

# Microsoft 「Power Automate」 はテスト自動化ツールたり得るか

Quality Assurance Service  
with high performance  
and good agility



日本ナレッジ株式会社  
山本 涼平

# 自己紹介



## 山本 涼平（やまもと りょうへい）

### 所属

日本ナレッジ株式会社

検証事業部 検証部

札幌開発・自動化センター マネージャー

### 経歴

エンジニア歴 8年くらい

日本ナレッジ入社7年目

2年前からテスト自動化チームのリーダー

# こんな会社です

## 日本ナレッジ株式会社

本社所在地 〒111-0042 東京都台東区寿3-19-5 JSビル9階

拠点 札幌事業所、郡山センター、つくばセンター、成田センター、諏訪センター、名古屋センター

設立 1985年10月

代表取締役 藤井 洋一

資本金 8,600万円

従業員数 330名 (2022年4月時点)

売上 32.6億円 (2021年度：第37期)予定

### 主な加盟団体

- SAJ 一般社団法人 ソフトウェア協会
- iVIA 一般社団法人 IT検証産業協会



# こんなことをやっています

## ■ NKC.JAM (NKC Joins all of Automation test Methods)

### JAM.Consulting

テスト自動化導入コンサルティングサービス



テスト自動化を始めようとしている方を対象にコンサルティングを行うサービス。プロジェクトの課題や対象システムのヒアリングを行い、その後実際に一部をトライアル実装しての実現性調査などを行い、今後の導入をサポート。

### JAM.Solution

テスト自動化支援サービス



テスト自動化は専門知識が必要であり、ノウハウがない状態での新規立ち上げは難しい。そこで日本ナレッジの自動化エンジニアがプロジェクトに参加し、自動化の計画～実装、その後の運用まで支援を行うサービス。

### JAM.Session

テスト自動化セミナー



自動化チーム創設のための教育や、社内の自動化推進のための啓発活動として、日本ナレッジのテスト自動化スペシャリストがセミナーを実施。ハンズオンセミナーでは実際に自動化ツールの利用体験を通して、テスト自動化の理解を深める。

# 目次

---

- ・はじめに
- ・経緯
- ・テスト自動化ツール要件
- ・Power Automateの課題
- ・課題の解決策
- ・その他の情報
- ・結論
- ・まとめ

# はじめに

## 前提

---

### 注意事項

本資料はMicrosoft社およびPower Automateを推奨・否定するものではありません

### ツールの使い方の話がメインではない

ちょっとツールの技術的な話は出ますが、詳細な説明は省きます

Power Automateを使ったことがない方は、技術部分は参考程度にご覧ください

### 記載の内容は2022年5月時点での情報

6月30日のアップデートの情報は未反映

## 今日話したいこと

---

### 自動化ツール選定で考慮すべきこと

自動化ツールを選ぶ際、まずは目的に応じて要件を定める

### Power Automateはテスト自動化ツールとして使えるのか？

実際に調査した結果を解説

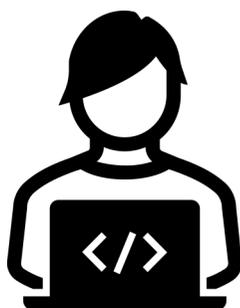
### Power Automateをどうやってテスト自動化ツールとして使うか？

実際にどうやって対処したのか？一部の実装例を元に説明

# 経緯

## 経緯①

自動化を推進しようにも、やはり「言語スキル」と「ツール価格」の壁が大きいと感じていた  
無償の(または安価の)ローコードツールとして、Power Automateを使えないか? と考えた



QA担当者の多くがプログラミング言語スキルを持たない

スクリプトコーディングの教育は大変

ローコードツールは色々あるが、費用は高め

Windowsアプリ向けの無償or安価ツールがあまりない

**Power Automateなら無償(安価)でWindowsアプリも対応**

## 経緯②

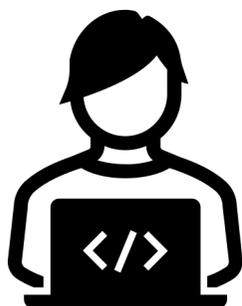
そんな折、ある販売管理系Windowsアプリのプロジェクトでテスト自動化を行うことに顧客の要望もありPower Automateを使ってみることになった



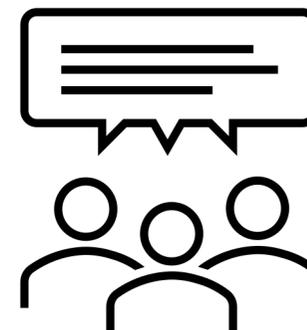
行く行くは自分たちで運用したいので、安くて簡単なツールを使いたい

**Power Automateを使って自動化できないか相談したい**

Power Automate調べたい人



Power Automate検討している人

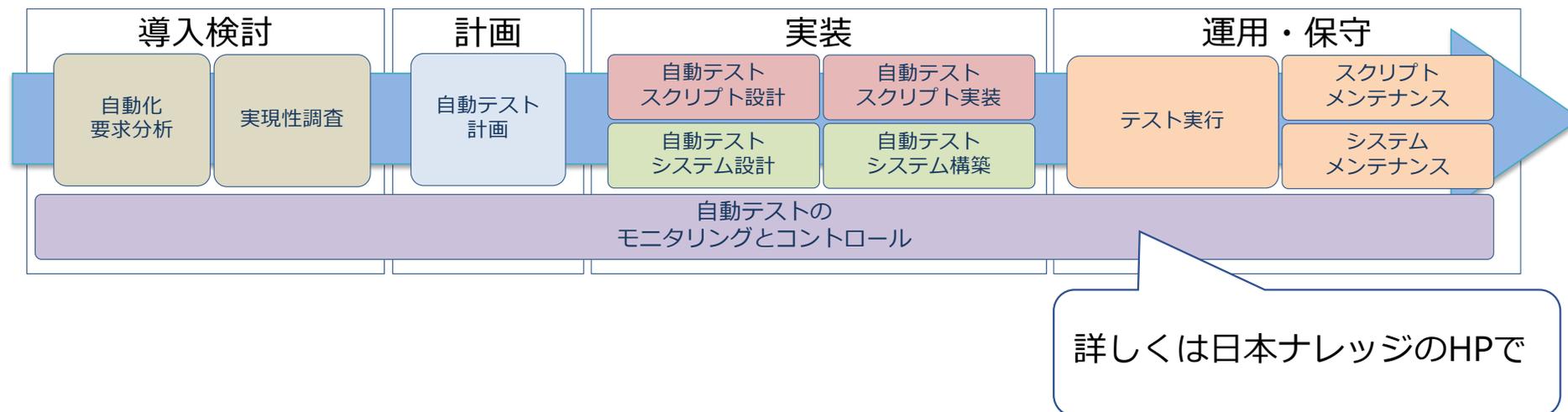


**Power Automateを使った  
テスト自動化の実現性調査**

# テスト自動化ツール要件

# テスト自動化ツールの選定

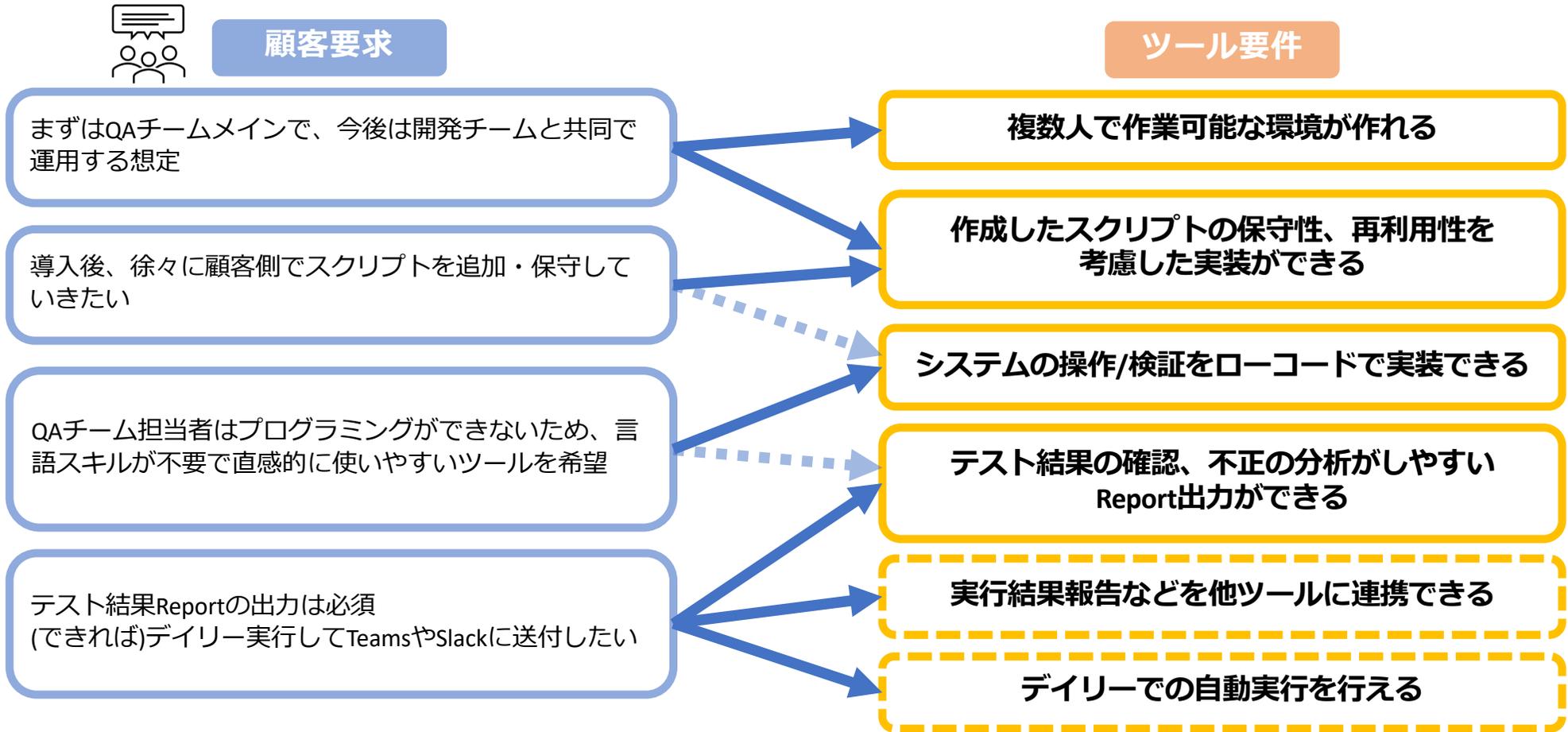
## ■テスト自動化プロセス（独自）



今回はPower Automateの導入を前提としているため、実現性調査で「**Power Automateが顧客の自動化要求を満たすか？**」を調査する

# テスト自動化ツール要件

## ■プロジェクトでの活用に必要なツール要件を洗い出す



# テスト自動化ツール要件まとめ

---

## ■ 今回のプロジェクト向けのテスト自動化ツール要件

要件1

対象システムの操作/検証をローコードで実装できる

要件2

作成したスクリプトの保守性、再利用性を考慮した実装ができる

要件3

テスト結果の確認、不正の分析がしやすいReport出力ができる

要件4

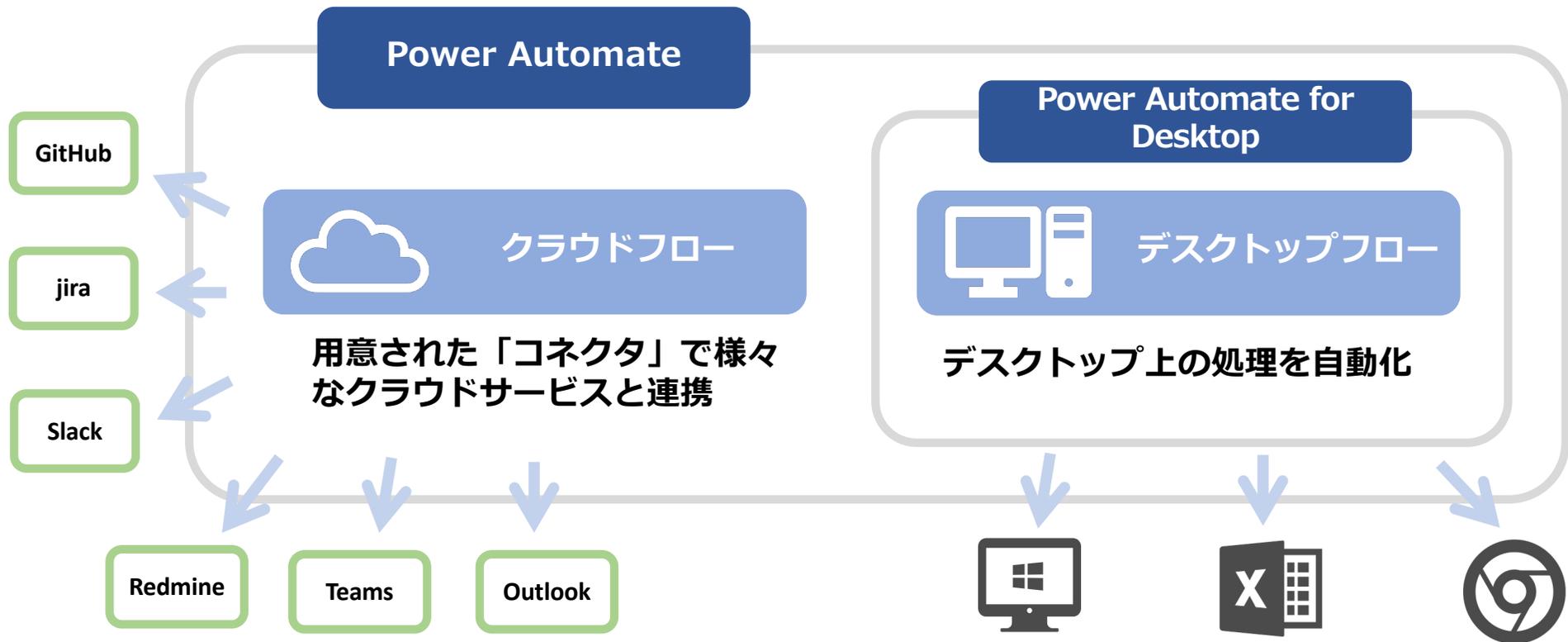
自動実行、不具合報告など、他ツール連携を含めた自動テストシステムとしての運用面の機能がある

ツール要件は洗い出せたので、Power Automateが上記を満たすか？を調査する

# Power Automate の課題

# Power Automateとは

クラウドサービスとの連携はクラウドフローで、PC上の操作はデスクトップフローで実行



# Power Automate for Desktop (PAD) の特徴

The screenshot shows the Power Automate for Desktop (PAD) interface. On the left is the 'アクション' (Actions) pane with a search bar and a list of categories including '変数' (Variables), '条件' (Conditions), 'ループ' (Loops), 'フロー-コントロール' (Flow Control), 'フロー-実行する' (Run Flow), 'システム' (System), 'ワークステーション' (Workstation), 'スクリプト' (Scripts), 'ファイル' (Files), 'フォルダー' (Folders), '圧縮' (Compression), 'UI オートメーション' (UI Automation), 'HTTP', and 'プラグイン自動化' (Plugin Automation). A callout box labeled '標準コマンド' (Standard Commands) points to the 'プラグイン自動化' category. The main workspace shows a flow with a step 'メッセージを表示' (Display Message) with a callout box labeled 'フローの引数、戻り値' (Flow Arguments, Return Values). The '変数' (Variables) pane on the right shows '入出力変数 2' (Input/Output Variables 2) with 'NewInput' and 'NewOutput' buttons, and 'フロー変数 1' (Flow Variable 1) with a 'ButtonPressed' button. A callout box labeled 'フロー内の変数' (Variables in Flow) points to the 'ButtonPressed' variable. On the right, the 'UI 要素' (UI Elements) pane shows a search bar, a 'UI 要素の追加' (Add UI Element) button, and a list of elements including 'Computer' and 'Quick Launch Bar'. A callout box labeled '画面要素を保存' (Save Screen Elements) points to the 'UI 要素の追加' button. At the bottom, the status bar shows 'ステータス: 準備完了' (Status: Ready), '0 選択されたアクション' (0 Selected Actions), '1 アクション' (1 Action), '1 サブフロー' (1 Subflow), and '実行遅延 100 ミリ秒' (Execution Delay 100 ms).

ローコードのテスト自動化ツールと似た感じの作り

インストールするだけですぐに使える上、Windows 11ではプリインストール

→導入も簡単で、直感的に使いやすい見た目

# 実際に使ってみて調査

## テスト自動化ツール要件まとめ

### ■ 今回のプロジェクト向けのテスト自動化ツール要件

要件1

対象システムの操作/検証をローコードで実装できる

要件2

作成したスクリプトの保守性、再利用性を考慮した実装ができる

要件3

テスト結果の確認、不正の分析がしやすいReport出力ができる

要件4

自動実行、不具合報告など、他ツール連携を含めた自動テストシステムとしての運用面の機能がある

ツール要件は考えたので、Power Automateが上記を満たすか？を調査する

前述のツール要件を満たすのか、実際に色々と試して確認してみた

# 要件1：ローコード実装

## 要件1

対象システムの操作/検証をローコードで実装できる

### ■対象システムの操作

コマンドを組み合わせるだけで操作を実装可能

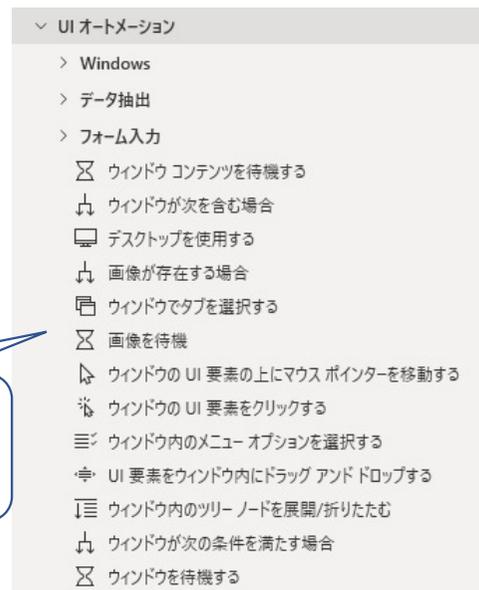
他自動化ツールと比較しても、実装自体はそれほど差はない

### ■対象システムの検証

PADの機能として検証用のコマンドは用意されていない

### ★課題1：Validation用コマンドがない

待機や条件分岐など、  
コマンド自体は豊富



## 要件2：スクリプトの保守性・再利用性

---

### 要件2

作成したスクリプトの保守性、再利用性を考慮した実装ができる

#### ■ 操作単位でのModule化（部品化）

Moduleに近い機能として、デスクトップフローには「サブフロー」という機能はあるものの、他デスクトップフローから呼び出して使うことができない

★課題2：デスクトップフローには汎用Module的な使い方をできる機能がない

## 要件3 : テスト結果 Report出力

---

### 要件3

テスト結果の確認、不正の分析がしやすいReport出力ができる

#### ■ Report出力

確認ポイント毎の判定結果、判定時の画面キャプチャ、テスト実行時間などのReportを出力したい

Report的な機能がPADの機能として存在しない

★課題3 : デスクトップフローにはReport的な操作結果を出力できる機能がない

## 要件4：自動テストシステム環境 (1)

### 要件4

自動実行、不具合報告など、他ツール連携を含めた自動テストシステムとしての運用面の機能がある

#### ■複数人での実装環境（フロー共有）

無償版では作成したフローをそのまま共有できない

共有する場合にはお互いが有償アカウントを利用していることが必須

#### ★課題4：無償では作成したフローの共有ができない

## 要件4：自動テストシステム環境 (2)

### 要件4

自動実行、不具合報告など、他ツール連携を含めた自動テストシステムとしての運用面の機能がある

#### ■ 複数人での実装環境（バージョン管理）

- ・ Gitなどのソースコード管理ツールとの連携はできない
- ・ 複数人が同時編集した内容のマージができず、インポート時に上書きされる
- ・ 同一名称のデスクトップフローを複数保存することができず、複数リビジョンの保持ができない

#### ★ 課題5：バージョン管理ができない

## 要件4：自動テストシステム環境 (3)

### 要件4

自動実行、不具合報告など、他ツール連携を含めた自動テストシステムとしての運用面の機能がある

#### ■ 結果報告の他ツール連携

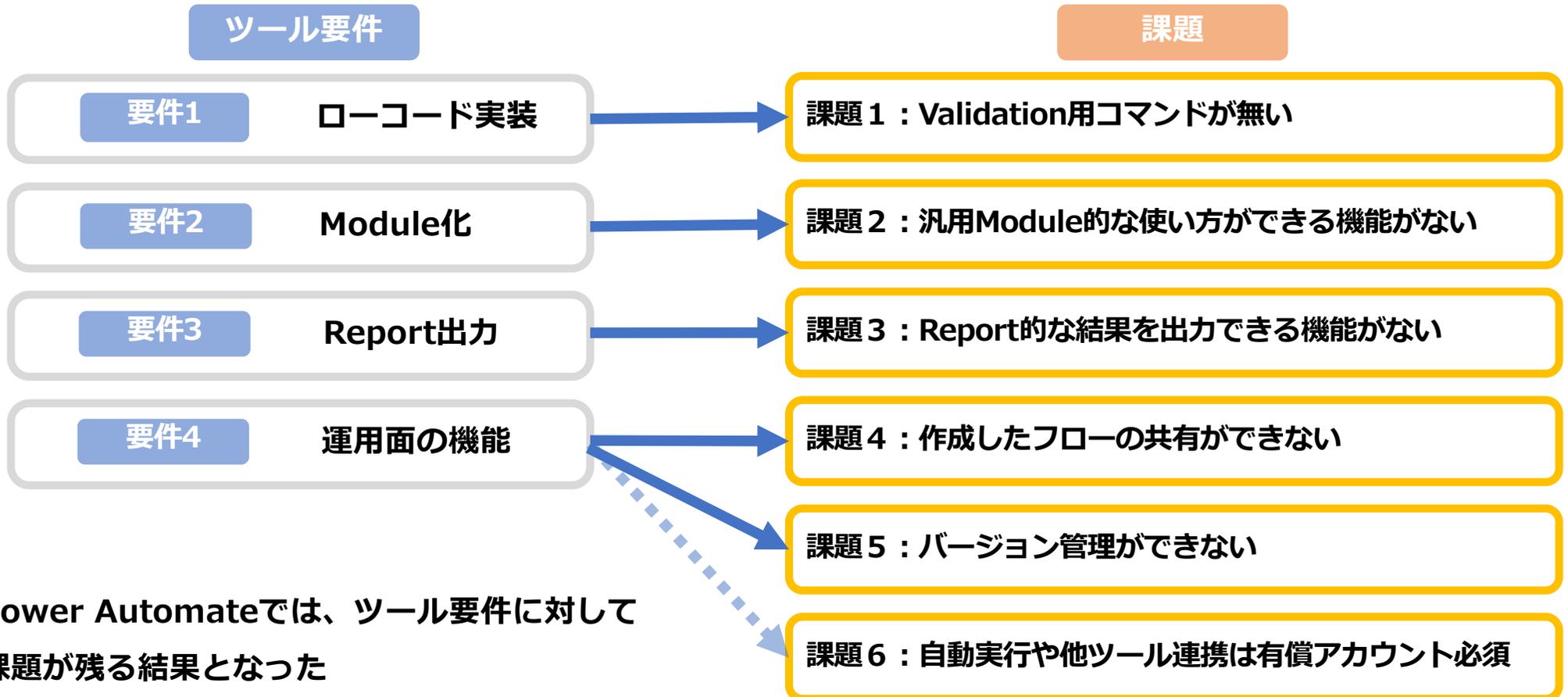
各クラウドツールへ投稿する場合は、有償アカウントでクラウドフローから連携する必要あり

#### ■ 自動実行

デスクトップフロー単体では自動実行は不可能であり、有償アカウントでクラウドフローからのトリガー実行が必要

★課題6：自動実行や他ツール連携は有償アカウントでクラウドフロー側での制御が必要

# テスト自動化ツール要件に対する課題



Power Automateでは、ツール要件に対して課題が残る結果となった

## 【TIPS】 テスト自動化ツールとRPAツール

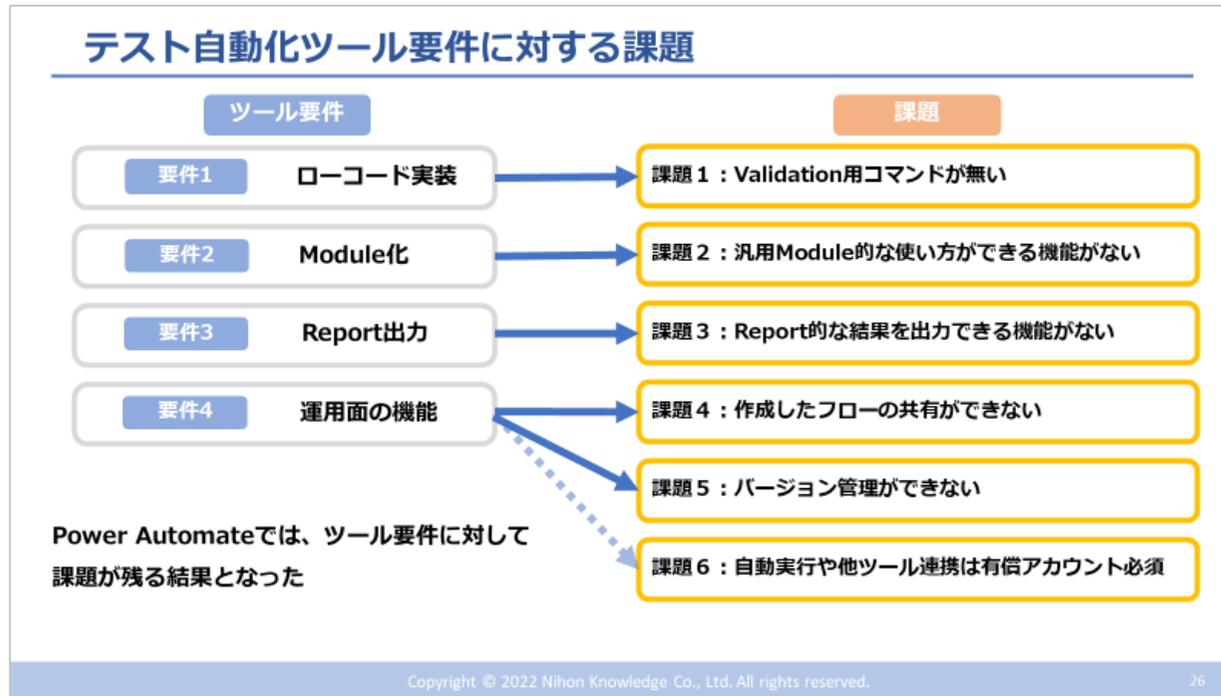
### ■ テスト自動化ツールとRPAツールの違い

テスト自動化ツールとRPAツールでは、そもそもの目的が違うので作りに違いがある

	テスト自動化ツール	RPAツール
Validation	テスト自動化では必須であるため どんなツールでも実装されている	RPAの目的は業務フローの実行であるため <b>Validationコマンドはほとんど無い</b>
保守性	テスト対象、環境の変化によるメンテナ ンスが発生するため、保守性を考慮 した作りになっている	Module化自体はできるものも多いが、RPAで は <b>高頻度でのメンテナンスはあまり考慮され ていない</b>
Report	Report機能は必須であり、詳細なカス タマイズが可能なものが多い	基本的には無いが、簡素なもののみ
共有	スクリプトそのものをローカルで渡す ことも、管理ツール経由で渡すことも 可能	ツールによるが、クラウド経由などで実行環 境に渡すことなどが可能
バージョン管理	殆どのツールがGitなどのバージョン管 理ツールに対応 ツールによっては独自の管理ツールを 採用している場合もある	シナリオをxml形式で出力することでバー ジョン管理ツールに組み込めるものもある

# 課題の解決策

# 課題の解決策の検討



色々課題はあるものの、導入しやすさや使いやすさは魅力的

テスト自動化への活用のため、各課題が解決できないか調査した

# 課題1 「Validation用コマンドがない」の解決策

## 課題 1 : Validation用コマンドが無い

### ■ 解決策 : 標準コマンドを組み合わせてValidationを作る

画面上の値を取得 (実測値)

ウィンドウにある UI 要素の詳細を取得する

🔍 ウィンドウにある UI 要素の属性値を取得する [詳細](#)

パラメーターの選択

全般

UI 要素:

詳細

属性名:

> 生成された変数 **Actual**

エラー発生時

保存 キャンセル

テストデータとして期待値を持たせる

	A	B
1	No	メッセージダイアログ文言
2	1	正常に処理されました
3	2	処理に失敗しました
4	3	パラメータが不足しています

コマンドで比較して、OK/NGを判定

# 課題1の解決策（実装詳細）

## ■ 判定部分はこんな感じ

The screenshot shows a workflow editor with the following steps:

- If Actual = Expected then**
  - 変数の設定** (Variable Setting): 変数 **Result** に値 **Pass** を割り当てる (Assign value **Pass** to variable **Result**)
- Else**
  - 変数の設定** (Variable Setting): 変数 **Result** に値 **Fail** を割り当てる (Assign value **Fail** to variable **Result**)
  - サブフローの実行** (Execute Subflow): サブフロー **CreateEvidenceFileName** を実行する (Execute subflow **CreateEvidenceFileName**)
  - スクリーンショットを取得** (Take Screenshot): プライマリ画面のスクリーンショットを取得して **EvidenceFolderPath** '\ **EvidenceFileName** に保存 (Take screenshot of primary screen and save to **EvidenceFolderPath** '\ **EvidenceFileName**)
- End**
- テキストをファイルに書き込みます** (Write text to file): **TempFilePath** に '{CategoryNo: ' **CategoryNo** ', 'CaseNo: ' **CaseNo** ', 'Expected: ' **Expected** ', 'Actual: ' **Actual** ', 'Result: ' **Result** ', 'Evidence: ' **EvidenceFileName** }' を書き込みます (Write '{CategoryNo: ' **CategoryNo** ', 'CaseNo: ' **CaseNo** ', 'Expected: ' **Expected** ', 'Actual: ' **Actual** ', 'Result: ' **Result** ', 'Evidence: ' **EvidenceFileName** }' to **TempFilePath**)

Callouts explain the logic:

- 「条件」の「if」コマンドで、実測値(Actual)と期待値(Expected)を比較
- 比較結果に応じて「Result」変数に値を割り当て
- 失敗時のスクリーンショットを取得する
- 一時ファイルに判定結果を保存

# 課題1の解決策（実装詳細）

## ■ 特定画面要素の有無の場合

▼ 1. ウィンドウが次を含む場合  
UI 要素 Dialog リスト付メッセージ がウィンドウ に存在する場合

2. ウィンドウにある UI 要素の詳細を取得する  
UI 要素 Label リスト付メッセージ の属性 'Own Text' を取得し、Actual に保存します

▼ 3. Else

{x} 4. 変数の設定  
変数 Actual に値 'NotExist' を割り当てる

5. End

6. Desktop フローを実行  
名前を指定してフローを実行: ValidateString

「ウィンドウが次を含む場合」コマンドで、期待する画面要素有無を判定する

「Actual」変数に対象画面要素の文字列か、Not Existの値を割り当てる

前ページの「条件」での期待値判定を行う

▼ 1. If Actual = Expected then

{x} 2. 変数の設定  
変数 Result に値 Pass を割り当てる

▼ 3. Else

{x} 4. 変数の設定  
変数 Result に値 Fail を割り当てる

5. サブフローの実行 サブフロー CreateEvidenceFileName を実行する

6. スクリーンショットを取得  
プライマリ画面のスクリーンショットを取得して EvidenceFolderPath \ EvidenceFileName に保存

7. End

8. テキストをファイルに書き込みます  
TempFilePath に {CategoryNo}; {CategoryNo}; {CaseNo}; {CaseNo}; Expected: {Expected}; Actual: {Actual}; Result: {Result}; Evidence: {EvidenceFileName} を書き込みます

期待値に対象画面要素の文字列を設定しておくことで、「ウィンドウが次を含む場合」がTrueならPass、FalseならFailになる

# 課題1の解決策（実装詳細）

## ■ 画像判定の場合

The screenshot displays a workflow editor with the following steps:

- Image Existence Check:** A process with a dropdown menu containing:
  - Image Existence Case:** A step for setting variables where 'Actual' is assigned the value 'Exist'.
  - Else:** A step for setting variables where 'Actual' is assigned the value 'NotExist'.
  - End:** The process concludes.
- Desktop Flow Execution:** A step to execute a flow named 'ValidateString'.

Callouts provide additional context:

- 「画像が存在する場合」コマンドで、指定画像有無を判定する（画像は埋め込み）
- 判定結果に応じて「Actual」変数にExists、Not Existsの値を割り当てる
- 前々ページの「条件」での期待値判定を行う
- 「画像が存在する場合」がTrueならPass、FalseならFailになる

The workflow continues with an **If** statement: `If Actual = Expected then`. Inside the **Else** branch, it sets `Result` to `Fail`, executes a sub-flow `CreateEvidenceFileName`, and takes a screenshot of the primary screen, saving it to `EvidenceFolderPath \ EvidenceFileName`. Finally, it writes the following text to a file:

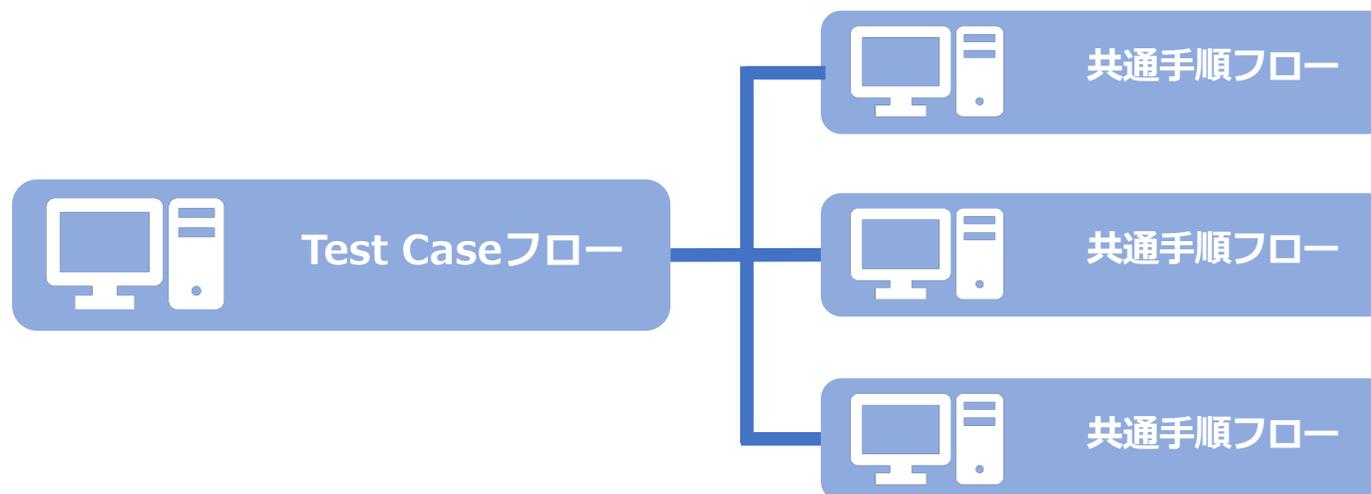
```
テキストをファイルに書き込みます  
TempFilePath に [{"CategoryNo": CategoryNo, "CaseNo": CaseNo, "Expected": Expected, "Actual": Actual, "Result": Result, "Evidence": EvidenceFileName}] を書き込みます
```

## 課題2 「Module機能が無い」の解決策

課題2 : 汎用Module的な使い方ができる機能がない

### ■ 解決策 : デスクトップフローをModuleとして使用する

共通Moduleとして汎用化した「**共通手順フロー**」を作成し、それをTest Caseのデスクトップフローから呼び出して実行する



## 課題2の解決策（実装詳細）

### ■ Module化したデスクトップフローを呼び出して実行

Test Caseフロー

ブフロー	Main
1	▶ アプリケーションの実行 アプリケーション 'test' を実行し、そのプロセス ID を <code>AppProcessId</code> に保存します
2	📄 Desktop フローを実行 名前を指定してフローを実行: <code>OpenProgram</code>
3	📄 Desktop フローを実行 名前を指定してフローを実行: <code>全体_ログイン</code>
4	⊗ プロセスを終了する 名前が 'Test' のすべてのプロセスを終了する

共通手順化したデスクトップフローを呼び出す

共通手順フロー

ブフロー	Main
1	0=0 テキストを数値に変換 テキスト <code>WaitTimeout</code> を数値に変換し、 <code>WaitTimeout</code> に保存する
2	🕒 ウィンドウ コンテンツを待機する ウィンドウ に UI 要素 <code>Window ログイン</code> が表示されるまで待機する
3	📄 ウィンドウ内のテキスト フィールドに入力する テキスト ボックス <code>Edit パスワード</code> に <code>Password</code> を入力する
4	🖱️ ウィンドウ内のボタンを押す ボタン <code>Button OK</code> を押す
5	🕒 ウィンドウを待機する ウィンドウ <code>Window メイン</code> が開くまで待機

変数の検索

入出力変数 2

- (x) Password
- (x) WaitTimeout

フロー変数 0

表示

フローを実行し終わったら元のフローに戻る

## 課題3 「Report出力機能がない」の解決策

### 課題3 : Report的な結果を出力できる機能がない

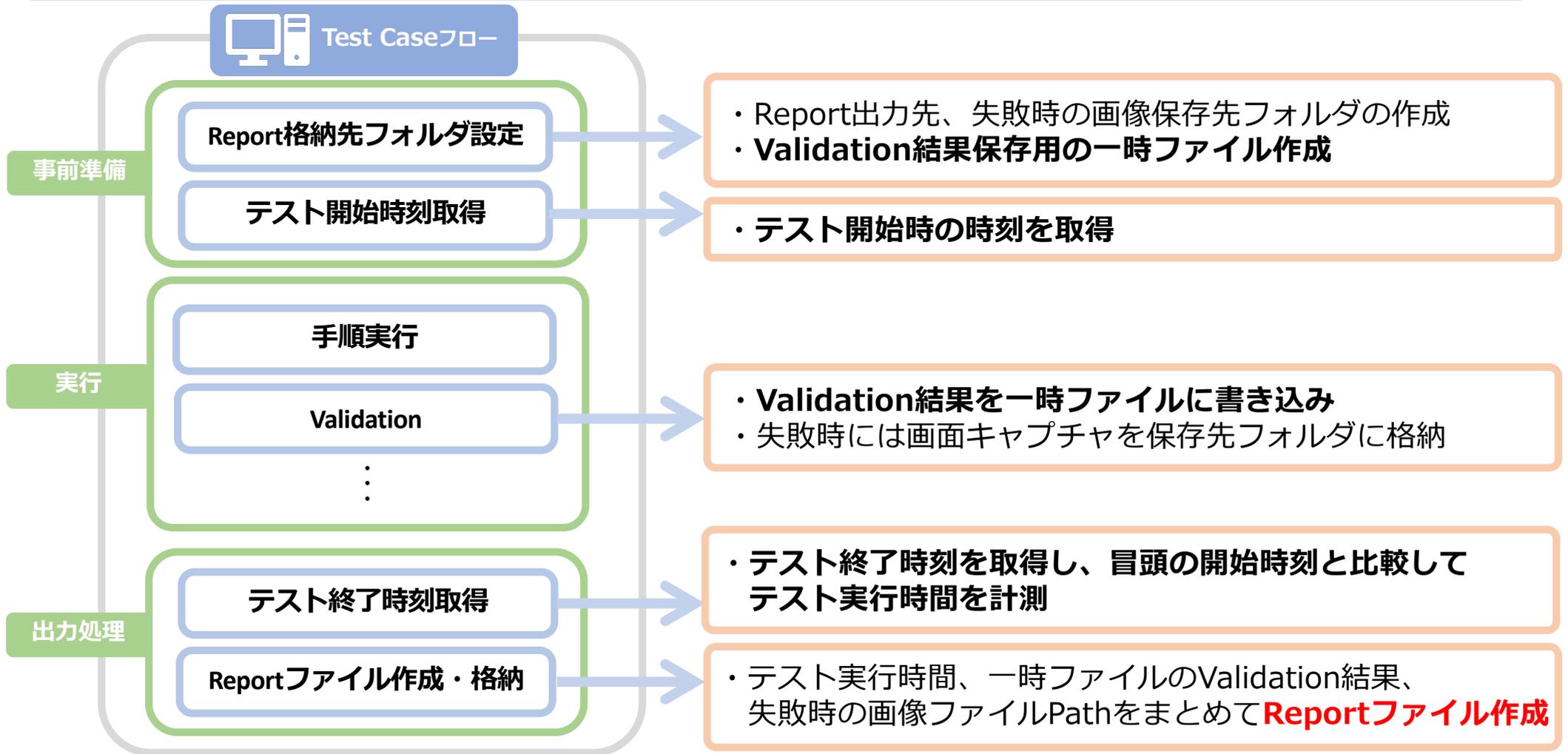
#### ■ 解決策 : 標準コマンドを組み合わせて、Report出力フローを作る

課題1解決策のValidationコマンド実行後に結果を退避させて置き、テストが完了した時点でまとめてReportファイルにする



「Result」を退避させておく

## 課題3の解決策（実装詳細）



# 課題3の解決策（実装詳細）

## ■ Report出力フローの例(一部)

13 ファイルからテキストを読み取ります  
ファイル TempFilePath の内容を読み取る

14 For each ResultItem in ResultList

15 JSON をカスタム オブジェクトに変換  
JSON ResultItem をカスタム オブジェクト ResultObject に変換

16 コメント  
Note: 先頭データは一時データに格納する

17 If TestCount = 0 then

18 変数を大きくする  
変数 TestCount を 1 大きくする

19 (x) 変数の設定  
変数 CurrentCategoryNo に値 ResultObject [CategoryNo] を割り当てる

20 (x) 変数の設定  
変数 CurrentCaseNo に値 ResultObject [CaseNo] を割り当てる

21 =: 項目をリストに追加  
項目 ResultObject [Expected] をリスト ExpectedList に追加

22 =: 項目をリストに追加  
項目 ResultObject [Actual] をリスト ActualList に追加

23 If ResultObject [Result] = Fail then

24 (x) 変数の設定  
変数 CurrentCaseResult に値 Fail を割り当てる

25 =: 項目をリストに追加  
項目 ResultObject [Evidence] をリスト EvidenceList に追加

26 End

27 > 次のループ

28 End

Validationコマンドで退避しておいた結果を読み取る

42 テキストの結合  
リスト ExpectedList の項目を','で区切って結合する

43 Excel ワークシートに書き込み  
Excel インスタンス ExcelInstance の列 ExpectedColumn および行 TestRow のセルに値 Expected を書き込み

44 テキストの結合  
リスト ActualList の項目を','で区切って結合する

45 Excel ワークシートに書き込み  
Excel インスタンス ExcelInstance の列 ActualColumn および行 TestRow のセルに値 Actual を書き込み

46 Excel ワークシートに書き込み  
Excel インスタンス ExcelInstance の列 ResultColumn および行 TestRow のセルに値 CurrentCaseResult を書き込み

Excelファイルに期待値、実測値、結果などを書きこむ

# 課題3の解決策 (実装詳細)

## ■ 出力Reportファイル例

	A	B	C	D
1	開始日時	20220616_134143		
2	終了日時	20220616_135301		
3	実行時間 (秒)	678.2611405		
4	テスト結果	Fail		
5	テスト件数		19	
6	Pass 件数		17	
7	Fail 件数		2	
8				
9				
10				

結果サマリーに「テスト実行時間」「テスト全体のPass/Fail件数」を記載

	A	B	C	D	E	F	G	H
1	CategoryNo	CaseNo	TestDataSheet	DataNo	Expected	Actual	Result	Evidence
2	1	1			Exist	NotExist	Fail	xxxx_0101_20220616_134143.png
3	1	2			この伝票を登録してよろしいですか?	この伝票を登録してよろしいですか?	Pass	
4	1	3	共通_xxxx	1	30, 30, 30	30, 30, 30	Pass	
5	2	1			Exist	NotExist	Fail	xxxx_0201_20220616_134143.png
6	2	2			この伝票を登録してよろしいですか?	この伝票を登録してよろしいですか?	Pass	
7	2	3	共通_xxxx	2	30, 30, 30	30, 30, 30	Pass	
8	3	1			Exist	Exist	Pass	
9	3	2			編集中の内容を登録してよろしいですか?	編集中の内容を登録してよろしいですか?	Pass	
10	3	3			Exist	Exist	Pass	
11	3	4			編集中の内容を登録してよろしいですか?	編集中の内容を登録してよろしいですか?	Pass	
12	4	1			Exist	Exist	Pass	
13	4	2			処理が完了しました。	処理が完了しました。	Pass	
14	4	3	共通_xxxx	3	40, 40, 40	40, 40, 40	Pass	
15	4	4			Exist	Exist	Pass	
16	4	5			処理が完了しました。	処理が完了しました。	Pass	
17	4	6	共通_xxxx	4	30, 30, 30	30, 30, 30	Pass	
18	4	7	yyyy_在庫確認	1	1, 10, 10, 0, 0, 10, 0, 0, 0, 10, 0	1, 10, 10, 0, 0, 10, 0, 0, 0, 10, 0	Pass	
19	4	8	yyyy_在庫確認	2	2, 20, 20, 0, 0, 20, 0, 0, 0, 20, 0	2, 20, 20, 0, 0, 20, 0, 0, 0, 20, 0	Pass	
20	4	9	yyyy_在庫確認	3	3, 30, 30, 0, 0, 30, 0, 0, 0, 30, 0	3, 30, 30, 0, 0, 30, 0, 0, 0, 30, 0	Pass	
21								

- テスト手順の項番
- 期待値 & 実測値
- 実施結果
- 失敗時の画像格納Path etc

## 課題4 「作成したフローの共有ができない」の解決策

### 課題4：作成したフローの共有ができない

#### ■ 解決策：なし（有償アカウントが必要）

無償ライセンスでは作成したデスクトップフローを共有することができない  
複数人で運用するのであれば有償アカウントが必須

※共有する側・される側の両方が有償アカウントであることが必要

複数人で運用するなら、少なくとも  
 $¥1,630 \times 2 = ¥3,260/\text{月}$  は掛かる

ユーザーごとのライセンス	ユーザーごとのライセンス	フローごとのライセンス
ユーザーごとのプラン	アテンダ型 RPA のユーザーごとのプラン	フローごとのプラン
<b>¥1,630</b> 毎月ユーザーあたり	<b>¥4,350</b> 毎月ユーザーあたり	<b>¥10,870</b> 毎月フローごと。最小5フロー <sup>2</sup>
各ユーザーがプロセスを分析し、独自のニーズに基づいてクラウドフローを無制限に作成できます。	個々のユーザーがプロセスを分析し、クラウドフローを無制限に作成でき、さらに <b>ロボティックスプロセスオートメーション(RPA)</b> と AI を経由してレガシーアプリケーションを自動化できます。	組織全体の無制限ユーザー向けに予約したキャパシティでクラウドフローを実装します。
<ul style="list-style-type: none"><li>クレジットカードで今すぐ購入できます。</li></ul>	<ul style="list-style-type: none"><li>毎月5,000の <b>AI Builder</b> サービスクレジットを含みます。</li><li>クレジットカードで今すぐ購入できます。</li></ul>	<ul style="list-style-type: none"><li><b>グローバル管理者または課金管理者ロール</b> で Microsoft 365 管理センターへのアクセス許可が必要です。</li></ul>

## 課題5 「バージョン管理ができない」の解決策

### 課題5：バージョン管理ができない

#### ■ 解決策：バージョン番号付与と定期的なバックアップによる運用対処

##### ▼ Power Automateクラウド環境の問題点

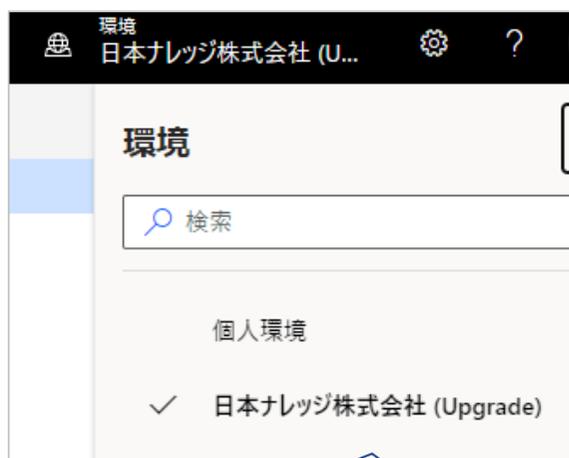
- ・クラウドでは同一名称のフローを複数置けない
- ・クラウド画面でフォルダ分けができない
- ・デスクトップフロー編集後、修正前へのロールバックができない
- ・同一フローをインポートした場合、差分をマージできない（上書き）

解決策がないため、**運用ルールを決めるなどして対処**するしかない

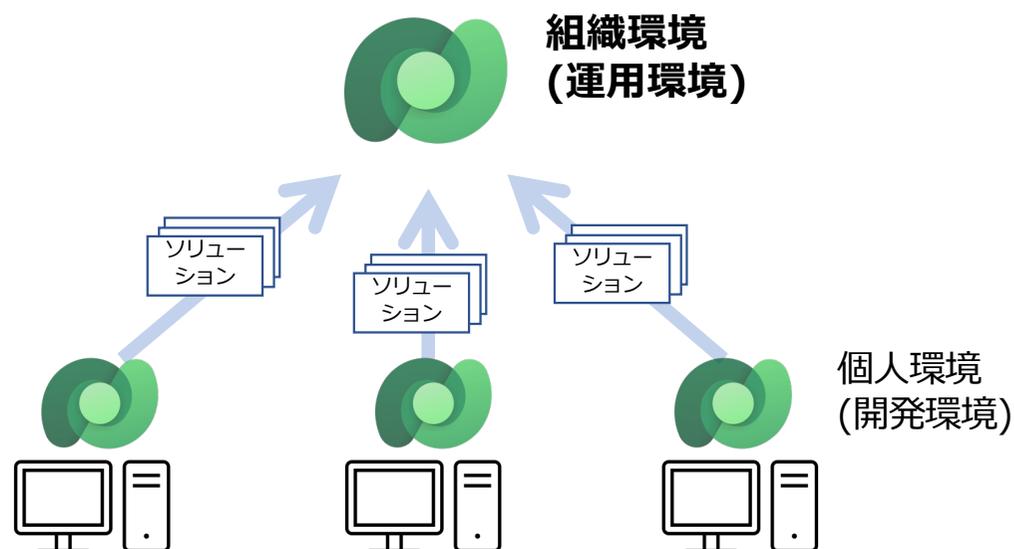
## 課題5の解決策

### ■ 運用ルール1：複数環境を用意して開発を行う

「編集後、修正前へのロールバックができない」ため、クラウドの共有環境上で直接編集は危険  
そのためフローの開発環境と、フローを実行する運用環境を分けて運用する



有償ライセンスを契約すると(容量の関係で)  
「環境」を追加で作成することができる



## 課題5の解決策

### ■ 運用ルール2：バージョン番号などで状態を可視化する

運用環境のフローがどんな状態なのか？をバージョン番号で可視化することで、  
誤って編集したり上書きしたりするリスクを回避する

バージョン番号による  
運用ルール例

**Version 1.0.0.0**

メジャー   マイナー   ビルド   リビジョン

- ・メジャー  
対象システムの改変や管理方法の変更などにより、**フロー構成自体が変わる大規模な変更**の場合にインクリメント
- ・マイナー  
**他フローからの参照に影響がある**変更を行った場合（入出力変数の変更）や、新規フローを追加した場合にインクリメント
- ・ビルド  
既存フローの不具合修正など、**他フローへ影響を与えない変更**がFIXした場合にインクリメント
- ・リビジョン  
変更を加えた場合にインクリメントし、レビューに提出する  
FIX済みの場合はビルドをインクリメントし、リビジョンは「0」となる  
※つまりリビジョンが0以外の場合には、修正が未FIXの状態

ソリューション設定

表示名\*

名前\*

Scenario\_WarehouseManagement

公開元\*

Default Publisher for org3b8a00bc L...

+ 新しい公開元

バージョン\*

1.0.0.0

その他のオプション v

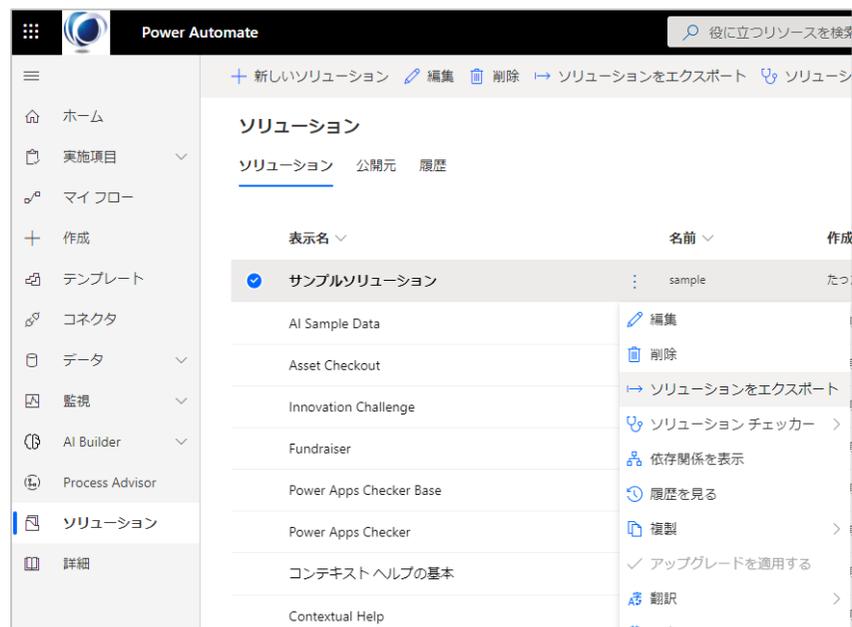
ルールに応じて値を修正

## 課題5の解決策

### ■運用ルール3：ソリューションを定期的にバックアップする

前述の二つの運用ルールを行ったとしても、誤って修正内容を上書きしてしまうことはある  
ロールバックができないため、定期的にソリューションのバックアップを取得しておく  
バックアップ契機については毎週○曜日、毎日、修正の度、などルール化する

Power Automateクラウド画面から、  
ソリューション単位でエクスポートしておく



## 課題6 「自動実行や他ツール連携は有償アカウント必須」の解決策

### 課題6：自動実行や他ツール連携は有償アカウント必須

#### ■ 解決策：なし（有償アカウントが必要）

SlackやJIRA、Redmineなどのツールと連携するには有償アカウントでクラウドフローが必要  
また自動実行については「アテンド型RPAユーザーごとのプラン」が必要

実行用のライセンスとして少なくとも  
¥4,350/月 のライセンスが1つ以上必要

#### ユーザーごとのライセンス

アテンド型 RPA のユーザーごとのプラン

**¥4,350**

毎月ユーザーあたり

個々のユーザーがプロセスを分析し、クラウドフローを無制限に作成でき、さらに ロボティクスプロセスオートメーション (RPA) と AI を経由してレガシー アプリケーションを自動化できます。

- 毎月 5,000 の AI Builder サービス クレジットを含みます。
- クレジットカードで今すぐ購入できます。

#### フローごとのライセンス

フローごとのプラン

**¥10,870**

毎月フローごと。最小 5 フロー<sup>2</sup>

組織全体の無制限ユーザー向けに予約したキャパシティでクラウド フローを実装します。

- グローバル管理者または課金管理者ロールで Microsoft 365 管理センターへのアクセス許可が必要です。

## 課題6の詳細

### ■ 自動実行について

自動実行はクラウドフローでトリガーを設定して行う

トリガーでクラウドフローが動き、そこからテストケースのデスクトップフローを実行することで、自動実行が行われる

スケジュールで自動実行されるクラウドフロー

テストケースであるデスクトップフローを呼び出して実行

完了後Slackなどにメッセージを投稿

Recurrence

毎日 に実行する 編集

デスクトップ用 Power Automate で構築したフローを実行する

\*Desktop フロー テストケース1 編集

\*実行モード アテンンド型 (サインイン時に実行)

詳細オプションを表示する

メッセージの投稿 (V2)

\*チャネル名 テスト自動化プロジェクト

\*メッセージテキスト テスト完了

詳細オプションを表示する

# 解決策まとめ

## ■ Power Automateの課題と解決策のまとめ

	課題内容	解決策
課題1	Validation用コマンドが無い	標準コマンドを組み合わせて新たに実装
課題2	汎用的なModule機能がない	デスクトップフローをModuleとして使用する
課題3	Reportを出力する機能がない	標準コマンドを組み合わせて新たに実装
課題4	作成したフローの共有ができない	※有償アカウントが必要
課題5	バージョン管理ができない	「複数環境を用意する」「バージョン番号で常態化を可視化する」「定期的にバックアップする」などで運用対処する
課題6	自動実行や他ツール連携	※有償アカウントが必要 クラウドフローで自動実行、他ツール連携を行う

結果として「有償アカウントを使って工夫すれば課題解決ができる」ことが分かった

# その他の情報

# よいところ

---

## ■ Power Automateのよいところ

- **Windowsアプリケーション向けのローコードツールとしては使いやすい**

無償（安価）のローコードツールはWindowsアプリケーション対応となると限定される

一部は存在するが環境面や操作の難易度が高く、比較するとPower Automateはかなり使いやすい部類と言える

- **RPAツールとしては優秀**

PADのコマンドやクラウドフローのコネクタの豊富さは非常に優秀で、RPAにおいては効果的

特にExcelなどのMicrosoft製品関連の操作は他ツールと比較しても充実

- **ライセンスが安い**

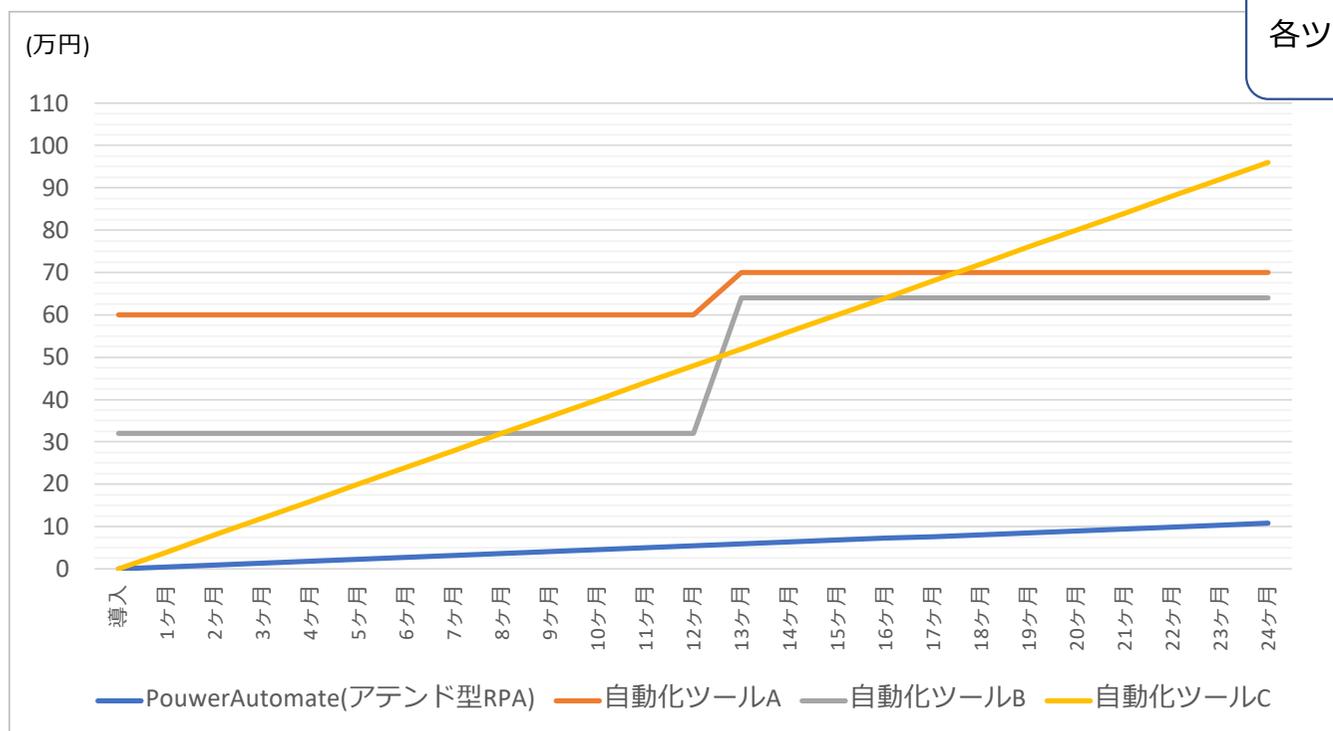
有償ライセンスを利用したとしても他テスト自動化ツールより安価で抑えられる

※次ページ参照

# ツール費用

## ■ テスト自動化ツールとの費用の比較

有償アカウントの費用はアテンド型RPAライセンスで4,350円/月、約5.3万円/年と一般的なテスト自動化ツールよりもかなり安価であり、コスト面での導入のしやすさはある



各ツールでアカウント一つを購入した場合の比較

- 自動化ツールA  
購入+保守費用(年)
- 自動化ツールB  
年額サブスクリプション
- 自動化ツールC  
月額サブスクリプション

# 結論

## 結論

---

- Power Automateはテスト自動化ツールたり得るか

### 結論：「大変だけど使える」

ただしカスタマイズしなければいけない部分が多く、運用面での注意点多いため

**「ローコードで簡単にテストを作れる」という考えであれば難しい**

## 今回のプロジェクトの状況

---

まずはテスト自動化を実現できることが分かったため、Power Automateでの自動化を推進中

### ■ 技術面の状況

- ValidationやReportなどの作りこみが必要な部分は用意できたので、今後はQA担当者でも使えそう
- 基本的な操作の実装については、ツールの標準コマンドで問題なくできている

### ■ 費用面の状況

- RPAアテンド型(4,350円/月)が必須にはなったものの、予算としては許容範囲内
- テスト自動化ツールを導入した場合との費用対効果の比較は計測中
  - カスタマイズにより**初期作業コストは自動化ツールよりも大きい**
  - 運用対処で対応しているため、**運用面での人的な作業工数は自動化ツールよりも大きい**
  - ツールの購入費用は掛からず保守費用が安価なため、**ツール費用は自動化ツールよりも小さい**

★今後も継続して計測していくが、テスト自動化ツールと同等かそれ以下のコストで運用できる見込み

# テスト自動化での有効活用方法

---

## ■ Power Automateのテスト自動化での有効活用

### ・ 小規模プロジェクト

課題である複数人作業とバージョン管理の影響を受けなければ活用しやすい

### ・ Microsoft製品、Windowsサービスの制御

Office製品、Windowsサービス、画面解像度設定など、他社製ツールでは制御が難しい部分に強い

### ・ Validationや詳細なReport出力が不要な作業

「テストの事前準備」や「テスト手順の実行」までを行うか、

Validationを目視で行うなどの部分的な自動化であれば比較的簡単に作れる

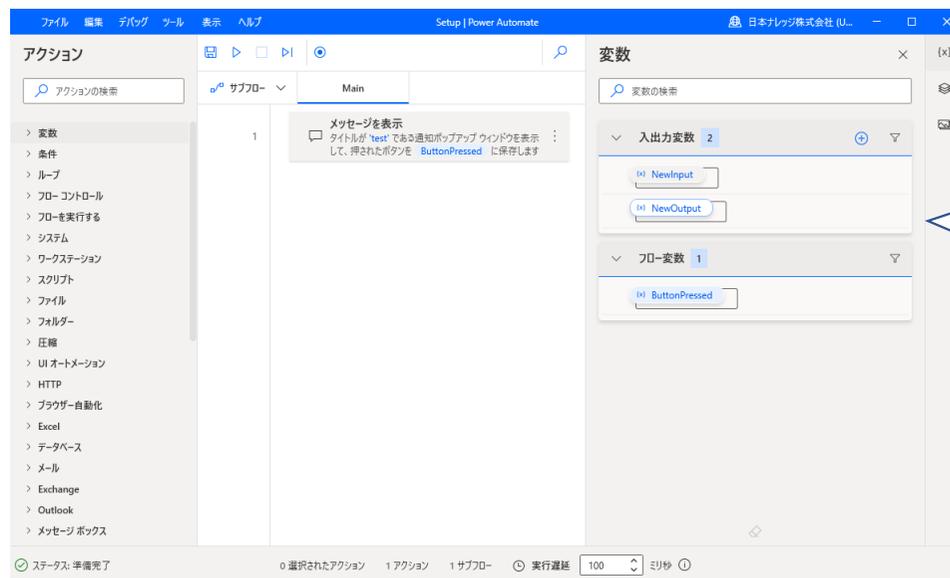
# 高頻度のアップデート

## ■ Power Automateのアップデート

割と高頻度でアップデートが行われており、今後の機能追加も計画されている

今は課題となっている部分でも、今後のアップデート次第では解決される可能性もある

<https://docs.microsoft.com/ja-jp/power-platform-release-plan/2022wave1/power-automate/planned-features>



6/30のアップデートで、入出力変数に色々なデータ型が使えるようになった

# まとめ

## まとめ

---

### □ 自動化ツール選定で考慮すべきこと

目的に応じてツールに必要なこと(要件)を定め、適したツールかどうかを検証する

### □ Power Automateはテスト自動化ツールとして使えるのか？

カスタマイズや運用方法の検討は必要だがテスト自動化ツールとしても使える

Microsoft製品を扱う場合や、小規模プロジェクトだと有効活用しやすい

### □ Power Automateをどうやってテスト自動化ツールとして使うか？

対処方法については各課題に対する解決方法を参考に

今後のアップデートでさらなる改善の可能性がある

END