テスト設計コンテスト’21

テス豆のテスト設計のご提案

Version: 2.0

チーム名：テス豆

目次

はじめに（１）

背景

QFユーザからの問い合わせが増加に伴い、QF開発チームは機能アップデートしてリリース頻度を増加する必要がある。

頻繁に機能アップデートすると、「テスト範囲」や「消化すべきテスト回数」が増える。これにより、テストにかけるべき時間が増える。

手動テストのみではテストできる時間は限界があるので、本来必要なテストにかける時間が確保できず、十分なテストができなくなる。

「十分なテストができない」とQFに埋め込まれた欠陥の除去が十分にできずに、品質が悪い状態でQFをリリースしてしまう。

この結果、QF開発チームの開発効率の低下するため、頻繁なアップデートができなくなる可能性がある。

はじめに（２）

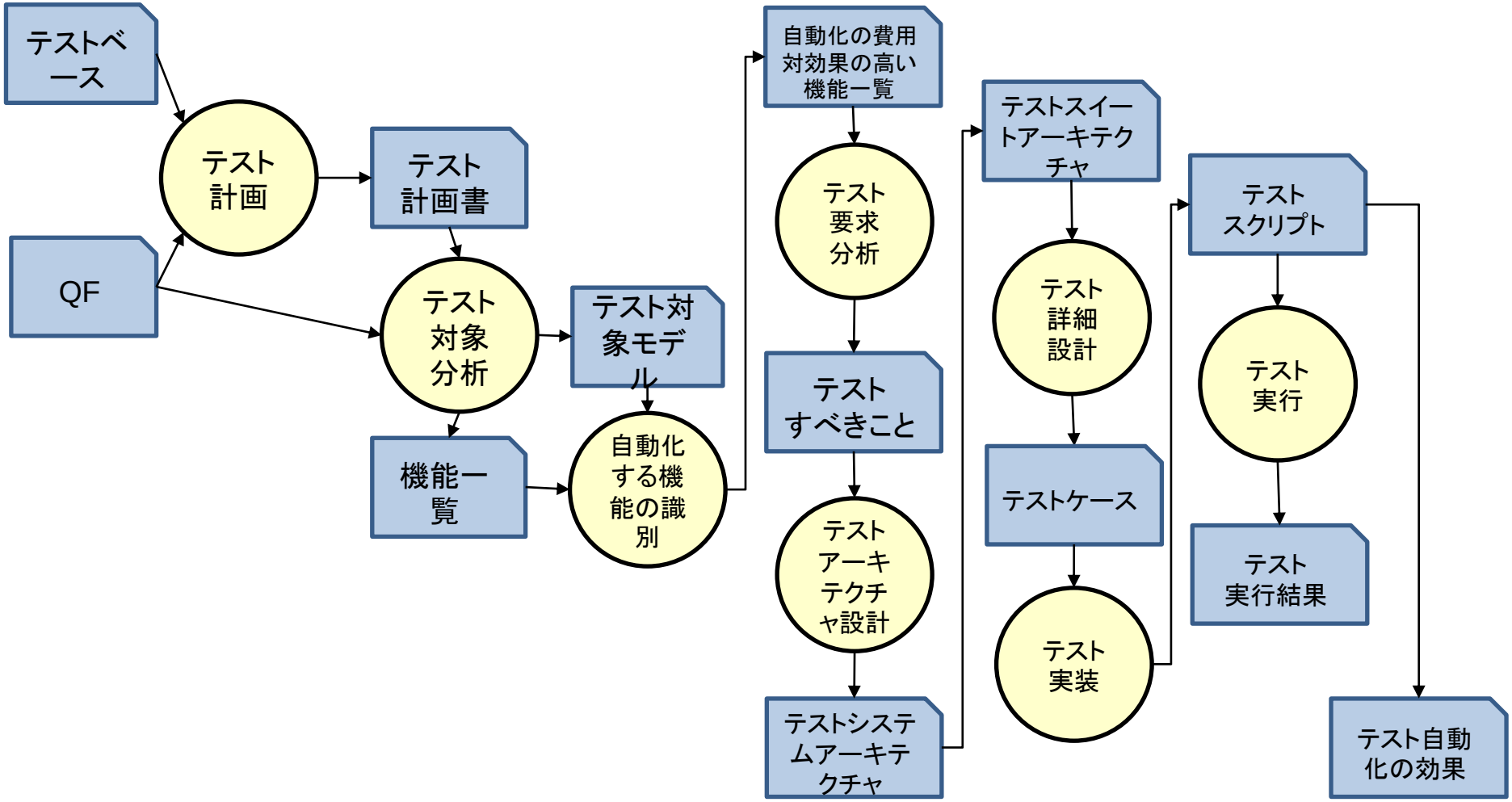
- 課題
  - 十分なテストをするために「テスト実装とテスト実行」にかかる「時間と金額」を削減できるようにテスト自動化する必要がある
  - 殺虫剤のパラドクス等も考慮してテストケースやテストスクリプト等は定期的に修正する必要がある
- 提案
  - 手動テストと自動テストのテスト活動の時間を比較して、自動化の費用対効果が良いテストを自動化する
  - 自動テストを動かすためにテスト環境の構成要素とそれらの関係を決める
  - 十分なテストができるように、テスト間の依存関係やテスト実行の効率が良くなるようにテスト実行の順序を決める
  - テストスクリプトの作成および修正にかかる時間を短くするためデータ駆動テストアプローチを適用する
- うれしいこと
  - テストにかかる時間を削減できる
  - 十分なテストができるようになる

はじめに（３）

- 費用対効果
  - 費用
    - テスト活動に関わる人数 × テスト活動に必要な時間 × 時給
      - テスト活動に必要な時間
        - テスト環境アーキテクチャ設計の時間
        - テスト環境の実装の時間
        - 自動テストスクリプト作成の時間
    - 設備費用
  - 効果
    - 自動化できた消化すべきテストケース数× {(1テストケースあたりの手動テスト実行時間 × 時給) - [1テストケースあたりの自動テスト実行時間 × (1時間あたりの電力消費費用 + 1時間あたりのサブスクリプション費用)]}

テストの全体像

- 以下のようなプロセスでテスト自動化を実現



テスト計画（１）

- 何を、どうやってテストするか

項目		詳細
目的・方針		QF開発チームの一員であるテス豆は、ASTER社様より「機能アップデートによるリリース頻度増加を見据えたテスト自動化」のご依頼を承りました。「費用対効果の高いテスト実行の自動化」をASTER社様に提案する。
スコープ		
	テストアイテム	クラウドテスト管理ツール　Quality Forward
	テスト対象機能	テスト対象モデルから導出した機能から、自動化の費用対効果の高い機能を選んだもの。
	非テスト対象機能	上記以外。手動テスト担当チーム（（仮）：Aチームに頼む）
アプローチ		
	テストレベル選定	システムテスト
	テストタイプ選定	機能テスト
	テストタイプ毎の目的	機能の合致性検証
	テストタイプごとのテスト対象	費用対効果の高い機能一覧
	テストアプローチ	該当機能の合致性検証のために必要な技法を選定し、必要なテストのパタンを洗い出す。

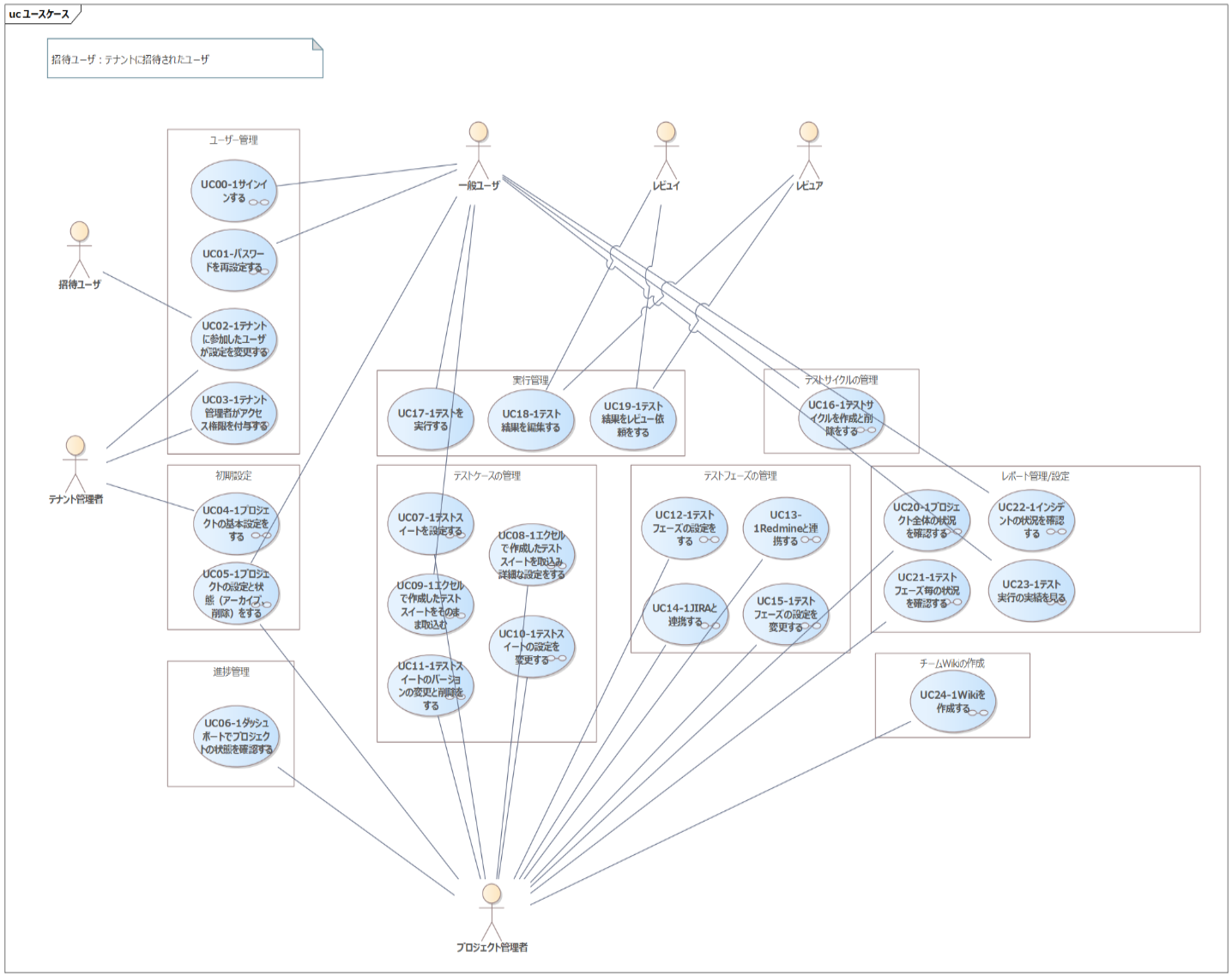
テスト計画（２）

- 何を、どうやってテストするか

マネジメント			
	テストタスクと担当	テス豆	
	テスト環境	シート名：テストシステムアーキテクチャを参照のこと	
	スケジュール	シート名：テストスイートアーキテクチャに定めるテストの順番を参照のこと	
	リスクとその対策		
		リスク	対策
		手を動かせるメンバが限られている。	費用対効果の高い機能一覧から、現メンバ（3人）で１Wで出来そうな機能を選定する。
		やみくもに自動化してもスクリプト保守コストがかかってしまう。	実装保守しやすいものを選定する。
		全て手動でやってしまうと、機能アップデートがかさむとテスト時間が膨大になってしまう。	何度もテストする見込みの物を選定する
		自動化ツールはどうすればいいのか。	無償で環境構築コストの低いものを利用する。
成果物	TBD	テスト設計書	
文書管理	TBD	ドキュメント体系書	

テスト要求分析（１）

- テスト対象モデルの作成



テスト要求分析（２）

- 自動化するテストの選定

自動化する基準	実装に手間がかからず、よく使われそうで、不具合がありそうなところを自動化する。

- 実装に手間がかからないとは
  - 入力する項目や遷移を数え、それらが少ないこと
- よく使われそうとは
  - マニュアルを見なくても操作できて、手も止まらないほどさわれること。
- 不具合がありそうとは
  - ひと通りさわってみて、テスト管理という仕事の阻害につながりそうなこと。

テスト要求分析（３）

実装に手間がかからず、よく使われそうで、不具合がありそうなところに絞って、自動化する

- 自動化するテスト

	実装想定工数											何度もテストされるか				費用対効果の高さ 数値が低いほど良い
	必須入力数 多いほど工数がかかる								画面遷移数 多いほど工数がかかる	合計	備考	よくつかわれそうか(1－3点) 低いほどよく使われそう	理由	不具合出そうか(1－3点) 低いほど不具合が多そう	理由	
	フォーム	チェックボックス	プルダウン	ファイルアップロード	オートフィルドラッグ	フィルタ	コピペ	クリック								
テナント使用量表示機能								1	1	2		3	今回は対象外。何度も	3	今回は対象外。	8
プロフィール設定機能	2			1				2	2	7		3		3		13
ユーザ招待機能	5	2	1					5	3	16		3		3		22
ユーザ削除機能								3	2	5		3	今回は対象外。何度も	3	今回は対象外。	11
ユーザ権限変更機能			1					3	3	7		3		3		13
テナント切り替え機能								2	2	4		3	今回は対象外。何度も	3	今回は対象外。	10
[illegible]																
サインイン機能	2							1	1	4		1	毎回使うため	3		8
プロジェクト追加機能	2	1						1	2	6		1	プロジェクトは次々作	3		10
プロジェクトへのユーザ追加機能		2						3	3	8		2		3		13
プロジェクト設定変更機能	3							4	4	11		1		2		14
カバレッジパネル表示機能		1		1				2	2	6		1	QFで何したいかと考え	1	表示はユーザの	8
テストスイート作成機能	2	1	1					2	2	8		1		3		12
[illegible] 機能		3	1	1				6	4	15		1		1		17
テストスイートダウンロード機能								2	2	4	対象外。ダウンロード	2		3		9
テストケース取込機能				1				3	3	7	対象外。取込用Excelを	1		1		9
テストケース編集機能	1		1		2	1	1	1	1	8	対象外。編集工数が数	1		1		10
テストスイート設定機能	2	1	2					3	2	10		3		3		16
テストスイートの新バージョン設定機能	1		2					2	2	7		3		3		13
[illegible]																
テストフェーズ作成機能	1							3	3	7		1		3		11
[illegible]																
テストサイクル作成機能	2							3	3	8		1		3		12
[illegible]	2							4	4	10		1		3		14
レビュー機能	3							3	3	9		1		1		11
テスト実行機能			1					3	3	7		1		2		10
レポート表示機能								4	4	8		1		1		10
[illegible] WIKI機能	2							4	4	10		1		3		14
BTS連携機能		3	1	1				6	4	15		1		1		17

テスト要求分析（４）

- テスト観点、テストアプローチの特定

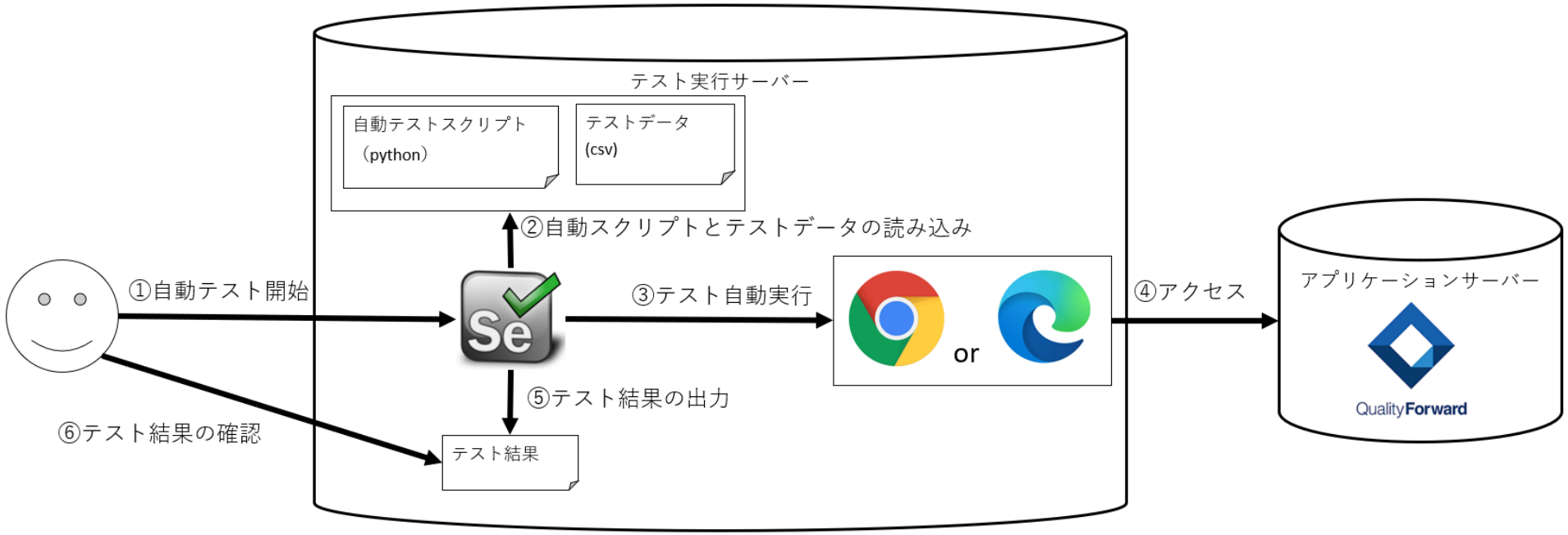
	テスト観点	テストアプローチ	テスト技法	網羅基準
サインイン機能	メールアドレスとパスワードを入れてサインインできること	ディシジョンテーブルをつかって、メルアド/パスワードのパタンを洗い出しておいて、ユースケースの振る舞いパタンを洗い出す。	ユースケーステスト	例外/代替処理網羅
プロジェクト追加機能	プロジェクト名をいれてダッシュボード画面が表示されること	ディシジョンテーブルをつかって、プロジェクト情報入力パタンを洗い出しておいて、ユースケースの振る舞いパタンを洗い出す。	ユースケーステスト	例外/代替処理網羅
カバレッジパネル表示機能	テストスイートの設定で指定した項目に対して、カバレッジパネルから、カバレッジを確認できる。	事前に、テストスイートで項目1の欄に複数のテストスイートをつくっておき、ユースケースの振る舞いパタンを洗い出す。	ユースケーステスト	例外/代替処理網羅
テストスイート作成機能	テストスイートメニューを開いて、テストスイートを名を入れて、カバレッジパネル集計ができて、テストブロッカー設定もできて、テストスイートが作成、削除までできる。	ディシジョンテーブルをつかって、チェックボックスの組み合わせを洗い出しておいて、ユースケースの振る舞いパタンを洗い出す。	ユースケーステスト	例外/代替処理網羅
テストフェーズ作成機能	テストスイート作成ボタンから、テストスイート名を入力して、作成、削除までできる。	事前に、テストスイートを作ってあるパタン、ないパタンを用意しておいて、ステータスを利用可にした、しない状態において、ユースケースの振る舞いパタンを洗い出す。	ユースケーステスト	例外/代替処理網羅
	スイート一覧をクリック後、サイクル一覧	事前にテストスイート、テストフェーズをつ		

※一部を抜粋。残りは、成果物1および、成果物2「テスト自動化機能選定とその効果」のシート名「テストすべきこと」にすべて列挙済

テストアーキテクチャ設計（１）

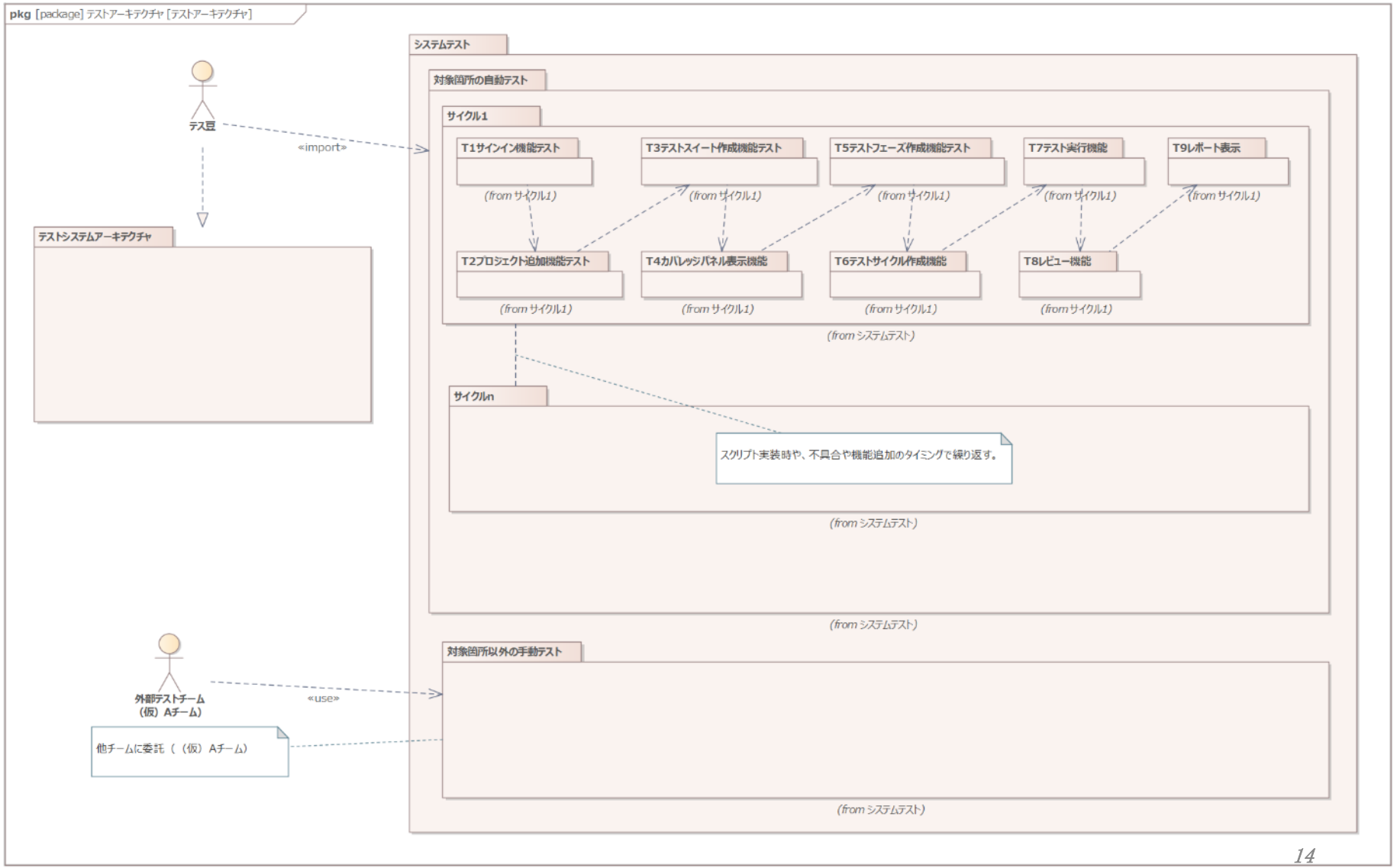
- テストシステムアーキテクチャ
  - メリデメ分析を経て、以下の仕組みで自動化を実現

候補		メリット	デメリット
無償ツール	Selenium（Java）	Java環境があればできる	Maven環境構築
無償ツール	Selenium（python）	python環境があればできる	現状なし。
無償ツール	その他	お金かからない。	学習コスト
有償ツール	その他	出来ることが多い	学習コスト



テストアーキテクチャ設計（２）

- テストスイートアーキテクチャ



テスト詳細設計

- テストアプローチにそって、テストケースの形に詳細化する。

		テスト観点	テストアプローチ	仕様項目	事前条件	入力	操作	期待結果
TC01	T1サインイン機能	メールアドレスとパスワードを入れてサインインできること	ディシジョンテーブルをつかって、メルアド/パスワードのパタンを洗い出しておいて、ユースケースの振る舞いパタンを洗い出す。	2.1ユーザサインイン	01_サインイン画面が開いている	02_異常メールアドレス 正常パスワード	03_サインインボタンを押す	04_サインイン画面のまま
TC02						05_正常メールアドレス 異常パスワード	06_サインインボタンを押す	07_サインイン画面のまま
TC03						08_異常メールアドレス 異常パスワード	09_サインインボタンを押す	10_サインイン画面のまま
TC04						11_正常メールアドレス 正常パスワード	12_サインインボタンを押す	13_プロジェクト一覧画面が表示される
TC05	T2プロジェクト追加機能	プロジェクト名をいれてダッシュボード画面が表示されること	ディシジョンテーブルをつかって、プロジェクト情報入力パタンを洗い出しておいて、ユースケースの振る舞いパタンを洗い出す。	3.1 新規プロジェクトを作成する	14_プロジェクト一覧画面が表示さ	―	15_プロジェクト追加ボタンを押す	16_プロジェクト作成画面に遷移す
TC06					17_プロジェクト作成画面が表示される	18_異常プロジェクト名 正常プロジェクト概要	19_登録ボタンを押す	20_プロジェクト作成画面のまま
TC07						21_正常プロジェクト名 異常プロジェクト概要	22_サインインボタンを押す	23_プロジェクト作成画面のまま
TC08						24_異常プロジェクト名 異常プロジェクト	25_サインインボタンを押す	26_プロジェクト作成画面のまま

※一部を抜粋。残りは、成果物1および、成果物2「テスト自動化機能選定とその効果」のシート名「テスト詳細設計」にすべて列挙済

テスト実装

- 自動テストスクリプトの作成や修正の時間を削減するため、データ駆動テストアプローチを適用
  - Seleniumの自動テストスクリプトとテストデータを分離
    - テストデータだけ変更が必要な場合は、自動テストスクリプトは変更せずに利用できる

テスト実装(サンプル)

- seleniumによる自動テスト
- csvからデータを取得するデータ駆動型テスト

```python
import unittest
import csv
from selenium import webdriver
from time import sleep
from ddt import ddt, data, unpack

LOGIN_FAIL_RESOURCE = "./resource/login_fail.csv"
LOGIN_URL = "https://aegis-contest.sg-apps.com/users/sign_in"

def get_data(file_name):
    # create an empty list to store rows
    rows = []
    # open the CSV file
    data_file = open(file_name, "r")
    # create a CSV Reader from CSV file
    reader = csv.reader(data_file)
    # skip the headers
    next(reader, None)
    # add rows from reader to list
    for row in reader:
        rows.append(row)
    return rows

@ddt
class SignIn(unittest.TestCase):
    def setUp(self):
        self.browser = webdriver.Chrome(executable_path='  omedriver.exe')
        self.driver = webdriver.Chrome('chromedriver.e  ')
        self.driver.get(LOGIN_URL)

    @data(*get_data(LOGIN_FAIL_RESOURCE))
    @unpack
    def test_sign_in_fail(self, user_email, password):
        elem_username = self.driver.find_element_by_id('user_email')
        elem_username.send_keys(user_email)
        elem_password = self.driver.find_element_by_id('user_password')
        elem_password.send_keys(password)
        elem_logn = self.driver.find_element_by_xpath("//input[@value='サインイン']")
        elem_logn.click()
        sleep(2)
        self.assertEqual(self.driver.current_url, LOGIN_URL)

    def tearDown(self):
        self.driver.quit()

if __name__ == '__main__':
    unittest.main()
```

ログイン失敗のテストケース

```
user_email,password
user@email,
,pass
user@email,pass
```

CSVの情報を元にテスト実行

テスト実行結果（１）

- 以下の問題に遭遇。不具合でなかったとしても、手間がかかり、よく使われやすい、不具合が出やすい機能にしぼって自動化して、機能追加や不具合確認で何度も生じるテストを次は短い時間でできる見込みがあることを示すことができた。

T9テストレポートの表示機能のテスト

問題点1	テストブロッカーの設定をしているが、チャートのテストブロッカー上位ランキングに表示されない
問題点2	自動テストスクリプトを複数回実行していると、サイクル一覧ボタンの表示不正がおきる

T6のテストサイクル作成機能のテスト

テスト実行結果（２）

- 問題点1:テストブロッカーの設定をしているが、チャートのテストブロッカー上位ランキングに表示されない

設定済

テストブロッカーの設定

☑ BLOCKの結果を集計する
☑ Q&Aの結果を集計する

設定済

テストブロッカーの集計に利用する項目設定

対象のカラム

項目1

BLOCKとなるテストケースがあるにもかかわらず

テスト実施日	テスト結果
2021/08/19	PASS
2021/08/19	PASS
2021/08/19	BLOCK
2021/08/19	Q&A

表示されない

テストブロッカーの上位10件

BLOCK	発生件数

ありません

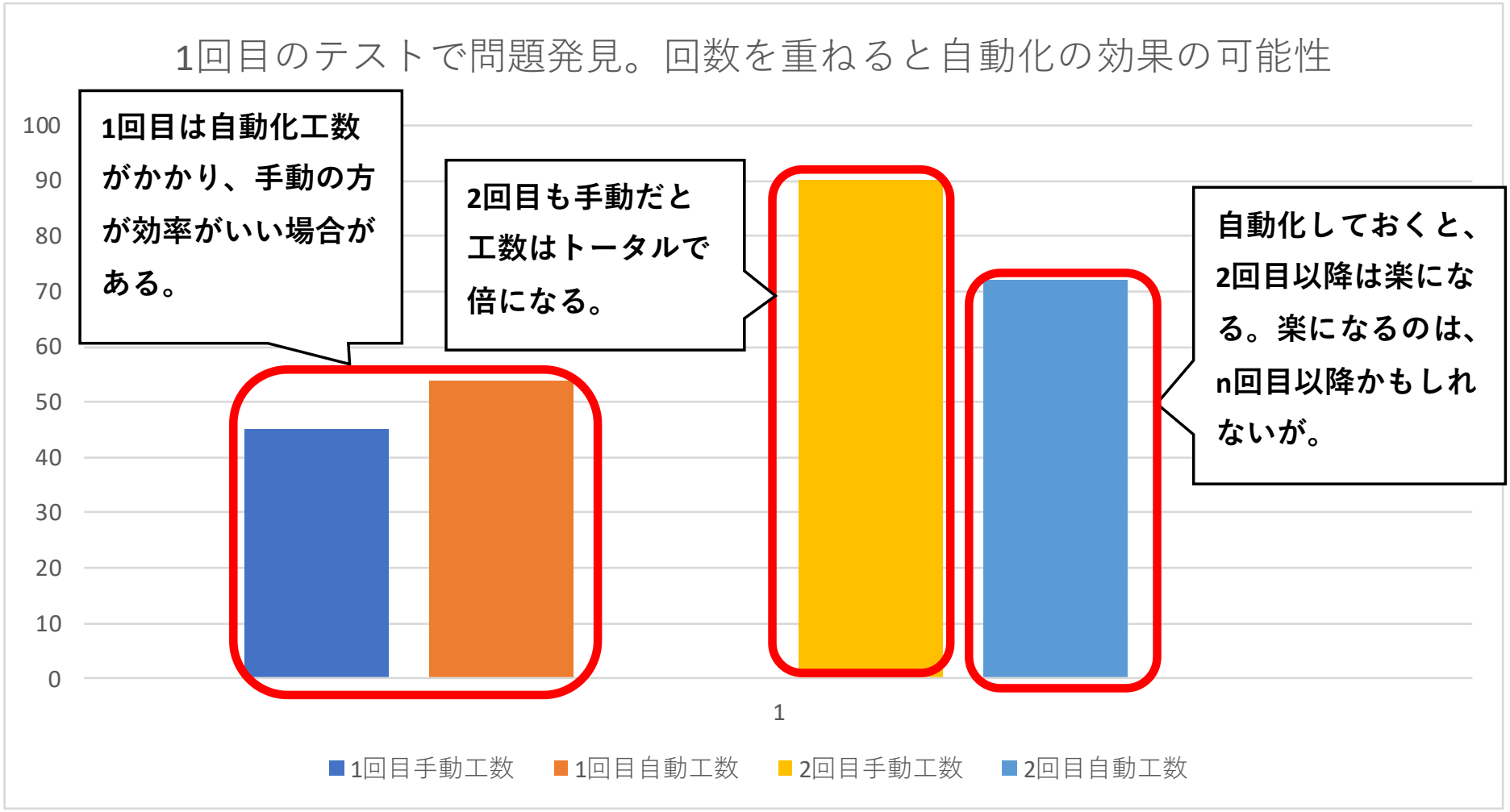
Q&A	発生件数

ありません

テスト実行結果（３）

- 問題点２：自動テストスクリプトを複数回実行していると、サイクル一覧ボタンの表示不正がおきる

自動化による効果



単位はひとまず5K￥/時間。数値については継続して見直していく。

テス豆
TESUMAME

発表は以上です。ご清聴ありがとうございました。

以降、付録（成果物1の図表）。