Excelで簡単設計！
スクリプトジェネレータを活用した組み込みソフトのテスト自動化

2019/10/10

アイシン・ソフトウェア㈱
田中　泰史

本日のアジェンダ

1．会社紹介
2．背景
3．改善策
4．テスト自動化支援ツール
5．導入／展開
6．導入に対する効果
7．振り返り／今後の課題

１．会社紹介

アイシン・ソフトウェア株式会社

1-1. 会社概要

アイシン・コムクルーズと、エィ・ダブリュ・ソフトウェアは2019年10月1日に経営統合しアイシン・ソフトウェアとして新たなスタートを切りました。

社名	アイシン・ソフトウェア株式会社
資本金	9800万円
株主	アイシン精機㈱、アイシン・エィ・ダブリュ㈱
主な得意先	アイシン精機㈱、アイシン・エィ・ダブリュ㈱、㈱アドヴィックス、をはじめ アイシングループ各社
代表者	取締役社長　伊藤康伸
本社所在地	愛知県刈谷市相生町1丁目1番地1　アドバンス・スクエア刈谷
事業所	札幌・盛岡・名古屋・刈谷・岡崎・福岡
従業員数	735名（2019年10月）

SPI Japan 2019のタイムテーブルでは旧社名の
アイシン・コムクルーズ表記となっています

1-2. アイシングループの事業領域

領域：自動車部品事業および住生活・エネルギー関連事業

小型トラック・バス用FR6速
オートマチックトランスミッション

エンジン冷却用
電動ウォーターポンプ

アクティブ
リアステアリングシステム

サンルーフ

ベッド

シャワートイレ

乗用車用FR10速
オートマチックトランスミッション

2モーターハイブリッド
トランスミッション

電子制御ブレーキシステム

パワースライドドアシステム

家庭用燃料電池
コージェネレーションシステム

乗用車用FF6速
マニュアルトランスミッション

ピストン

駐車支援システム

塗布型制振材

ガスヒートポンプエアコン

出典：AISIN GROUP REPORT 2018より

2. 背景

AISIN

アイシン・ソフトウェア株式会社

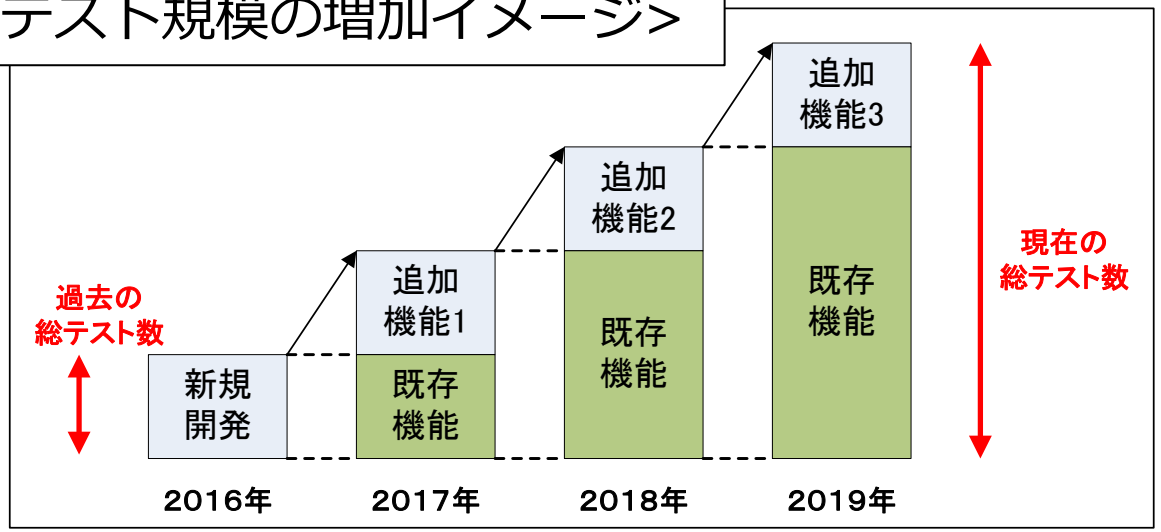
2-1. 背景

ソフトの高機能化に伴い、テスト工数が爆発的に増加

- 車載ソフトの高機能化がますます進展
  （派生開発の繰り返しと、搭載ROM容量の増加のため）
- クルマは量産までに何度も試作を繰り返すが、その度に膨大なテストが発生

機能追加の度に
テスト規模が拡大

<テスト規模の増加イメージ>

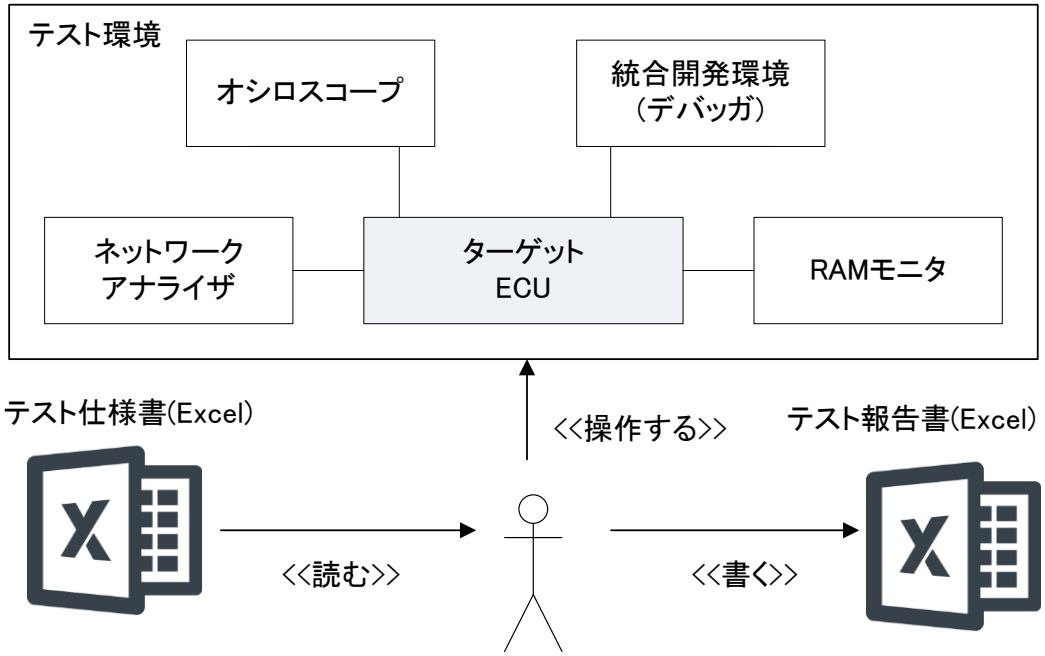


ソフト開発において、テストがより大きな比重を占めるようになった

2-2. 従来のテスト手法

従来のテストは人手頼み

- Excelテスト仕様書をもとにテストを実施し、結果を再びExcelテスト報告書に記入する手法

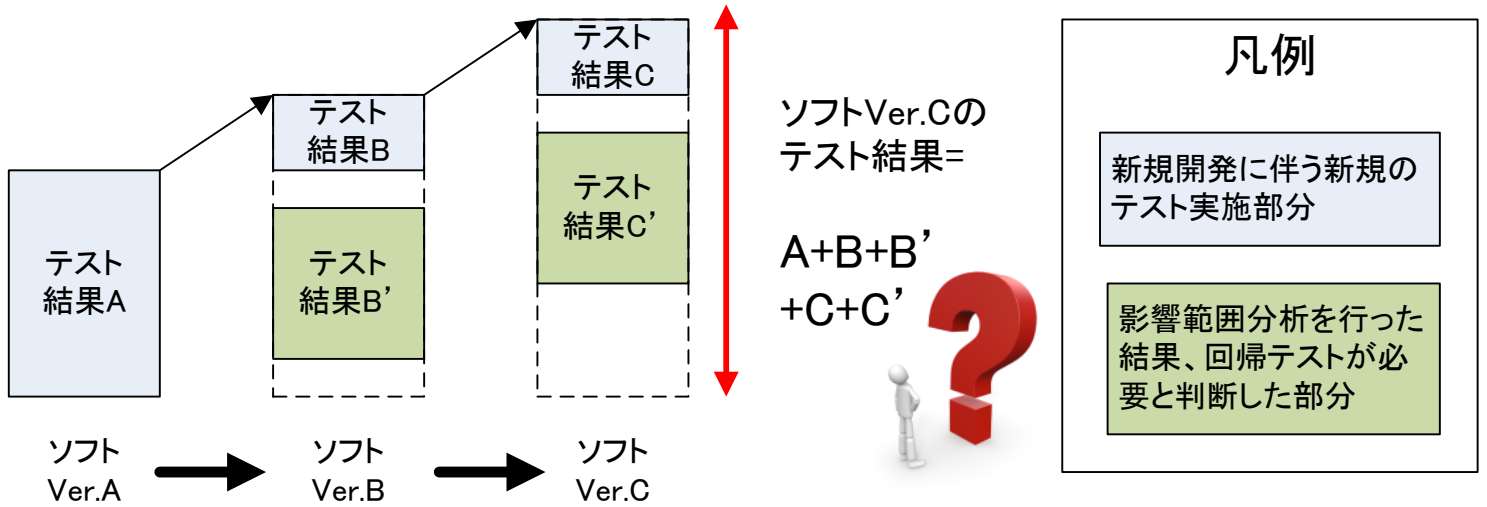


増加するテスト規模に対し要員増強で対応。しかし…

2-3. 従来型の問題点

ソフトウェア規模の増加に対応できない

- ソフトのバージョンアップにテストが追いつかない



- 自然言語を介することによる解釈誤り、ケアレスミスの発生
- テスト実施にもノウハウが必要（暗黙知の増加）

テスト規模の増加に伴い、手動実施の限界が見えてきた

2-4. テスト自動化に対する障害

組み込みソフト特有の事情も考慮しなければならない

- テスト実施のため複数の機器を必要とする

電源装置

RAMモニタ

オシロスコープ

- テスト自動化にはスクリプトがつきもの

- C言語は１００%の技術者が理解できる
- しかし、それ以外の言語は習熟度がバラバラ
- 新規言語に対する抵抗感もある

多くの技術者にとって、新規言語・スクリプトは**ハードルが高い**

3. 改善策

アイシン・ソフトウェア株式会社

3-1. ターゲット選定(1)

対象範囲を絞った上で、足元から着実に進める戦略

- 既存のテスト仕様書の分析から始めた

以下の条件に合うテストから自動化したい

① 複数のプロジェクトに適用できる
② **最小限の**機器構成で実現できる
③ 合否判断の条件が明確に決まる

- 統合開発環境(IDE)のデバッガをテスト仕様書に従って操作するタイプのテストを自動化対象として選定した

統合開発環境の**自動運転**と、動作結果の**自動判定**がポイントとなる

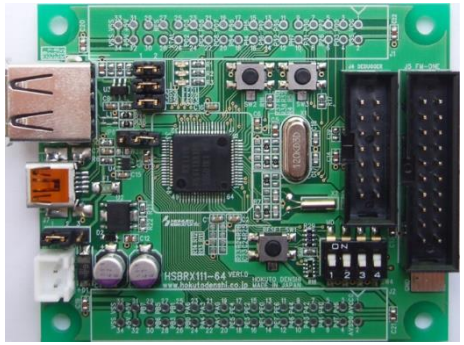
3-2. ターゲット選定(2)

ターゲット統合開発環境の選定

- ルネサス製CS+をターゲットに選定

画像はルネサス　エレクトロニクスWebサイトより引用
https://www.renesas.com/cs+

- ルネサス製の複数のマイコンに対応した統合開発環境



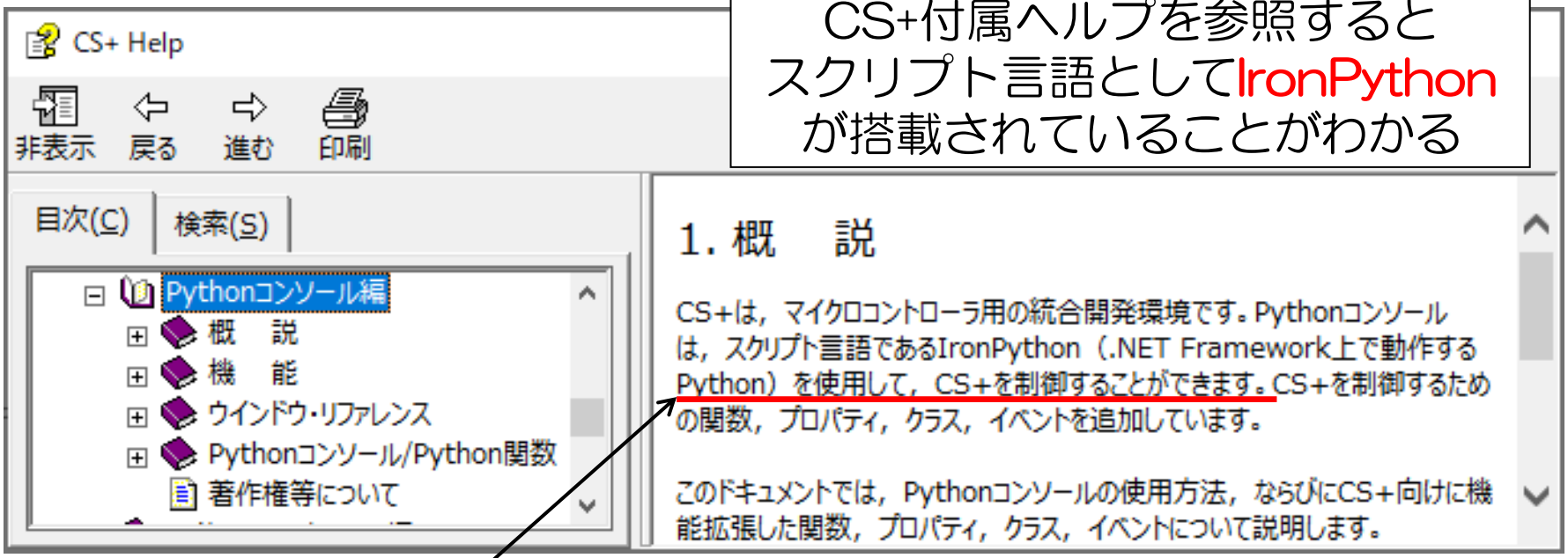
CS+に対応したマイコンボードの一例：
(株)北斗電子 HSBRX111-64 (RXマイコン搭載)

画像は北斗電子Webサイトより引用
https://www.hokutodenshi.co.jp/7/HSBRX111_64.htm

3-3. ターゲット選定(3)

ターゲット言語の選定

- CS+に搭載されているスクリプト言語を調査した



CS+付属ヘルプを参照すると
スクリプト言語としてIronPython
が搭載されていることがわかる

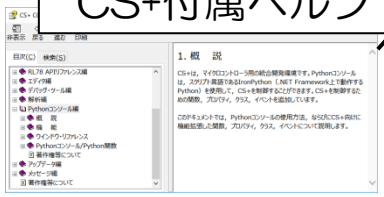
CS+の制御ができることから、Pythonをターゲットに選定した

3-4. 技術ノウハウの習得と蓄積

実験を繰り返し、ノウハウを蓄積

通称「オレンジ本」
Python言語解説の定番書

CS+付属ヘルプ



デバッガ操作ライブラリ

```
#
# 関数:リターン・アウト
#
def program_return_out():
    output_log("リターン・アウト実行")
    debugger.ReturnOut()

    # ジャンプ
    debugger.Jump.Address(JumpType.Source, debugger.GetPC())

#
# 関数:CPUリセット
```

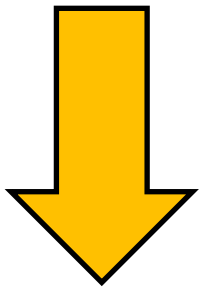
CS+で実験

Python言語でデバッガ操作ライブラリ関数群を作成した

3-5. 壁を打ち破るためのブレークスルー

実現可能性は見えてきたが、ここで大きな課題が…

Python言語でスクリプトを記述するには、テスト設計者がPython言語をマスターしていることが前提になる。しかし、**それは現実的なのか**？



新たなプログラミングスキルを習得することなく、使えるものとしたい…

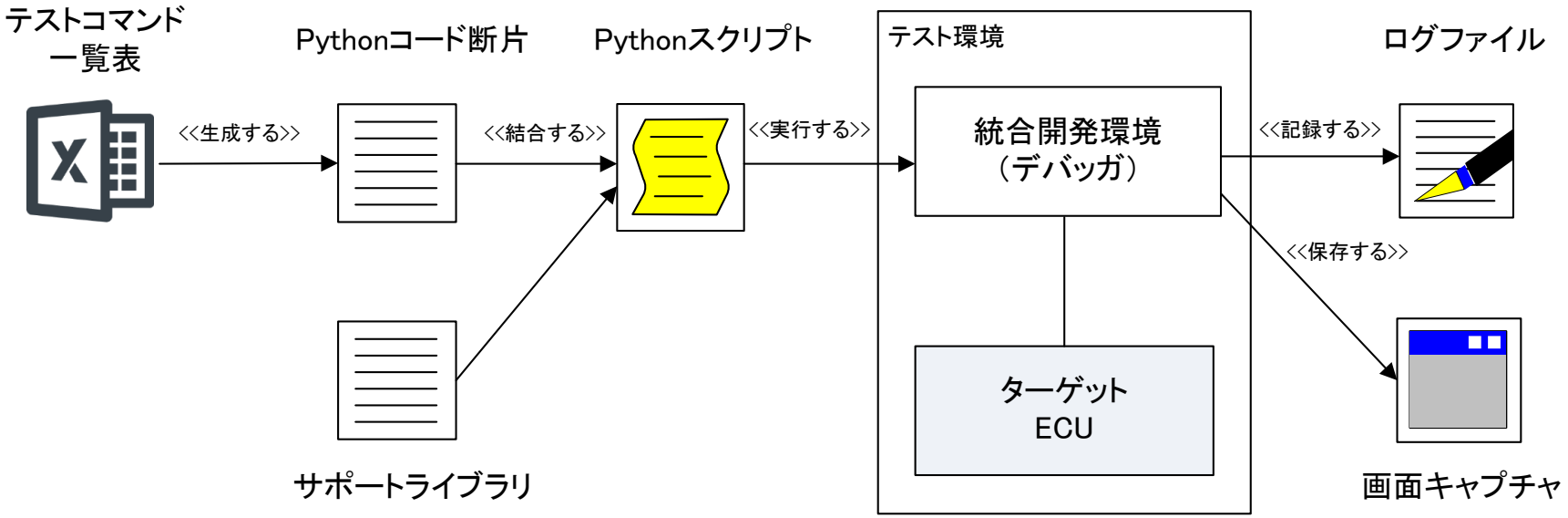
Excelのコマンド選択肢から、Pythonスクリプトを自動的にジェネレートする仕組みが出来れば、問題は**解決する**！

「コマンド選択肢から選ばせるだけ」に発想を転換した

3-6. Excel to Pythonのアイデア

誰でも簡単に、テスト自動化スクリプトを設計・作成できる

- Excelのテストコマンド一覧表から、デバッガ自動運転用のPythonスクリプトを生成
- 自動運転時にログファイルと画面キャプチャを保存

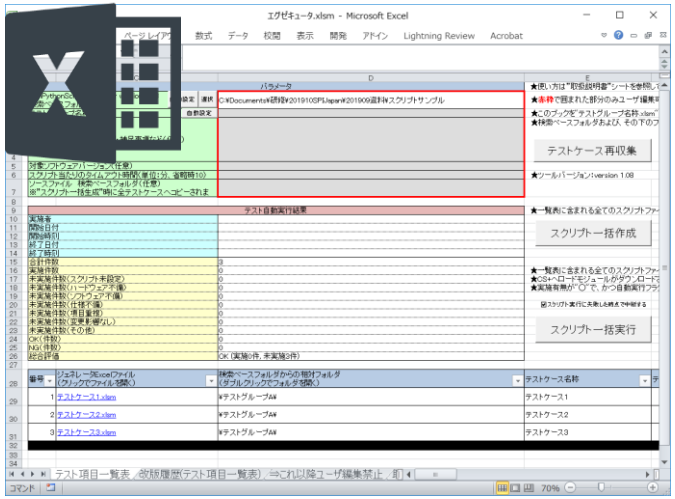


3-7. 無人連続運転の実現

複数のスクリプトを無人(自動)で逐次実施

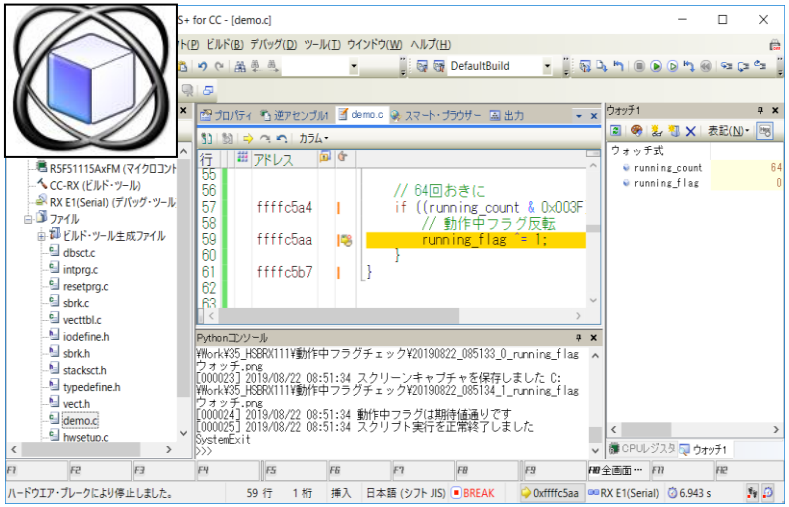
- テスト自動化には、無人(自動)でスクリプトを逐次実施する仕組みと、合否判定結果を自動集計する仕組みが不可欠

Excel



VBAから
自動運転

CS+



過去に習得したWindows APIの知識を活用して
ウィンドウ検索・擬似キー入力・クリップボード操作等をVBAで実装

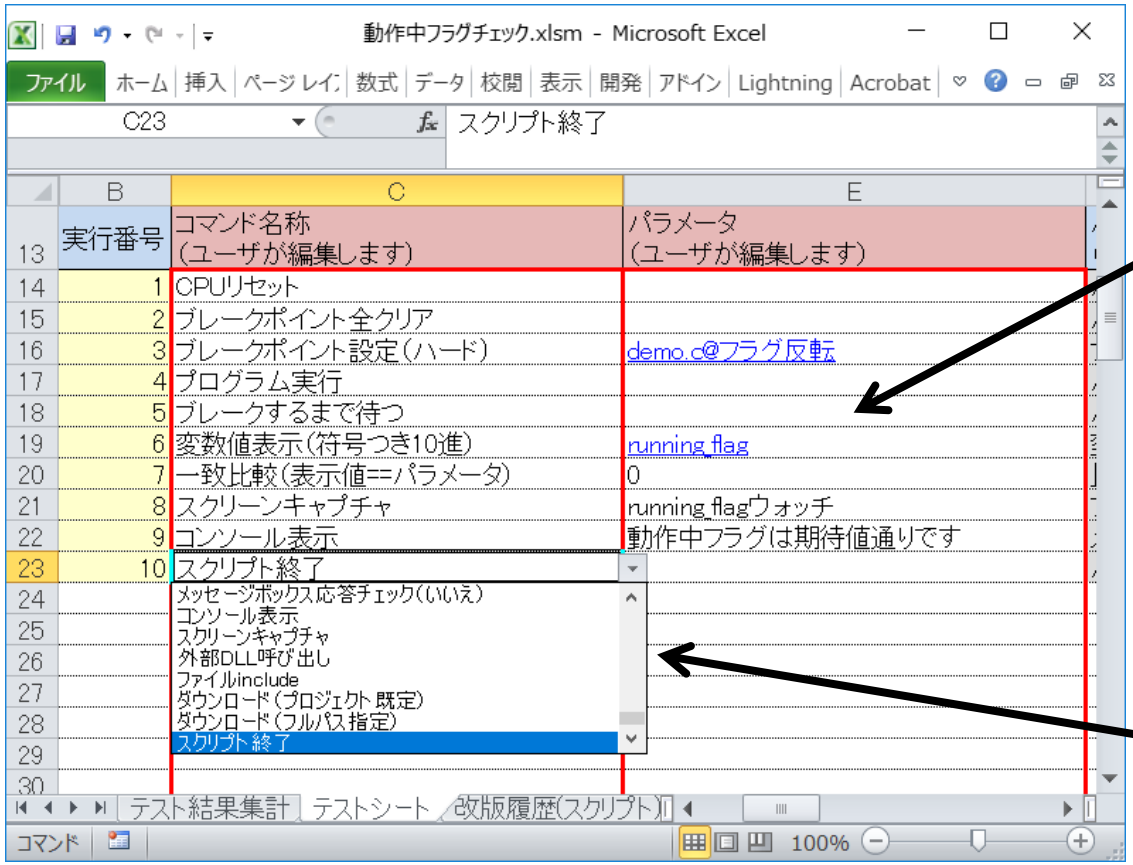
4. テスト自動化支援ツール

アイシン・ソフトウェア株式会社

4-1. スクリプトジェネレータ(スクリプト生成ツール)

コマンド選択＋パラメータ記入の繰り返しで設計を進める

- オペレータの操作手順を、そのまま「**なぞる**」イメージ



2．パラメータを記入
（以下は一例）

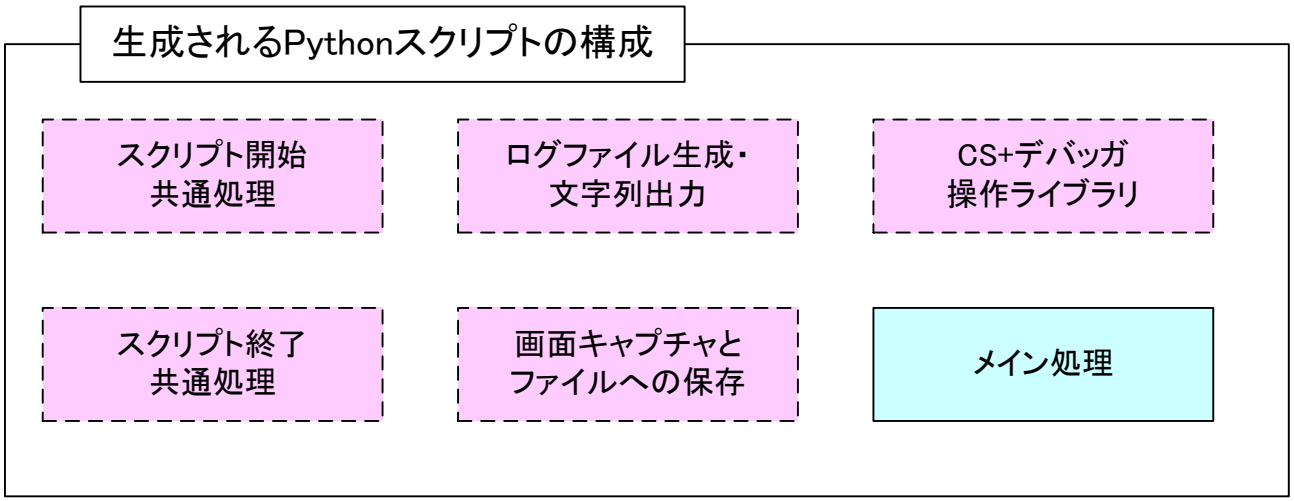
- ブレークポイント位置
- 評価対象の変数名
- スクリーンキャプチャのファイル名
- コンソール表示文字列

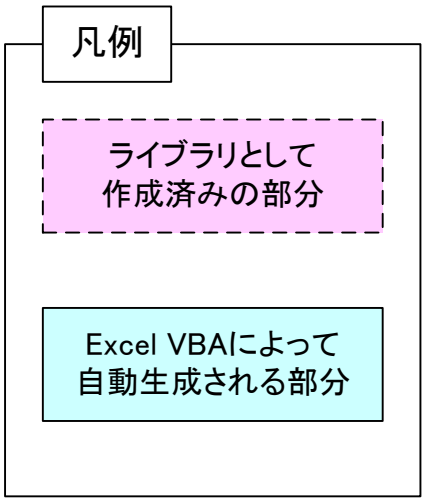
1．ドロップダウンリストからコマンドを選択

4-2. Pythonスクリプトの構成

Excel VBAはコマンド呼び出し部を自動生成するだけ

- 「スクリプト生成」コマンドで*.pyファイルが生成される



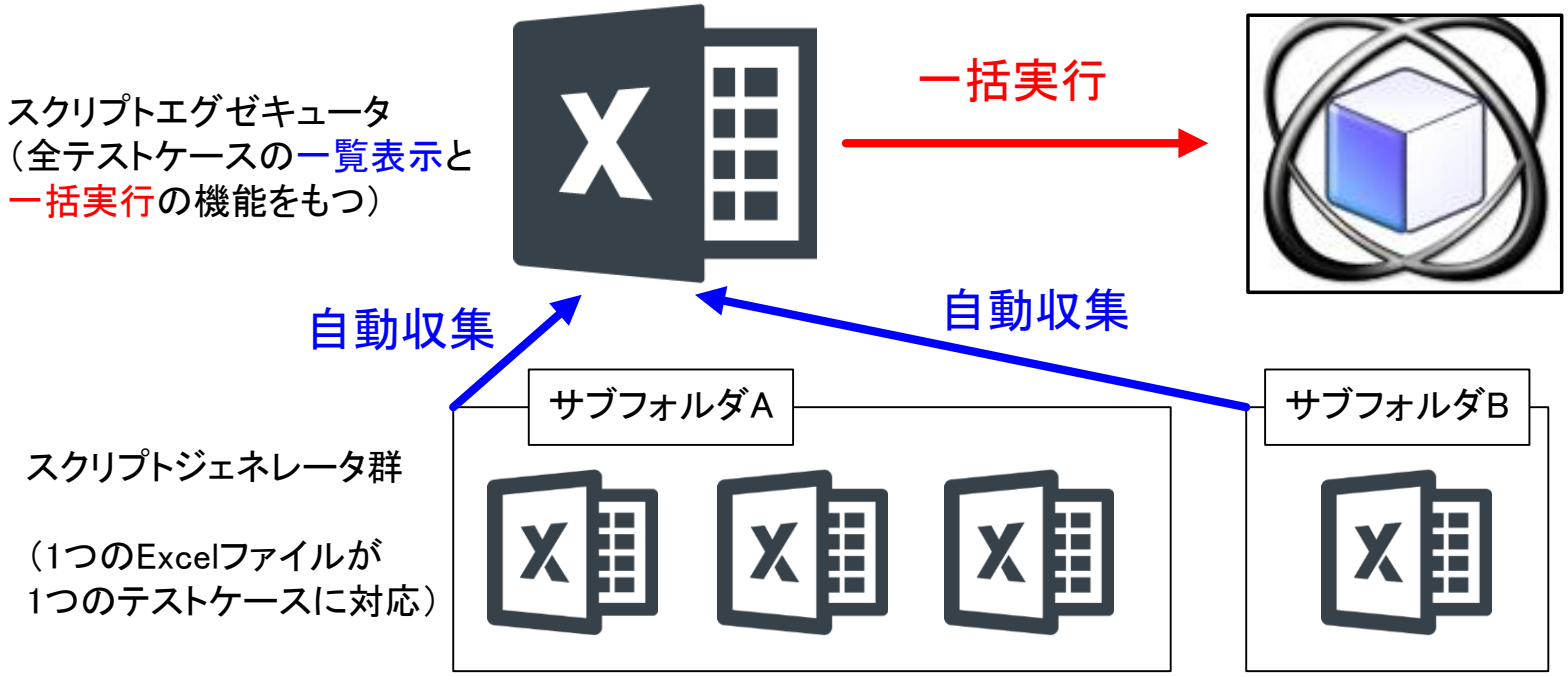


大半の処理はサポートライブラリとして実装済み(点線部分)
コマンド＋パラメータをPython言語に変換し、メイン処理とする

4-3. スクリプトエグゼキュータ(一括実行ツール)

複数のスクリプトをハンドリング・自動運転

- 階層下のフォルダを再帰的に検索しジェネレータファイルを自動収集（存在するファイルをリストアップ）
- 収集済みファイルに対し、スクリプト一括作成／一括実行



4-4. 不十分となってしまったところも

試作品ゆえの制約もある

- コマンドを沢山作ることが精一杯で、文書化は二の次

まずは「使いものになる」レベルを目指し、
ある程度、受け入れられるようになったら
その時に、詳しいドキュメントを書こう

- 一通りのデバッグはしたが、十分な品質とは言い難い

特定のボードで一通りテストしたが、
様々な動作環境／ユース・ケースで期待通り
動作するか否かは、未知数の要素もある

試作品ならでは制約事項と割り切り「**見切り発車**」した

5. 導入／展開

AISIN

アイシン・ソフトウェア株式会社

5-1. STEP1 ～先行導入～

せっかく作っても、使ってもらえなければ意味がない

- 技術説明会を開催し、新しもの好きそうな技術者に声かけ

こういうものを作ってみたんだ
試しに使ってみないか？

発表者

テスト
担当者

なるほど、面白そうですね

- テスト設計例のサンプルを提供

「現場の嬉しさ」をアピールすることで、現場の共感を得た

5-2. STEP2 ～試行プロジェクトでの導入～

実業務で使えるか否か、そこが分かれ目

- 顧客の説得と同意の取り付け

回帰テストのメリットを強調して粘り強く説得することで
顧客同意の取り付けに成功！

5-3. STEP3 ～適用範囲の拡大～

ファンを増やしていくことが近道

- 社内の改善事例発表会で取り組みを説明した

当チームの改善事例を発表します。
今回はテストの自動化に取り組み…

発表者

聴衆

- 今のプロジェクトにも適用できる？
- 他の統合開発環境で使えるかな？
- 今度レクチャーに来て欲しい！

多くの人に「こういう物があるんだ」と思ってもらうことが必要

5-4. フィードバックと、それに対応する改善

開発時には気付かなかった問題点も見えてきた

- ユーザーが拡がれば、それに伴って要望も増える

新規ユーザー

- 浮動小数点が扱えないと困る
- 列挙型も使えるようにしてよ
- スクリプトのincludeを行いたい
- 特定の環境で上手く動かない！

等々…

ツールのバージョンアップで対応した

5-5. 異なる開発環境への応用

基本となる考え方は同じ

- 「スクリプト」がキーワードとなるものは**何でも応用可能**

同様の発想で言語を意識することなく
スクリプトを作れる！

- 統合開発環境A （Tcl/Tk言語）
- 統合開発環境B （独自言語）
- SILS環境 （Python言語）
- HILS環境 （独自言語）

SILS: Software In the Loop Simulator
（ソフトウェアシミュレーションによるECUテスト）
HILS: Hardware In the Loop Simulator
（ハードウェアシミュレーションによるECUテスト）

適用しやすい環境から順次、同様のシステム構築を図った

6. 導入に対する効果

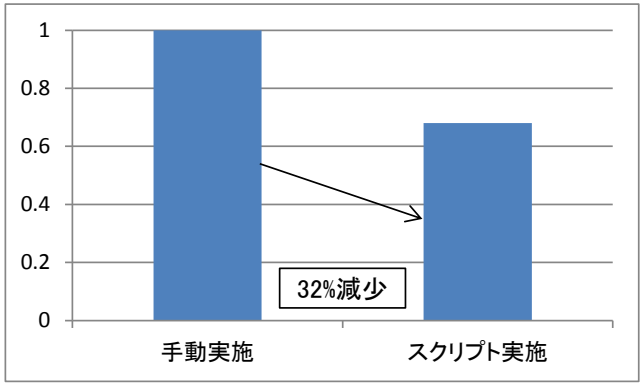
AISIN

アイシン・ソフトウェア株式会社

6-1. 改善による効果(1)

工数削減効果

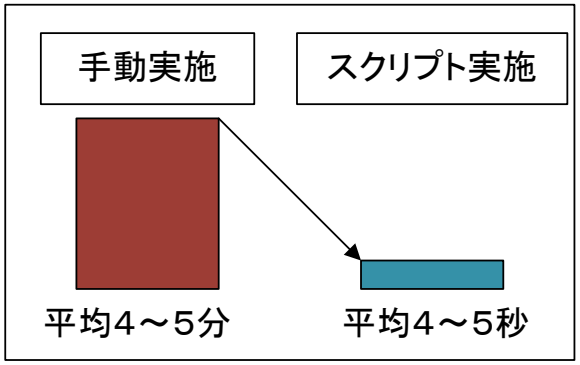
- スクリプト言語の習得は不要だが、それでも慣れが必要



手動実施とスクリプト実施の双方を行ったプロジェクトは無い。しかし、

「1テストケースあたりの所要時間」で比較した結果では、**スクリプト実施の方が有利**とのデータ有り

- 回帰テストに対する工数削減効果は圧倒的



- 自動運転により、テスト実施効率は**約60倍**に向上した
- 延べ2000件のスクリプトを8時間で一括実行したケース有り

6-2. 改善による効果(2)

テストケースの資産化

- 従来のテスト仕様書はテストケースの再利用に難があった

数MB～数10MBに及ぶExcelファイルから、
目的のテストを探してコピー＆ペースト…
なんだか面倒だなぁ

- スクリプトジェネレータはテストケースの再利用促進に寄与

- Excelファイルをフォルダ単位でコピー
- いらないものを消して
- エグゼキュータで一括リストアップ

1Excelファイル ＝ 1テストケースの仕組みによるメリットが出た

6-3. 改善による効果(3)

テスト設計に対するモチベーション向上

- テスト設計にプログラミング的・パズル的要素がある

テスト設計者

- テスト設計がコマンドの組み合わせになるため、あたかも**パズルを解いている**かのような楽しさがある
- 自分が設計したテストをすぐに実行でき、ログを見ることで動作状況を把握できる

- 従来の手法より、ジェネレータによるスクリプト設計の方が面白さ・やりがいを感じるとの声が挙がった

導入前に予想していなかった、モチベーションに対する効果があった

7. 振り返り／今後の課題

AISIN

アイシン・ソフトウェア株式会社

7-1. 振り返り

兎にも角にも、テスト自動化への足がかりを作れた

- 手作りの「**からくり**」による仕掛けで実現した

外部調達コスト ＝ **ゼロ**を達成！
（すべて手作り）

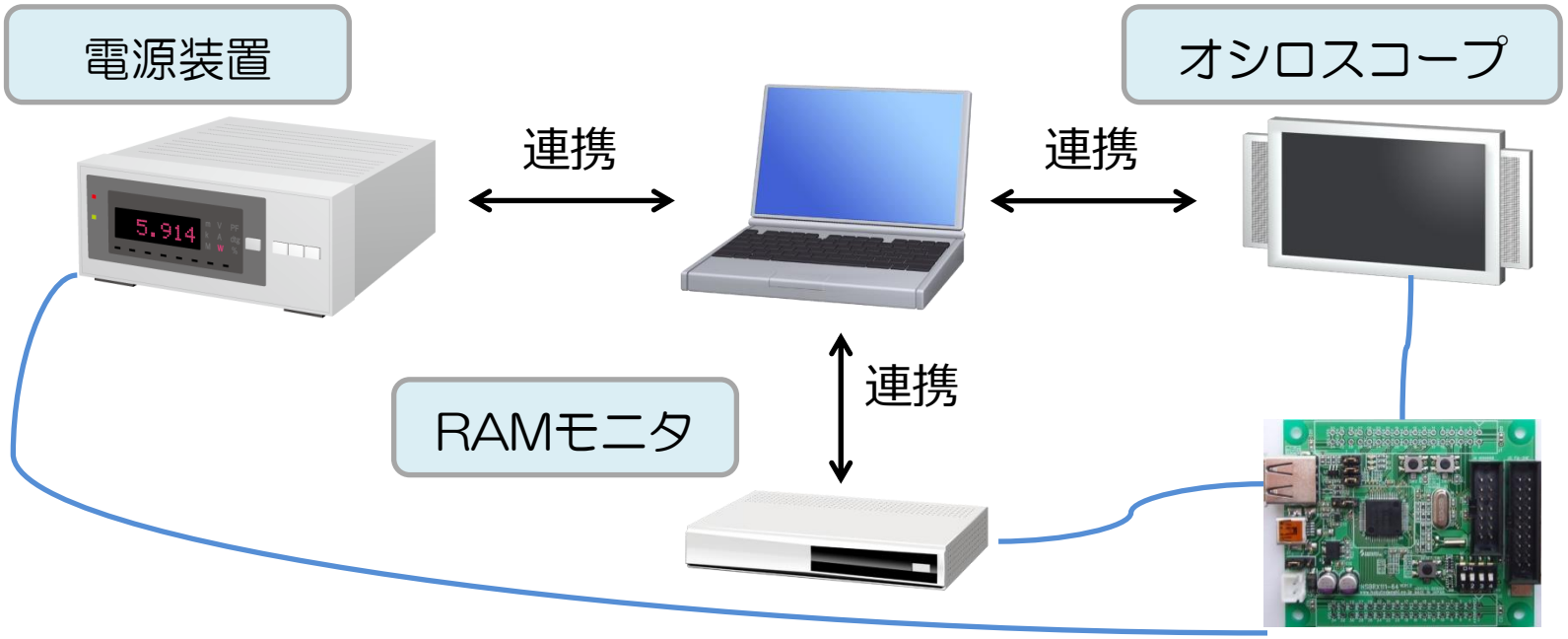
- テスト自動化マインドの醸成ができた

これからはテスト自動化の時代だ！
出来るものは何でも自動化しよう！

7-2. 今後の課題(1)

外部機器との連携強化

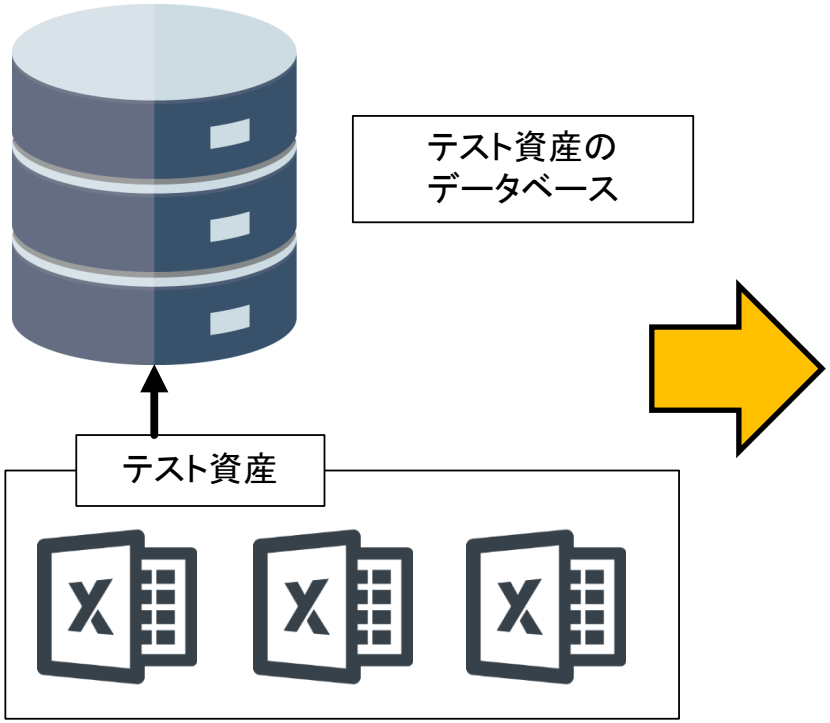
- 電源装置/RAMモニタ/オシロスコープとの連携を強化したい
  (ベンダ提供SDKの利用 or
  Windowsアプリの自動運転で実現)



7-3. 今後の課題(2)

テストケースの資産化

- テストケースをデータベース化し、再利用を促進したい



＜期待される効果＞

派生開発や流用開発の際に、従来のテストケースを参考または転用でき、QCDの向上が期待できる

＜課題＞

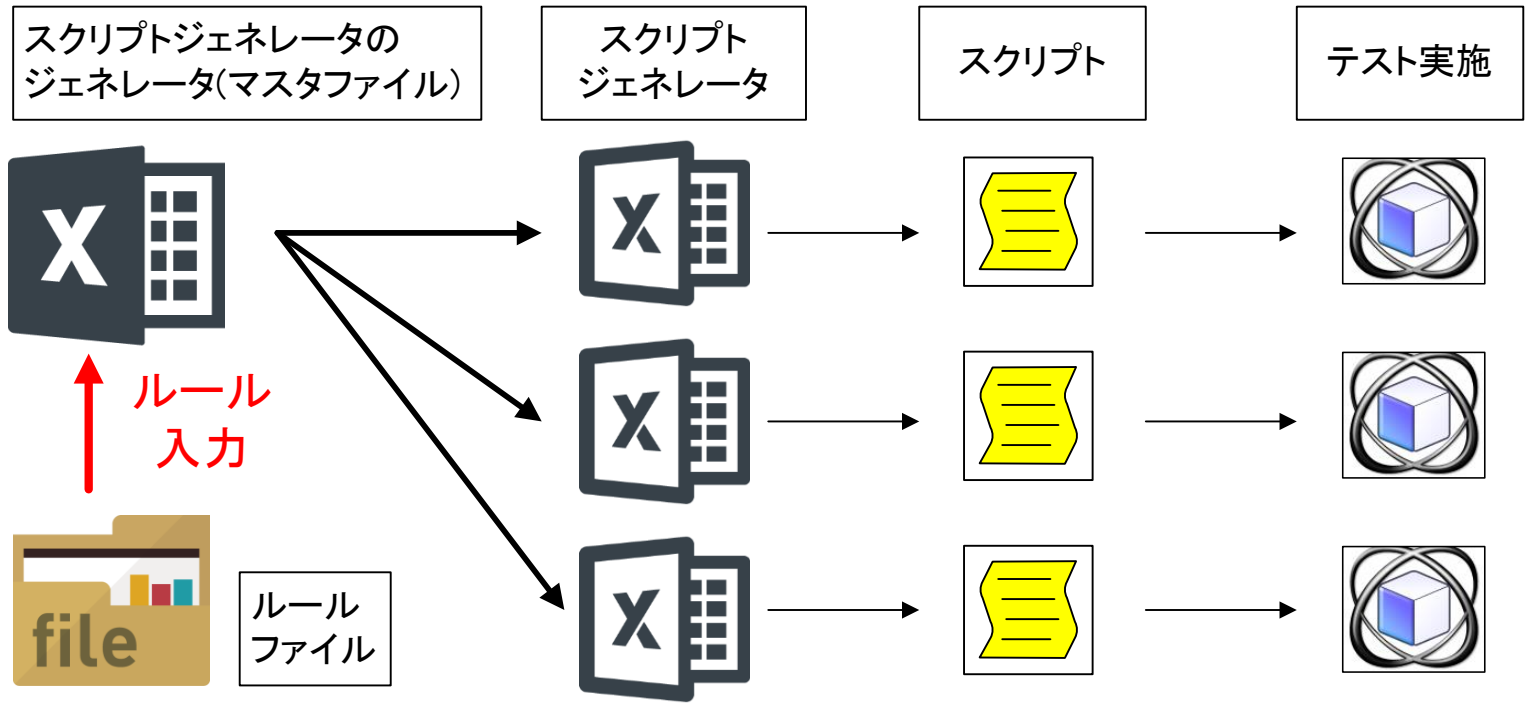
誰が、いつ、どのようにして、データベースを設計／管理／運用していくのか？

さらに一段レベルの高いテスト資産管理・システムが必要となる

7-4. 今後の課題(3)

バリエーション違いテストの自動化

- ルールに従ってテストケースを生成するシステムの開発
（ジェネレータをジェネレートする）



ご清聴ありがとうございました