テスト設計チュートリアル テスコン編‘ 22

テスト設計コンテスト’21 OPENクラス決勝戦 成果物の解説

ソフトウェアテスト技術振興協会（ASTER）

@Satomi Ogusu

テストベース

Quality Forward ユーザーマニュアル

Quality Forward 提案書

2021年テスト設計コンテスト結果

順位	チーム名
優勝	テス豆
準優勝	シン・田町補充計画
3位	エムスリーQAチーム
4位	ジョゼ

成果物共有

テス豆

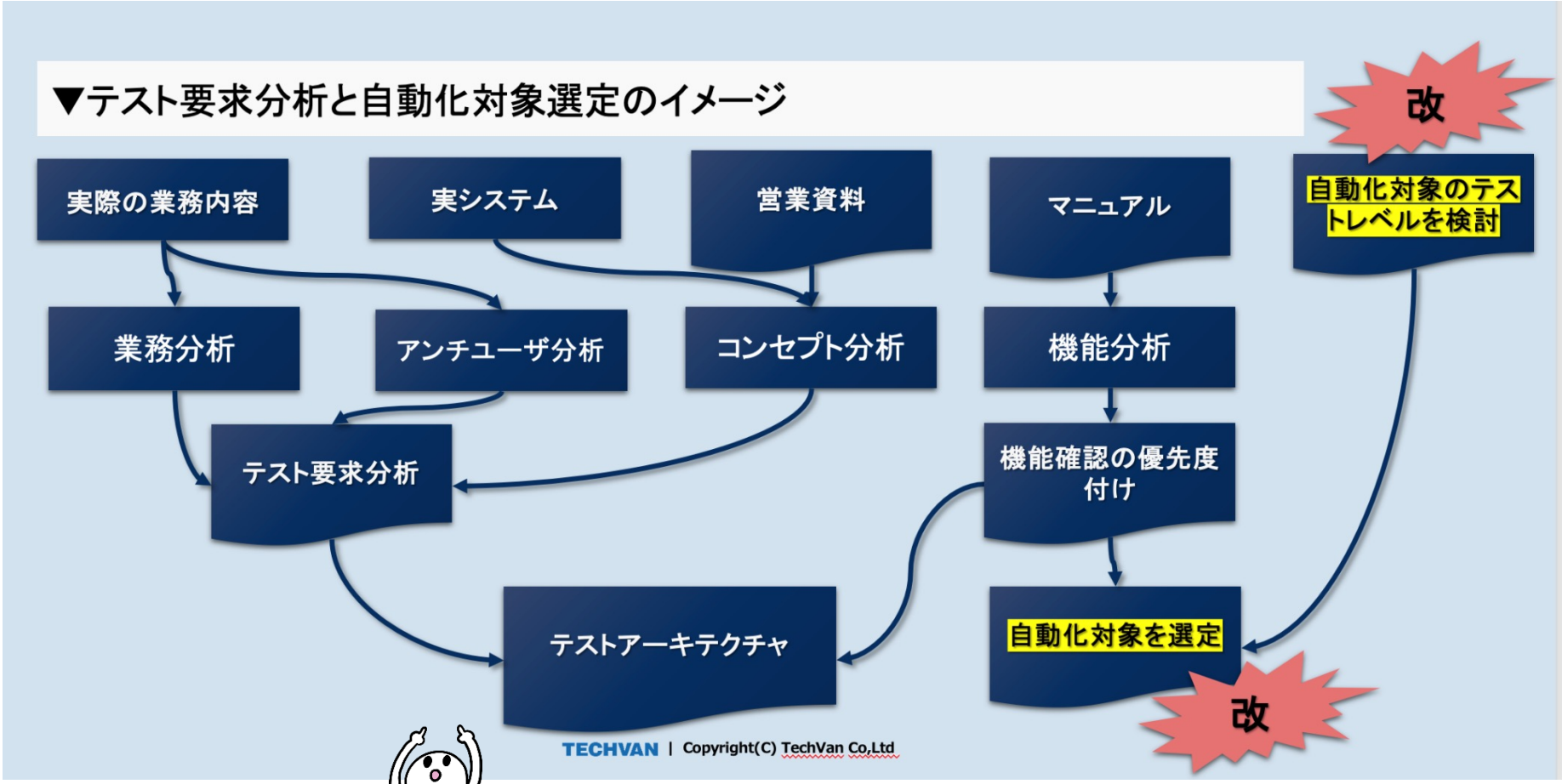
テスト計画書

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。	
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	

テスト全体の目的や方針、スコープや進め方がまとめられています。

これをもとに今後テストを行っていくことになります。後工程で変更が生じた場合はここもしっかり修正することで、ぶれないテスト設計ができるようになります。

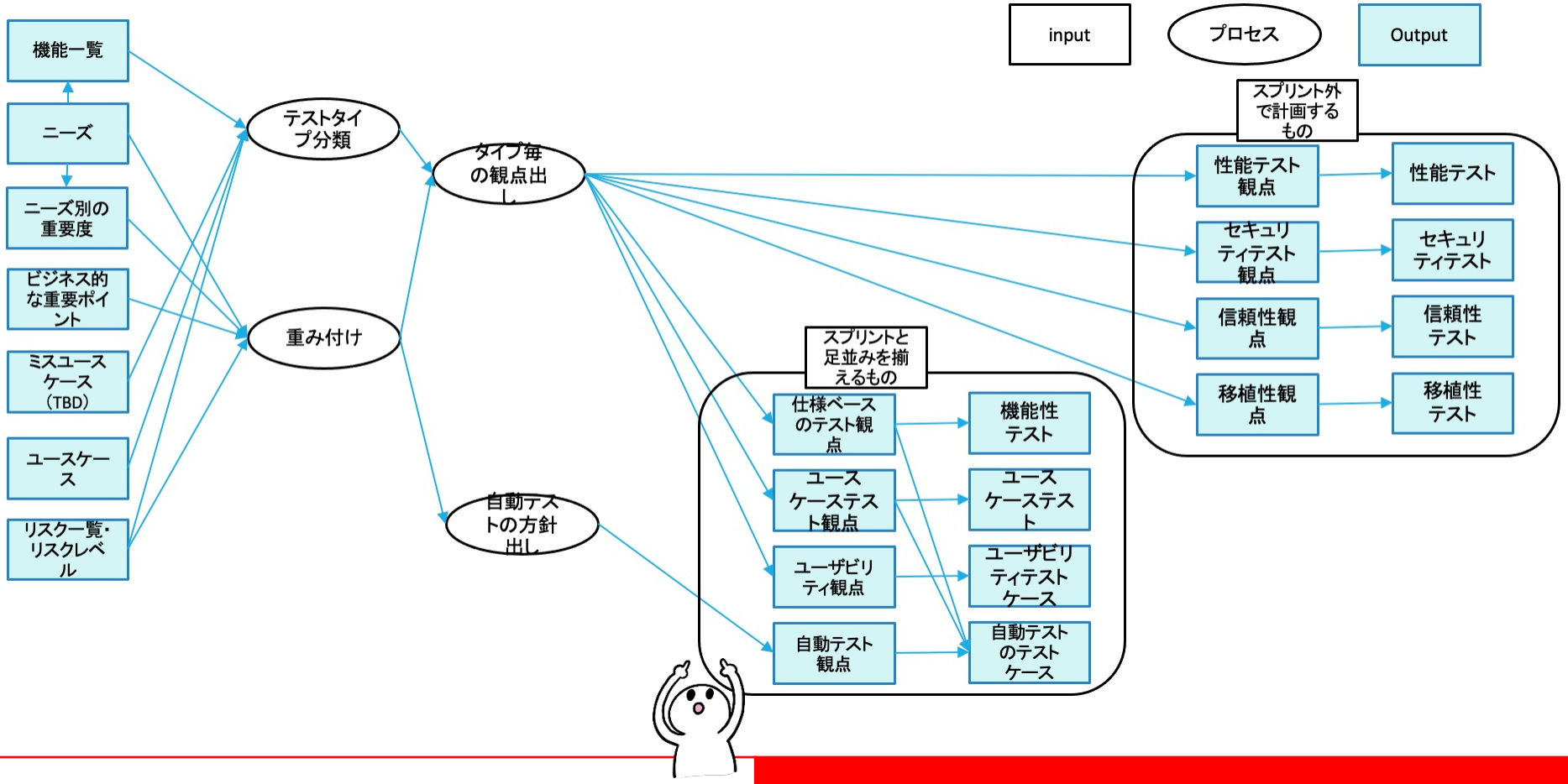
テストの全体像



どんなテストベースからどんなテスト分析を行っていくかがまとめられています。

「改」の部分のように、後から追加になった部分は追加して関係性をわかりやすくするとよいでしょう。

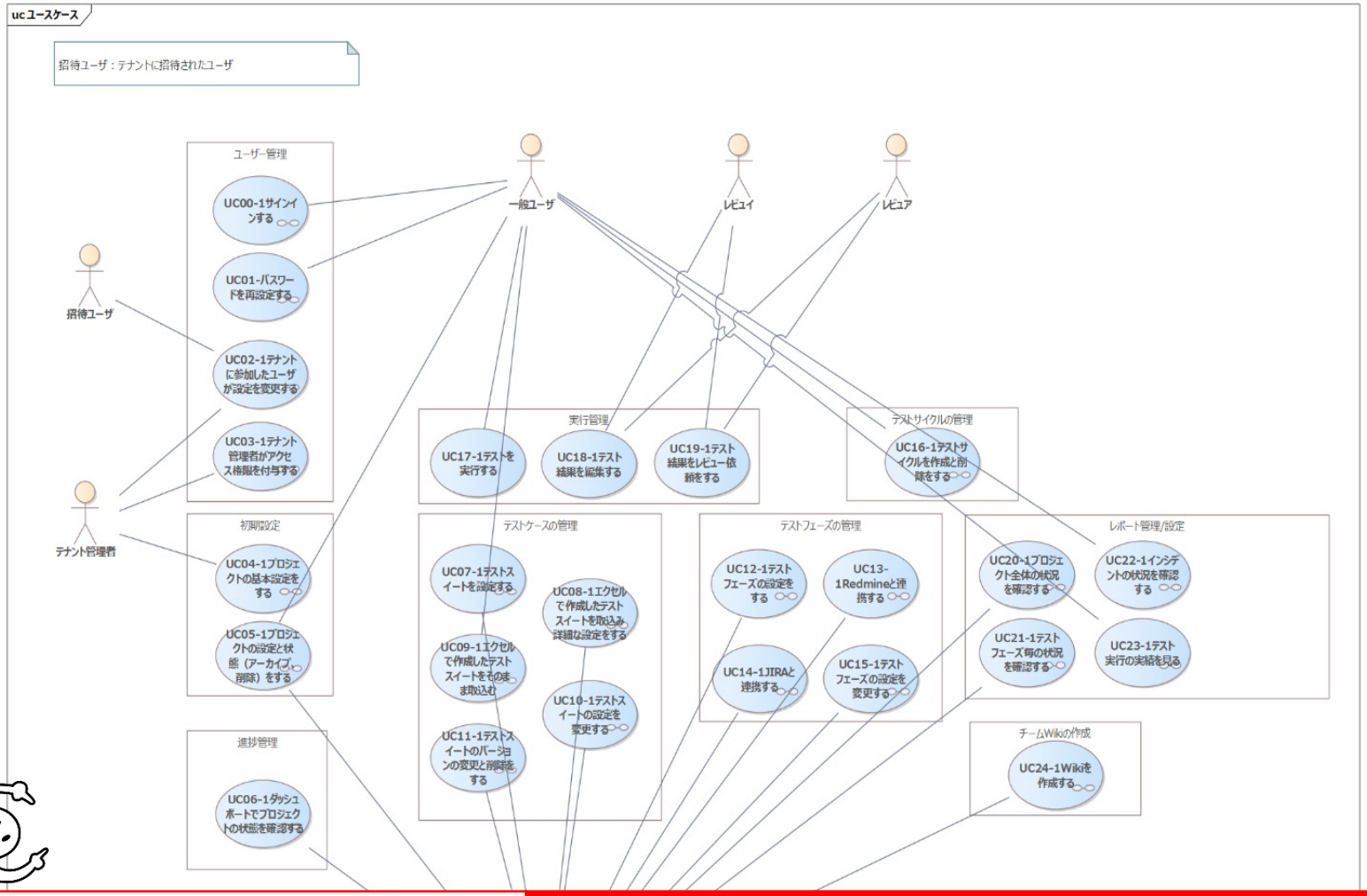
テストの全体像



自分たちが行うテスト分析からどんなテストをいつ行うか、まとめられています。

要求分析もいろんな観点から複数の方法で分析してみると、よりよいテスト設計につながります。

テスト要求分析



テスト要求分析として、仕様をユースケースで整理しています。

この他のやり方でも分析できないか考えてみてください。

ユースケース分析

優先度定義

影響度	説明	ユースケーステストの実施
A	主要なユースケースであり、期待通り動作しない場合、業務上大きな影響があると思われるユースケース	ユースケーステストの実施を強く推奨する
B	利用頻度が高いユースケースだが、期待通り動作しない場合でも回避策があるユースケース	ユースケーステストの実施を推奨するが、場合によっては割愛可能
C	発生頻度が低い、または期待通り動作しなくても影響が少ないと思われるユースケース	変更内容に応じて実施を推奨

ユースケース一覧

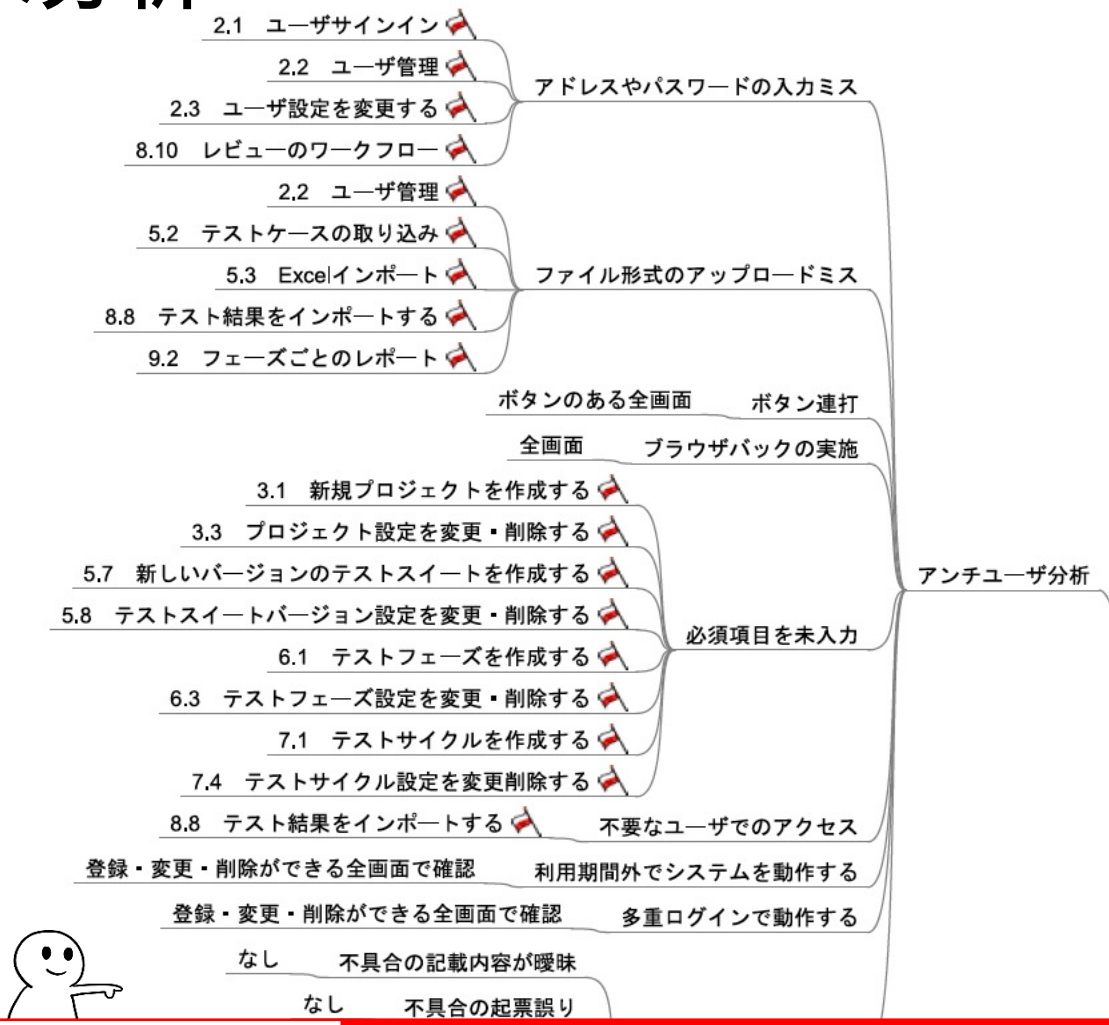
001 秘密保持契約を遵守するため、招待したユーザのみログインさせる

ID	パス種別	概要	優先度
001_01	正常パス	ユーザは正しいメールアドレス、パスワードでログインする	A
001_02	代替パス	ユーザはログイン画面のURL以外を入力してログインし、ログイン後遷移しようとしていたURLにリダイレクトする	A
001_03	代替パス	ユーザはメールアドレス、パスワードの入力ミスによるエラー発生後正しいメールアドレス、パスワードを入力しなおしてログインする	B
001_04	代替パス	ユーザはパスワード再発行を経由してログインする	A
001_05	例外パス	ユーザは登録されている情報と合致しないメールアドレスまたはパスワードを入力したためログインできない	A

こちらもユースケース分析ですが、ユースケース図ではなく、文字でまとめています。優先度もつけて重要なパスがわかりやすいように工夫しています。

リスクや機能の重要度なども優先度に影響することがあるので、他の視点からも考えてみるともっとよくなると思います。

テスト要求分析



各分析の視点からテスト観点を出してそれに関連する機能を洗い出しています。

1つの機能が、複数の観点に紐づくはずです。
その重複をどう効率よくテストしていくかをここから考えていけると良いと思います。

リスク分析

影響度ランク

影響度	説明	会社の損失例
4	機密情報の漏洩	解約＋損害賠償を請求される
3	ユーザーの業務が止まる、目玉機能が動かない	解約される、新規顧客獲得に悪影響
2	ユーザー影響はあるが代替手段あり	クレームが発生する
1	ユーザー被害はないが、問題あり	クレームは発生しないが問題あり

発生頻度　機能の使用頻度と複雑度で評価する

影響度	説明	例
4	使用者数、使用頻度共に多い	ログインやログイン後の画面表示などほぼすべてのユーザが通る領域
3	使用者数、頻度の少なくとも一つが高いが特定シーンでは使用しない可能性がある領域	パスワード変更など全てのユーザーを対象としているが、毎日は使われない領域など
2	使用者数は少ない領域	ユーザ追加やユーザの権限変更など、ユーザー企業の管理者のみしか使わない領域
1	使用頻度は低く、ユーザ影響は弱い	サービス提供側の管理機能など（今回のテストの範囲では言及しない）

リスク一覧

機能系

No.	ユースケース名	起こってほしくない事象	影響度	発生頻度	リスク判定
28	秘密保持契約を遵守するため、招待したユーザのみログインさせ、また、設定した通りの権限とプロジェクトへのアクセス権を付与する	正しいID、パスの組み合わせでログインできない	3	3	大
29		誤ったID、パスの組み合わせでログインに成功してしまう	4	3	大
30		ログインを規定回数失敗してもロックがかからない	4	2	大
31		一定期間操作しなかった後ログアウトしない	4	2	大
32		登録のないメールアドレスにパスワード再発行メールが送信できてしまう	4	1	中
33		パスワード再設定用URLが期限後も使用できる	2	1	低
34	招待したユーザに権限とプロジェクトへのアクセス権を付与する	招待時に意図した以上の権限が付与される	2	1	低
35		ユーザ招待用URLが期限後も使用できる	4	1	中

影響度と発生頻度の定義をしっかり立てて、リスク分析をしています。

このリスクの結果を、今後のプロセスのどこに活用したか見えるようにしておくと信頼度の高いテストになるでしょう。

テスト観点ツリー

大観点	中観点	テスト対象	小観点	更に小観点
データ・状態				
	データ形式	ファイルを扱う機能		
		コード・文字入力を扱	xlsx形式	
			xlsm形式	
			HTML形式	
				予約文字
			CSV形式	
				文字コード
				予約文字
			JSON形式	
				予約文字
	文字	ファイルを扱う機能		

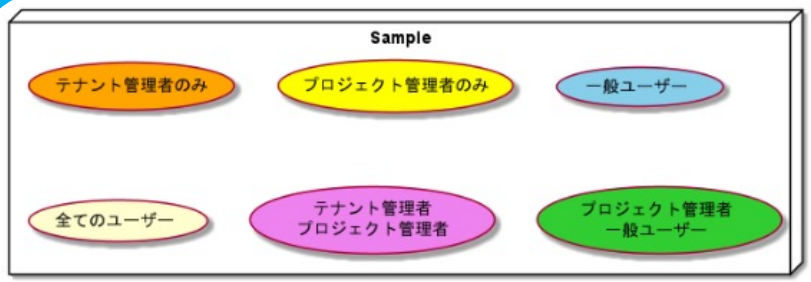
大観点	中観点	テスト対象	小観点	更に小観点
GUI				
	入力チェック	Input関連機能(UI)		
			必須項目	
			文字数	
	画面遷移	Input関連機能(UI)		
		Output関連機能(U	ページ上のボタン押下	
			ブラウザのボタンによる遷移/更新	
			セッション切れ	
			同一画面に遷移	
			ページリンクからの直接遷移	
	画面制御	Input関連機能(UI)		
		Output関連機能(U	ポップアップウインドウ制御	

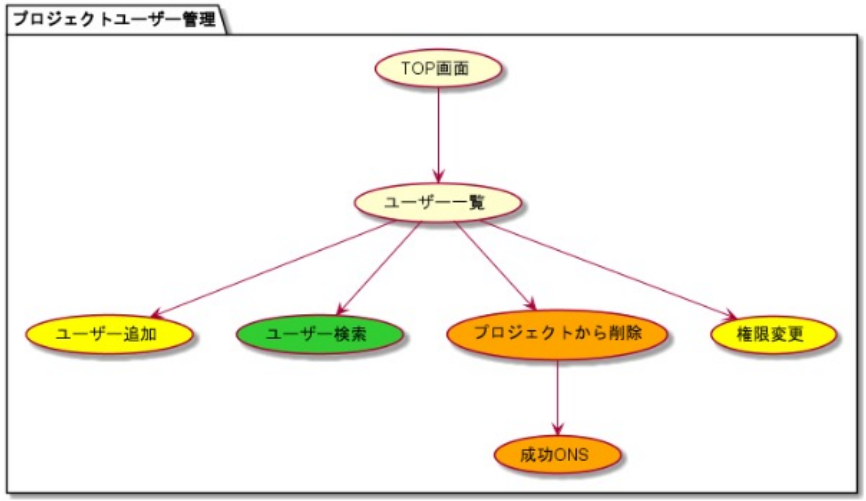
機能1	機能2	機能3	機能4	機能5	観点
チームWiki	ページ一覧表示	「削除」リンク	−	−	ボタン
チームWiki	内容表示	「編集」リンク	−	−	ボタン
チームWiki	内容表示	「ページ一覧」リンク	−	−	ボタン
パスワード再発行	メールアドレス入力	−	−	−	入力チェック
パスワード再発行	メールアドレス入力	空白確認	−	−	入力チェック
テストフェーズ	テストサイクル実施	テスト結果インポート	ファイル選択	−	データ形式
テストフェーズ	テストサイクル実施	インポート成功ONS表示	−	−	文字/画面遷移
テストフェーズ	テストサイクル実施	テスト結果エクスポート	−	−	データ形式

抽象度の大きい観点から詳細な観点を導くとともに機能とも結びつけています。

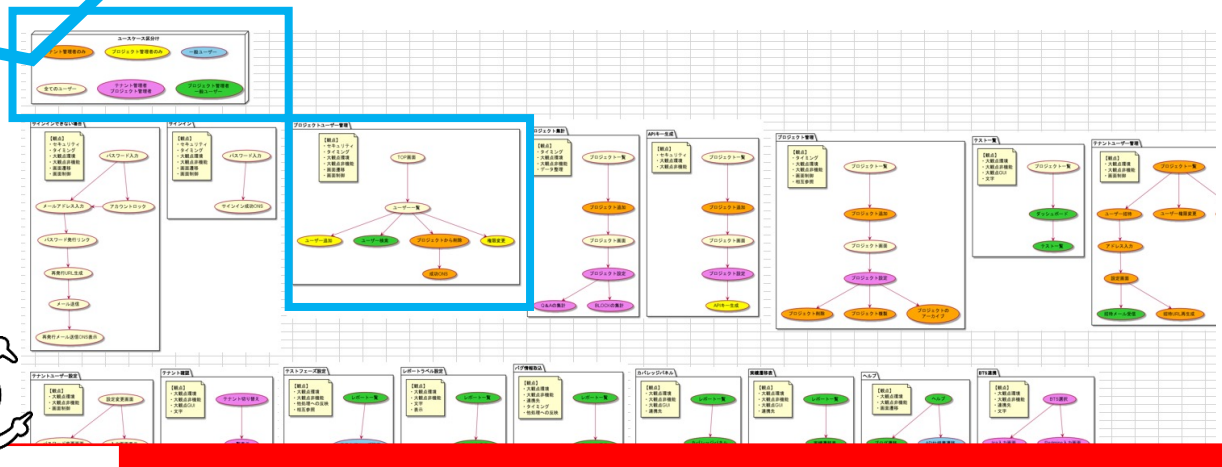
導出した観点と機能を、重複なく整理できると、効率のよいテストができるようになるでしょう。

テストアーキテクチャ設計





拡大



観点と関連画面、ステークホルダーの操作を、機能ごとに箱を作って分類しています。

ここで各テストの重複を減らしたり、優先度や実行順も見えるようにするとみんなにわかりやすいものになると思います。

ジョゼ

テスト詳細設計結果

1．観点因子・水準

No.	観点１	観点２	水準
1	データ・状態	データ形式	xlsx形式
2	データ・状態	データ形式	xlsm形式
3	データ・状態	データ形式	HTML形式
4	データ・状態	データ形式	CSV形式

2．詳細設計

No.	詳細設計番号	観点分類１	観点分類２	詳細観点	関心ごと	水準	性質	制約	テスト設計技法	テストタイプ	テスト目的
1	A-1	データ・状態	データ形式	・水準欄に記載している形式のファイルを使用できることを確認する ・対応する形式外のファイルが使用できないことを確認する	なし	xlsx形式 xlsm形式 HTML形式 CSV形式 JSON形式	ファイル タグ形式 互換性	記載されている形式のファイル以外は使用できない	-	互換性テスト	データ互換性を確認する
2	A-2	データ・状態	文字	・水準欄に記載の文字コードの変換、環境依存文字を使用し文字化けしないことを確認する ・2,4バイト文字が使用できることを確認する	なし	文字コード変換 環境依存文字 改行 空白文字 2, 4バイト文字 RTL文字	文字コード 境界値 機能	表示させられない文字がある	境界値分析	機能テスト	入力可能な文字を確認する
3	A-3	データ・状態	限界値	・入力可能な文字数の境界値、前後値を確認する ・NULL値/0状態を確認する	なし	境界値と前後値 NULL値/0状態	表示数に限りがある 境界値		境界値分析	-	入力可能な文字数を確認する

テスト観点ごとにどんな目的でどの技法を使ってテストするかをまとめています。

残念ながら、せっかくまとめたテストアーキテクチャ設計との関連がわからなくなってしまっていました。トレーサビリティをつけると漏れがみつかりやすくなり、信頼性も上がると思います。

テスト実装結果(自動化テスト)

テス豆_成果物2_データ駆動型テストスクリプト.pdf

```
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
```

テスト自動化する部分のスクリプトを実際に書き出しています。

こちらも、全てもれなくできているかがわかるようにまとめられると良いと思います。

テストケース(手動テスト)

性能効率性

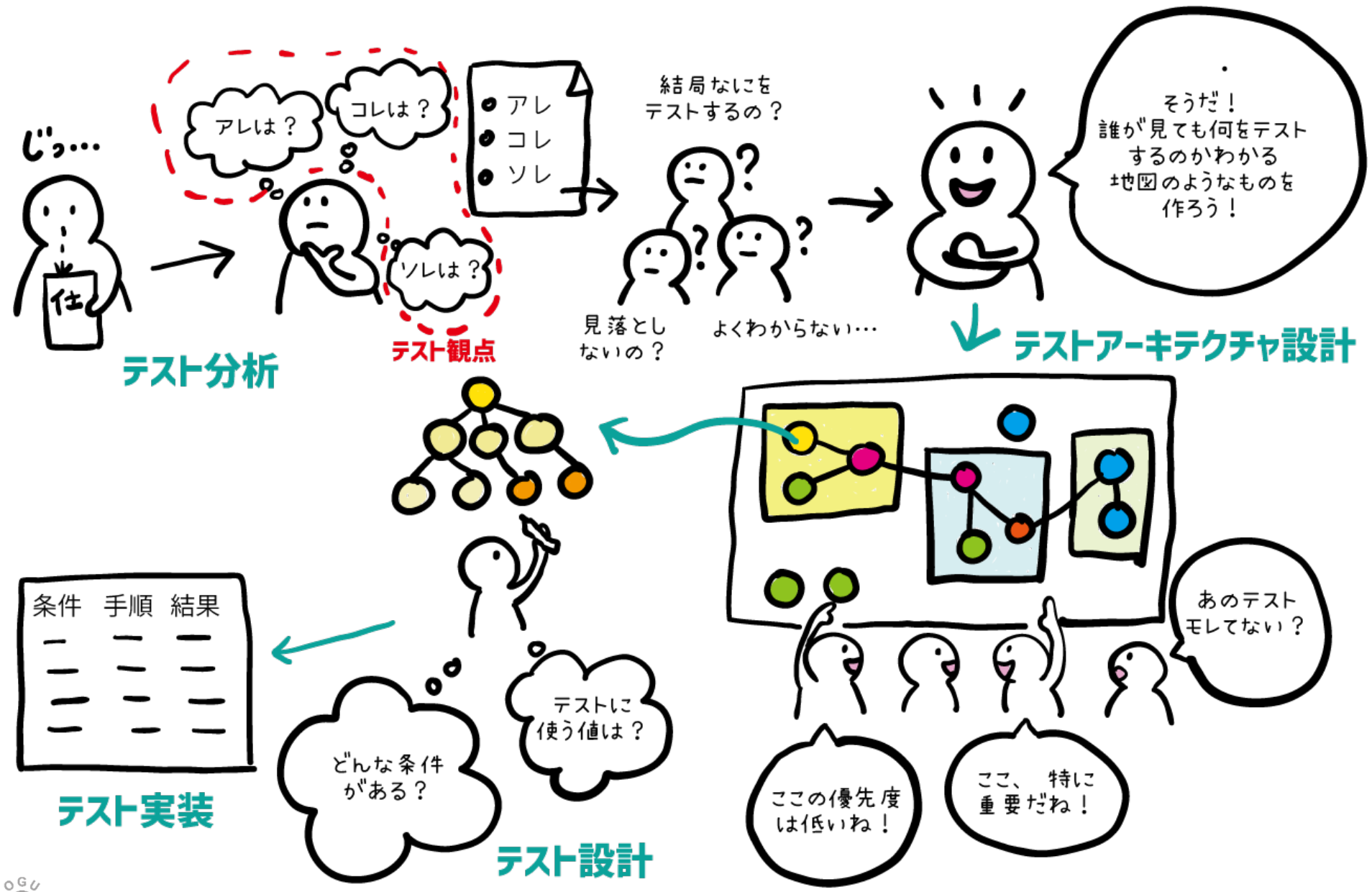
■性能テスト 【テナント管理者】
①時間効率性、②資源効率性、③容量満足性

No	機能名	確認ポイント	ユーザ権限種別	自動/手動	優先度	手順	期待結果
3	テストスイートダウンロード	テストスイートのダウンロード確認 ・想定最大ファイルサイズ = ファイルサイズ	テナント管理者	手動	中	①オンラインで作成済のテストケースをダウンロードする ②テストケースは以下をダウンロードする 想定の最大ファイルサイズと同等のファイルをダウンロード	・Excel形式でダウンロードできること ・ダウンロード処理時間に影響がないこと ・QF機能全体に遅延等、影響が無いこと
4	テストフェーズの新規追加	テストフェーズの新規追加におけるレビュワーのメールアドレス確認 ・想定最大レビュワーアドレス件数 = レビュワーメールアドレス ・想定最大レビュワーアドレス件数 <= レビュワーメールアドレス	テナント管理者	手動	高	①テストフェーズの新規追加登録を行う ②レビュワーのメールアドレス入力は以下条件を設定してそれぞれ登録する 想定の最大レビュワーアドレス件数と同等のメールアドレス件数を入力 想定の最大レビュワーアドレス件数を超えるメールアドレス件数を入力	・想定の最大レビュワーアドレス件数と同等のメールアドレス件数を入力でテストフェーズが新規作成され、テストフェーズ一覧に表示されること ・登録処理時間に影響がないこと ・QF機能全体に遅延等、影響が無いこと ・想定の最大レビュワーアドレス件数を超えるメールアドレス件数を入力できないこと ・出力されるメッセージが仕様通りであること
5	テストフェーズ設定変更	テストフェーズ設定変更の同時更新確認 ・想定する最大可能人数による一斉更新	テナント管理者	手動	中	①テストフェーズの変更を行う ②変更は以下で更新する 想定の最大可能人数による一斉「更新する」ボタン押下	・想定の最大可能人数による一斉更新で更新ができること ・更新処理時間に影響がないこと ・QF機能全体に遅延等、影響が無いこと
6	チームWiki作成	チームWiki新規作成確認 ・想定最大ファイルサイズ = ファイルサイズ ・想定最大ファイルサイズ <= ファイルサイズ ・想定する最大可能人数による一斉更新	テナント管理者	手動	低	①チームWiki新規作成を行う ②添付ファイルは以下条件を追加しそれぞれ更新する 想定の最大容量と同等のファイルを取り込む 想定の最大容量と同等のファイルをドロップする 想定の最大容量を超えるファイルを取り込む 想定の最大容量を超えるファイルをドロップする ②更新は以下で行う 想定の最大可能人数による一斉「更新する」ボタン押下	・想定の最大容量と同等のファイルを取り込みで、更新できること ・想定の最大容量と同等のファイルをドロップで、更新できること ・想定の最大可能人数による一斉更新できること ・登録処理時間に影響がないこと ・QF機能全体に遅延等、影響が無いこと ・想定の最大容量を超えるファイルを取り込みで、更新できないこと ・想定の最大容量を超えるファイルをドロップで、更新できないこと ・出力されるメッセージが仕様通りであること

実際にテストを実行するためのテスト条件、手順、期待結果などが書かれています。

ここで利用するテストの値や条件などを、テスト技法を使って導きだせるようになると良いと思います。

まとめ


じっ…
仕
アレは？
コレは？
ソレは？
テスト分析
テスト観点
アレ
コレ
ソレ
結局なにを
テストするの？
見落とし
ないの？
よくわからない…
そうだ！
誰が見ても何をテスト
するのかわかる
地図のようなものを
作ろう！
テストアーキテクチャ設計
条件 手順 結果
テスト実装
どんな条件
がある？
テストに
使う値は？
テスト設計
あのテスト
モレてない？
ここの優先度
は低いね！
ここ、 特に
重要だね！