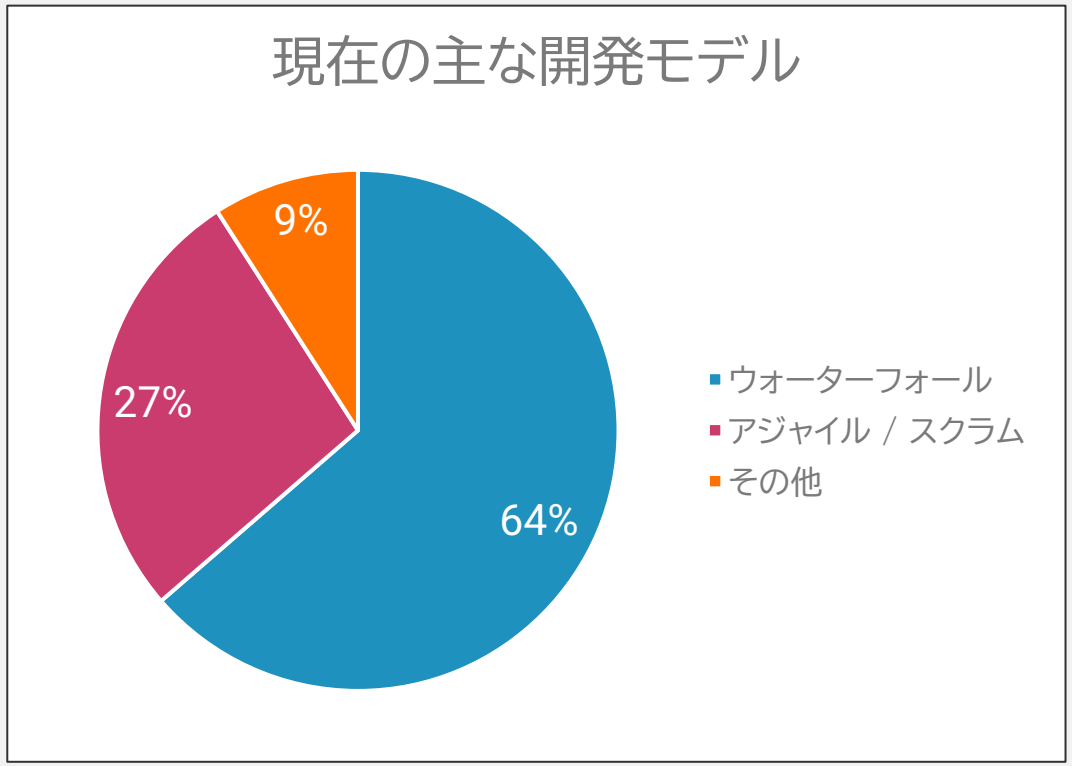
- リーンやアジャイルで言うところの





今後のアプローチ（6/8）

- 組込み開発においては



今後のアプローチ（6/8）

- 組込み開発においては

WFがまだ主流

アジャイルに転換しきれない

一度のテスト期間は長いが、
「繰り返しテストする」「長期間運用する」
というイメージが持てない

テスト対象が動くまで検証できず
製品開発に合わせ、
テスト環境もWFで開発せざるを得ない

リソース・スケジュールの優先は
開発 >> テスト

テスト自動化の具体的なイメージが
つかめないまま自動化導入に踏み切り
結果としてうまくいかない・効果が出ない
というネガティブな結果・印象を
残して終わってしまうリスクが高い

今後のアプローチ（7/8）

- 組込みソフトウェアテストツールとして目指す方向性

自動化で大きな効果を出せるツール

今後のアプローチ（7/8）

- 組込みソフトウェアテストツールとして目指す方向性

自動化を定着させるためのツール

素早く
環境が作れる

誰でも
使える

ピンポイント
でも効果が
出せる

今後のアプローチ（8/8）

- AUTOmealの今後の展開

標準I/Fの拡張

- ロボットアームとの連携強化
- カメラ映像での画像判定
- マイクでの音声判定

楽な運用のための機能

- ノーコード/ローコードでのスクリプト作成
- テスト実行・テスト結果の管理
- スクリプトデバッグ・NGの解析補助

まとめ

まとめ

課題や効果が不明瞭ならまずは小さく

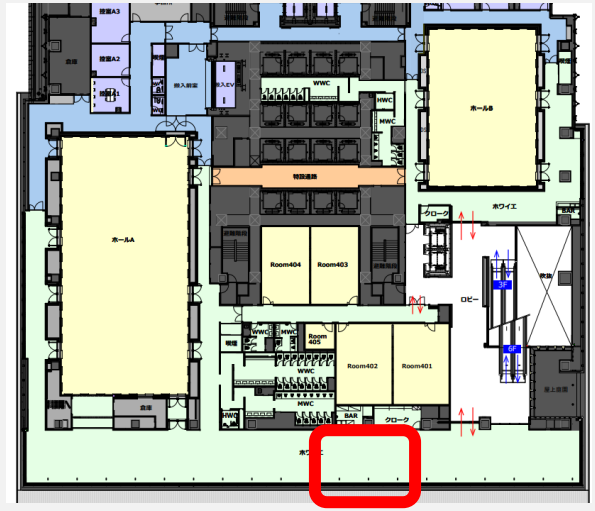
小さく始めたら継続が大事

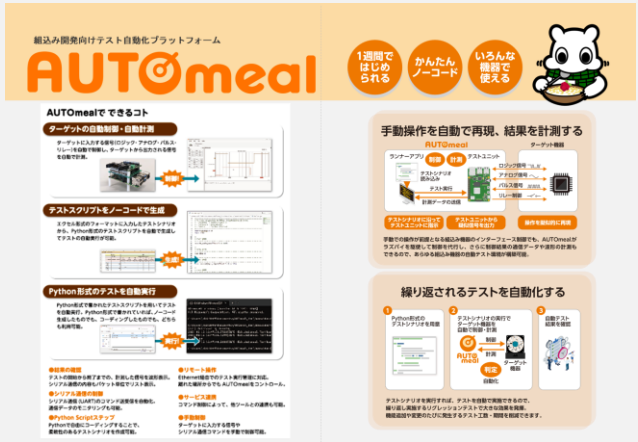
そのためのツール始めました

さいごに

ブース＆オンラインセッション

- AUTOmealの実機展示中





- E7）不具合流出しにくいソフトウェアテスト 組込み開発におけるテスト運用のポイント

○いつでもご視聴可能

E7）不具合流出しにくいソフトウェアテスト 組込み開発におけるテスト運用のポイント（登録制・いつでも視聴可）

新井 雅嗣（ハートランド・データ）

セッションの内容

不具合流出を防ぐためには、ソフトウェアテストは非常に重要なフェーズであることは疑いようがありません。そもそも流出が問題になるような致命的な不具合とは何なのか。それらを流出させないためには、どのような程度のテストを網羅させるべきなのか。そのテストを実現するにはどのようなテスト管理をすべきなのか。本セミナーでは、そんなテーマについて、テストエンジニア目線で掘り下げます。

Discordにて

#b2_組込みソフトウェアテスト自動化ツールの開発者が考えるテスト自動化の課題と対策
にてアンケートを展開しております。
　〇本講演のご意見、ご感想
　〇ハートランド・データに対するご質問
　〇自動化に関するお悩み
などなどぜひご協力お願い致します！

また、#ハートランド・データ でも情報発信しております。
お気軽にお問い合わせください。

Thank you!!

▲ ”Designed by macrovector / Freepik”