割り勘支援アプリ
テスト設計のアピールポイント

チーム名　：てす娘び
メンバー　：小出、小林

Agenda

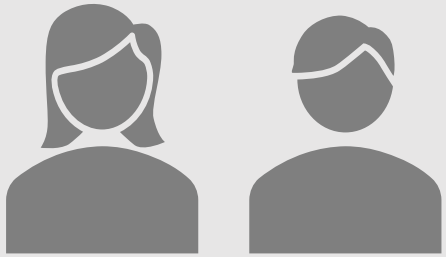
1. チーム紹介
2. テスト概要
3. テスト設計のプロセス
    - 3-1. テスト要求分析
    - 3-2. テストアーキテクチャ設計
    - 3-3. テスト詳細設計
    - 3-4. テスト実装
4. まとめ

1. チーム紹介

チーム名　てす娘び

構成　テスト2年目と1年目の初心者で構成

由来　テスト設計初心者の、女性2名で、
お米どころ新潟から参加する



小林　小出

2. テスト概要

評価対象　**Warikan**：割り勘を支援するためのスマートフォンアプリケーション

担当のテスト　**システムテスト**

要求補足書より

テストの目的

1. 用途を満たしていること
2. リリースできる品質レベルであること
3. テストベースへ改善のフィードバックを行うこと

チームの責務

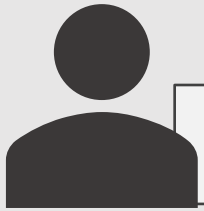
1. 高頻度の仕様変更、リリースに対応すること
2. テストの再現性を確保すること
3. アジリティ重視の開発のため、テスト活動の工数はなるべく小さく実施すること

3. テスト設計のプロセス

3-1. テスト要求分析


テスト
要求分析
テスト
アーキテクチャ
設計
テスト
詳細設計
テスト
実装
利用分析
テスト方針
の検討
機能と観点の
洗い出し
仕様書への
フィードバック
テスト方法の
検討
構造化による観点
漏れの洗い出し
機能テストの
詳細設計
探索的テストの
詳細設計
機能テストの
手順書作成
探索的テストの
手順書作成

3-1. テスト要求分析　- 利用分析 -



割り勘計算を行い集金を行う人

よく使う
利用フロー

行動分析

ユーザーストーリー

求める
品質

3C分析

リリースできるレベルの品質

よく使う
機能

利用率の
分析

アプリの利用用途

3-1. テスト要求分析　– テスト方針の検討 -

テスト依頼元の要求を満たすために

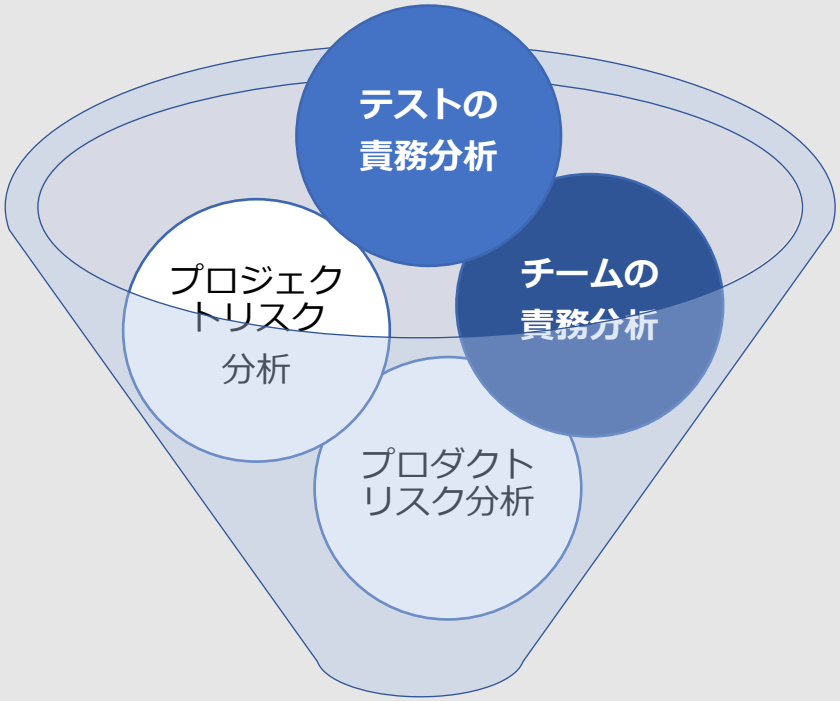
コンセプト　**テストの目的、チームの責務を明確にして、テスト設計する**

要求補足書

責務分析・リスク分析

システムテスト方針

3-1. テスト要求分析　– 責務分析 -



システムテスト
方針

テストの責務分析

	コンポーネントテスト	結合テスト	システムテスト	結合テストとシステムテストの違い	システムテストの方針
テスト方法	実機を使用しない	実機を使用しない（エミュレータ）	実機を使用する	結合テストでは実機評価をしていない	実機を使用して評価を行う
保証内容	・実装に問題ないこと	・起動できること ・一連の画面遷移ができること	・用途を満たしていること → ①割り勘計算 ②ジャスPayによる集金 ③登録して種類をユーザーができること[※1] ・リリースできる品質レベルであること → ユーザーが下表のようなことができることを確認する[※1] ①正確に割り勘計算できること ②ジャスPayによる集金が正確にできること ③入力した情報を正確に登録でき、確認できること	結合テストでは、ユーザーの利用用途や、リリースできる品質レベルであることを保証していない	ユーザーの利用用途や、リリースできる品質レベルであることを保証する
仕様書	未確認	未確認	確認し、フィードバックを行う	結合テストでは、仕様書を確認していない	仕様書を確認し、フィードバックを行う
より詳細な責務を定義↓					
	• リスクの軽減 • コンポーネントの機能的振る舞いが設計および仕様通りであることの検証	• リスクの軽減 • インターフェースの機能的振る舞いが設計および仕様通りであることの検証	• リスクの軽減 • システムの機能的/非機能的振る舞いが設計および仕様通りであることの検証		

各テストの責務を定義
→ システムテストで確認すること
を明らかにする

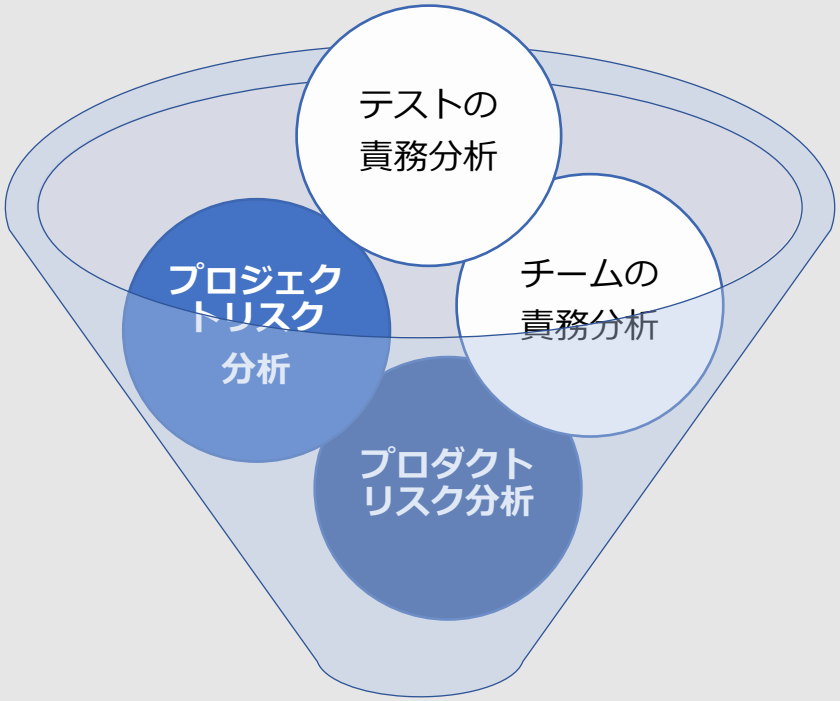
チームの責務分析

要求補足書の内容	存在するリスクの種類	
• 高頻度の仕様変更、リリースに対応する。 仕様変更への対応しやすさや、テストの繰り返し再実施のやりやすさを確保できるように、テスト成果物を作成すること。	プロダクトリスク / プロジェクトリスク	・(プロダクトリス ・(プロジェクトリ
• 将来的なチーム内でのテスト担当のローテーションや引継ぎのため、テスト実	プロジェクトリスク	・(プロジェクトリ

要求補足書記載 /
「3. システムテストを担当するチームの責務」より

リスクが
発生するか
検討する

3-1. テスト要求分析　– リスク分析 -



プロダクトリスク分析

プロダクトリスク	プロダクトリスクの分析手順	• どこが上手くいかないか（リスク）を分析する（定期的に再評価する）。	• リスクの重要度を決める。			リスクの重要度の根拠	• 重大なリスクを処理する方法を実装する。	システムテストの方針
			発生確率	影響度	予防策検討対象			
• ソフトウェアの意図されている機能が仕様通りには動かないかもしれない。		・設計が間違っており、仕様通りに動作せず、ユーザーの利用目的が達成できなくなるリスク ・コーディングが間違っており、仕様通りに動作せず、ユーザーの利用目的が達成できなくなるリスク	2	4	•	結合テストでは確認していないため発生確率は高い。最低限のユーザーの目的が達成できないことは回避したいので、基本は仕様通りに動作する必要がある。だが、頻繁な仕様変更があるため、その仕様通りに動け辛いが多い	最低限のユーザーの目的が達成できる機能が備わっていることを保証する	仕様書から機能や観点を洗い出す
• ソフトウェアの意図されている機能がユーザー、顧客、およびまたはステークホルダーのニーズ通りには動かないかもしれない。		・不適切な仕様に沿って作成されており、ユーザーの利用用途が達成できなくなるリスク ・市場の分析が不足しており、ユーザーの利用用途が達成できなくなるリスク ・仕様変更に対応できないリスク	3	4		結合テストでは確認していないため発生確率は高い。ニーズ通りに動かない場合、ユーザーは不満を抱くので、ニーズ通りに動作する必要がある。だが、抱く不満の度合	達成できないと抱く不満が高くなるユーザーニーズの通りに動作していることを保証する 仕様変更後も動作することを確認する	・ユーザーニーズを考慮して機能や観点を洗い出す ・アップデートに対応できるようにする
• システムアーキテクチャーが非機能要件を十分にサポートしないことがある。		実機で非機能要件が満たされない場合に、不満を感じるリスク	3	3				
• 特定の計算結果が状況によって正しくないことがある。		利用者に金銭的		5				
• ループ制御構造が正しくコーディングされていないことがある。		画面遷移不正により、ユーザーの利用目的が達成できなくなるリスク 一部の環境下で目的を達成できなくなるリスク	2	3				

リスクを
網羅的に洗い出す

プロジェクトリスク分析

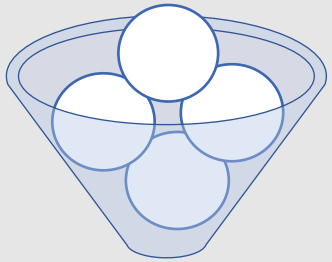
プロジェクトリスク		発生確率	影響度	予防策検討対象	根拠	予防策	システムテストの方針
• プロジェクトの懸念事項：	o リリース、タスク完了、終了基準または完了（done）の定義の達成が遅延することがある。	2	4	•	リリースは高頻度のため、遅延があっても、ユーザーへの影響は低いが、顧客やステークホルダーの目的は達成されない。	工数削減の施策を検討する	優先度の低い機能の確認に使う工数は最小減にする
	o 不正確な見積り、優先度の高いプロジェクトへの資金の再割り当て、組織全体での経費節減により資金が不足することがある。		4	•	資金不足になった場合、評価工数が少なくなり、ユーザーに必要な品質を確保できない場合がある	確認対象の機能に優先度をつける。	確認対象の機能に優先度をつける。
	o プロジェクト終盤での変更により作業の大規模なやり直しが必要な場合がある。	2	3	•	評価作業の手戻りが起きた場合のこのリスクが発生する。リリースは高頻度のため、大規模なやり直しが発生しても、直接的なユーザーへの影響は低いが、顧客やステークホルダは不満を抱く	手戻りが発生した場合、手戻りの範囲を特定できるようにし、過剰な削減できるようにする。素早く、顧客やステークホルダーと情報共有できるようにしておく	テスト結果を視覚的に確認できるようにする
• 組織の懸念事項：	o 人員不足、および	3	4	•	ローテーション		
	o 人間関係				チームが解散		
					仕様に関する		
					テストが不足		
					テスト担当者		
					正し評価し		
					必要な機能だった場合、目的を満たせない場合がある。	ン時に、別の担当者が正しく評価できるようにする。	
					重要な要件を定義できず、品質レベルの低いままリリースしてしまう可能性がある。 仕様変更もした場合に要件を定義できない可能性がある。	仕様書とニーズの両側面から要件を定義する	・仕様書から機能や観点を洗い出す ・ユーザーニーズを考慮して機能や観点を洗い出す ・仕様変更に対応できるテスト設計にする

予防策を検討する

リスクの重要度を
検討する

システムテスト
方針

3-1. テスト要求分析　– テスト方針の検討 -



システム
テスト方針

システムテスト方針リスト

		ドキュメントが含まれる工程	要求分析						
システムテスト方針の参照元	関連のある工程	システムテストの方針	てす娘び_成果物0_テスト開発プロセス.xlsx	てす娘び_成果物1_001_要求分析の説明.docx	てす娘び_成果物1_002_利用分析.xlsx	てす娘び_成果物1_003_システムテストの方針.xlsx	てす娘び_成果物1_004_機能と観点を洗い出す方法.xlsx	てす娘び_成果物1_005_フィードバック一覧.xlsx	てす娘び_成果物1_006_観点マップ
プロジェクトリスク分析	全体	仕様変更に対応できるテスト設計にする	●	●	-	-	●	-	-
テストの責務分析		独立したチームがテストを行う	●	●	-	-	●	-	-
テストの責務分析 プロジェクトリスク分析		ユーザーの利用用途や、リリースできる品質レベルであることを保証する	●	●	-	-	●	-	-
テストの責務分析		実機を使用して評価を行う	●	●	-	-	●	-	-
テストの責務分析	要求分析	仕様書を確認し、フィードバックを行う	●	●	-	-	●	-	-
プロダクトリスク分析		仕様書から機能や観点を洗い出す	●	●	-	-	●	-	-

システムテスト方針に沿って
作成した成果物を視覚化する

効果 1　**各成果物と相互の
トレーサビリティをとれる**

効果2　**システムテスト方針の
達成度合い, 達成根拠を
確認できる**

※ 仕様変更でプロダクトリスクが変更した場合も想定して設計

3-1. テスト要求分析 – 機能・観点の洗い出し -

＋

ユーザーストーリー

アジリティ重視の開発
→ 頻繁に変更される
→ **機能・観点の抜け漏れが想定される**

アプリの位置づけ・利用ユーザーは大きく変わらない
→ ユーザーストーリーは大きく変わらない
→ **ユーザーストーリーも洗い出しに利用する**

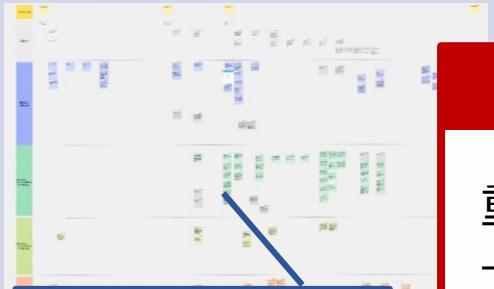
機能観点マップ

横軸：ユーザーストーリー

高

優先度

低



機能・観点

効果

重要な機能・観点を漏れなく確認
→ **リリースできる品質を保証**できる

3-1. テスト要求分析 – 機能・観点の洗い出し -

アプリを利用して生じる
不満が少ない ことが優先される

※ リリースできる品質レベルを保証するため

優先度	
1	目的達成に必要な機能
2	満たされないとユーザーが不満を抱くリスクが高い機能
3	満たされないとユーザーが不満を抱くリスクが低い機能
4	満たされなくても目的を達成できる機能

機能観点マップ

横軸：ユーザーストーリー

優先度

高

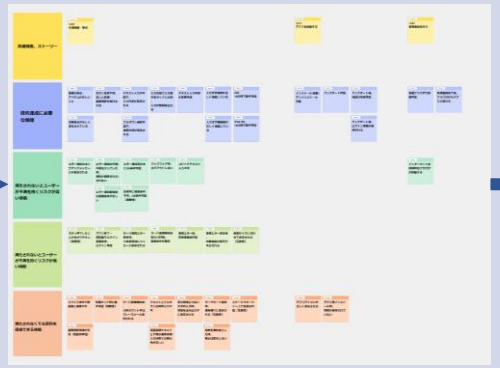
低

非機能要件	観点の付箋として優先度を付与
不明点・提案	**フィードバック一覧**に記載

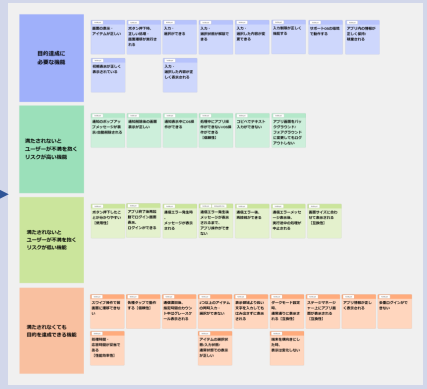
3-1. テスト要求分析　– 機能・観点の整理 -

機能観点マップ の整理過程

機能・観点を洗い出す



共通するものを
整理する



抽象化してまとめる

機能観点リストにまとめる

機能観点ID	機能観点の内容
A1	画面の表示・アイテムが正しい
A2	初期表示が正しく表示されている
A3	ボタン押下時、正しい処理・画面遷移が実行される
A4	入力・選択が可能であること
A5	入力・選択内容が正しく表示されること
A6	入力・選択状態が解除できる
A7	入力・選択した内容が変更できる
A8	入力制限が正しく機能する
A9	サポートOSの環境で動作する
A10	アプリ内の情報が正しく保持/破棄される
B1	通知のポップアップメッセージが表示/削除/自動削除される
B2	通知削除後の画面表示が正しい
B3	通知表示中にOS操作ができる
B4	処理中にアプリ操作ができない/OS操作ができる
B5	コピペでテキスト入力ができない

効果

トレーサビリティに対応できる
仕様変更に対応しやすい

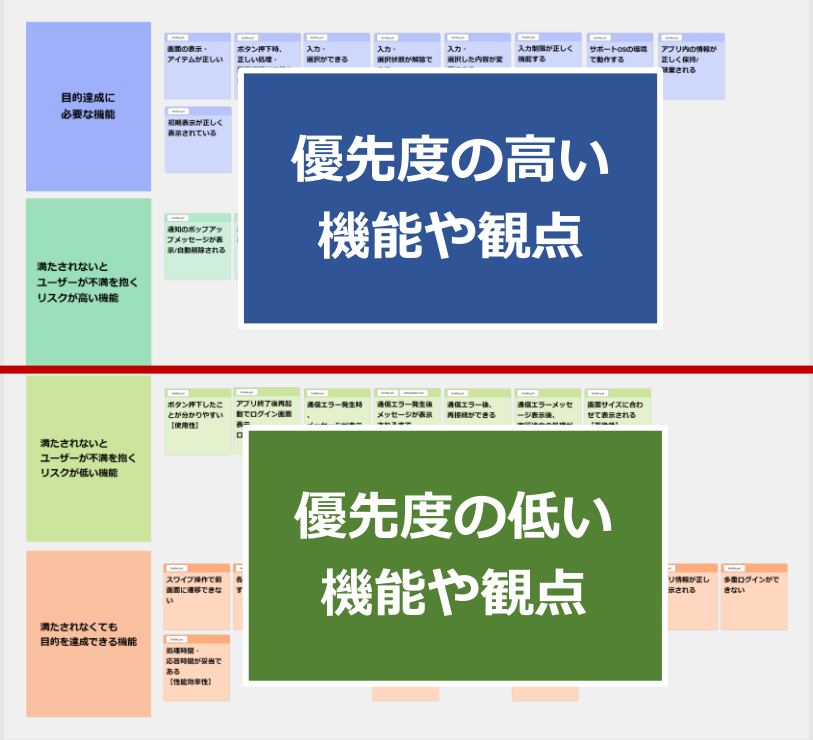
3-2. テストアーキテクチャ設計

3-2. テストアーキテクチャ設計 – テスト方法の検討 -

確認が漏れた場合に

- ユーザーの利用用途が達成できなくなる可能性がある機能
- ユーザーが許容できない品質レベルになる可能性がある機能

優先度の高い機能や観点



経験値、テスト環境　に依存しない方法で確認

▼

機能とOSのマトリクスを用いて機能要件に適合しているか確認するスクリプトテスト

→ **機能テスト**　と呼ぶ

テスト活動全体の工数を小さくできる方法で確認

▼

テストチャーターとセッションベースドテストを適用して探索的に行うテスト

→ **探索的テスト**　と呼ぶ

※ 探索的テストでは、テストチャーター / テストレポート　/ セッションあたりの時間 も検討

3-2. テストアーキテクチャ設計 – 全体の構造 -

機能テストの機能群

目的達成に必要な機能群

満たされないとユーザーが
不満を抱くリスクが高い機能群

探索的テストの機能群

満たされないとユーザーが
不満を抱くリスクが低い機能群

満たされなくても目的を
達成できる機能群

機能テストの構造

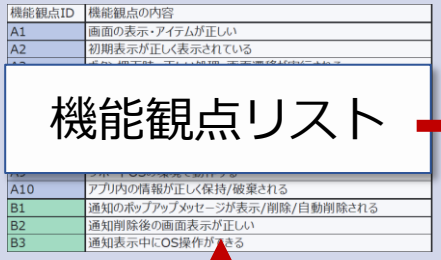
探索的テストの構造

俯瞰して抜け漏れを確認
できるように機能・観点 を抽象化

3-2. テストアーキテクチャ設計 - 機能テスト -

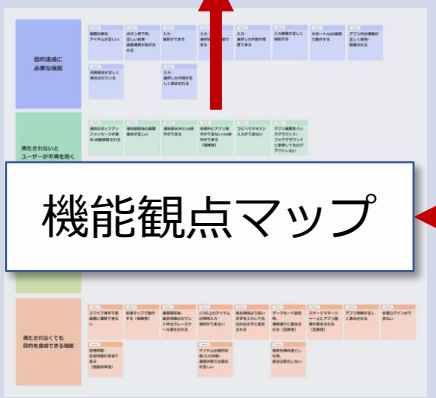
機能テスト　ツリー構造にする

テスト要求分析



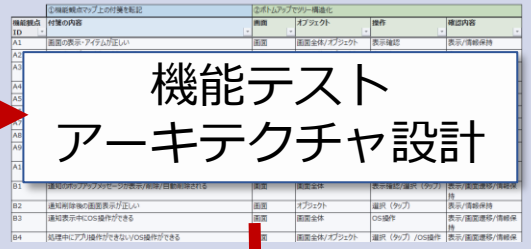
機能観点ID	機能観点の内容
A1	画面の表示・アイテムが正しい
A2	初期表示が正しく表示されている
A10	アプリ内の情報が正しく保持/破棄される
B1	通知のポップアップメッセージが表示/削除/自動削除される
B2	通知削除後の画面表示が正しい
B3	通知表示中にOS操作ができる

機能観点リスト

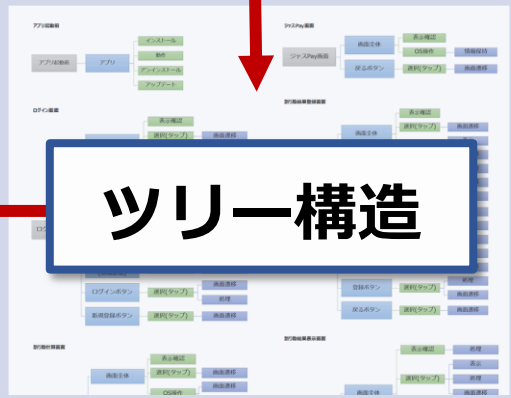


機能観点マップ

テストアーキテクチャ設計



機能テスト
アーキテクチャ設計



ツリー構造

抜け漏れは
機能観点マップに追加

テスト要求分析

↕

テスト
アーキテクチャ設計

効果　**観点の抜け漏れ、重複 を防止**

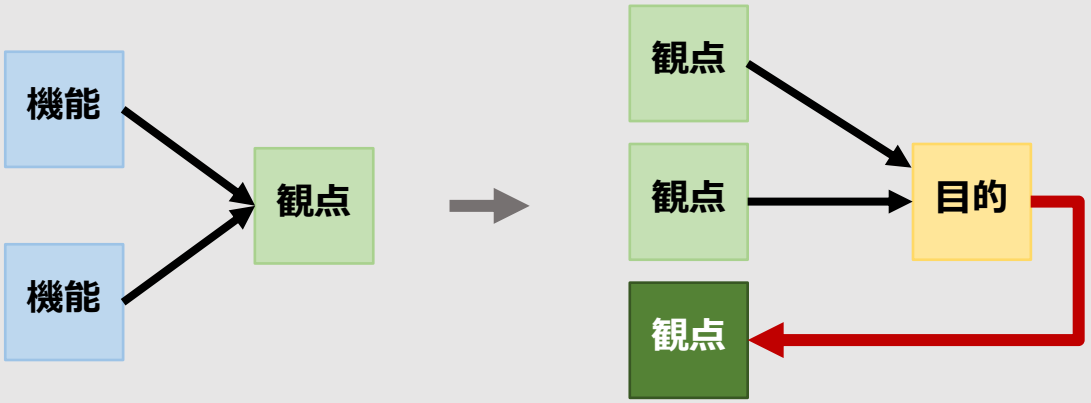
その他検討内容		
	• 実施順の検討	• トレーサビリティ
	• 仕様変更への対応	• 機種選定

3-2. テストアーキテクチャ設計 - 探索的テスト -

探索的テスト　コンテナ構造にする

1 機能を観点に抽象化

2 セッションの目的を設定し、
目的に沿った観点をさらに追加

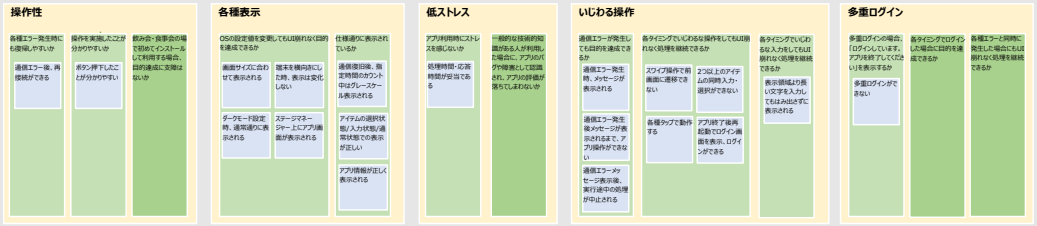


効果

観点の漏れを防止

目的、観点がセッションごとで明確化

3 目的ごとに観点をまとめてコンテナ構造にする



その他検討内容

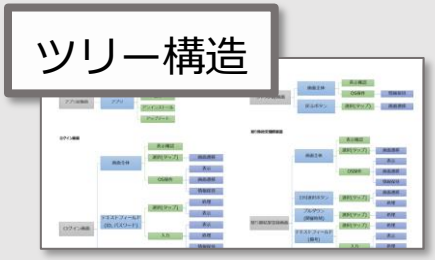
- 実施順の検討
- 仕様変更への対応
- トレーサビリティ
- 機種選定

3-3. テスト詳細設計


テスト
要求分析
テスト
アーキテクチャ
設計
テスト
詳細設計
テスト
実装
利用分析
テスト方針
の検討
機能と観点の
洗い出し
仕様書への
フィードバック
テスト方法の
検討
構造化による観点
漏れの洗い出し
機能テストの
詳細設計
探索的テストの
詳細設計
機能テストの
手順書作成
探索的テストの
手順書作成

3-3. 詳細設計　– 機能テストの詳細設計 -

機能テスト



① ツリー構造から画面ごとの機能を記載する

② 機能や観点から、**テスト技法** を活用してテストパターンを検討する

機能観点ID	詳細設計ID	機能・観点	テストパターン	テスト手順	期待結果	利用機種条件			指定機種	
						UIの確認	OS依存の無い機能	基本機能	iOS	iPad
A2	d2_1	初起動時、ログイン画面が表示される	シート[技法_入力・表示仕様整理]の「初期値・初期表示」を参照	初期値・初期表示を確認	初期値・初期表示が正しく表示されること			●		
A1		値がPICTと同			PICTの通り表示されること	・			アスペクト比①	アスペクト比②
A6	d2_3	IDが入力中の状態で入力欄外をタップした場合は入力状態が解除される(入力欄外 / ボタン)	-	入力フィールドをタップした後、それ以外の領域やログインボタンをタップして入力状態を解除する	入力状態が解除されること		●			
	d2_4	パスワードが入力中の状態で入力欄外をタップした場合は入力状態が解除される(入力欄外 / ボタン)		入力フィールドをタップした後、それ以外の領域やログインボタンをタップして入力状態を解除する	入力状態が解除されること					
A5	d2_5	テキスト入力ができ、入力した内容が表示される (ID/パスワード)	-	1. ログイン画面を表示する 2. ユーザIDのテキストフィールドをタップする 3. 任意のテキスト(半角英字/半角数字/記号を1文字以上)を入力する 4. 画面内のテ	入力した内容が正しく表示されること		●			
A8	d2_6	「ユーザID」には 15 文字のみ入力できる	シート[技法_入力・表示仕様整理]の「境界値分析」を参照	入力フィールドに指						
	d2_7	「ユーザID」には 15 文字以内の範囲外の文字数 (16文字以降は受け付けない)		入力フィールドに指						
	d2_8	「パスワード」には 20文字のみ入力できる		入力フィールドに指						
	d2_9	「パスワード」には 20文字以内の範囲外の文字数(21文字以降は受け付けない)		入力フィールドに指						
	d2_10	入力された「パスワード」は入力後すぐに「●」で入力文字数分マスクする。		入力フィールドに指						
A8	d2_11	「ユーザID」には 半角英数字のみ入力できる	シート[技法_入力・表示仕様整理]の「同値分割法」を参照	入力フィールドに指						
	d2_12	「ユーザID」には 半角英数字の範囲外の文字		入力フィールドに指						
	d2_13	「パスワード」には 半角英数字のみ入力できる		入力フィールドに指						
	d2_14	「パスワード」には 半角英数字の範囲外の文字		入力フィールドに指						
B5	d2_15	「ユーザID」がコピペ入力できないこと	-	入力フィールドに指						
	d2_16	「パスワード」がコピペ入力できないこと		入力フィールドに指						
A8	d2_17	「ユーザID」が空欄のとき、「IDが登録されていないか、パスワードが不正です」のエラーが表示される	-	1. ログイン画面 2. ユーザIDのテ						

ID

画面ごとの機能

テストパターン

確認内容の例

- アップデート、情報の引継ぎ
- 計算機能の重点的確認　など

その他検討内容

- 実施順の反映
- 仕様変更への対応
- トレーサビリティ
- 機種決定

3-3. テスト詳細設計 – 探索的テストの詳細設計 -

探索的テスト

テストチャーター の**項目**の内容を各セッションで検討する

実施順		1
テストチャーター		**操作性**
テストの概要・目的		飲み会・食事会の場で初め して利用するユーザー視点 を実施する
どんな欠陥を見つけたいか		使用性の欠如 / ユーザ 動作ができないこと
そのためにどんなデバイスを使うか	UIの確認	-
	OS依存の無い機能	●
	基本機能	-

セッション名	確認観点	確認観点	機能観点ID	付箋		
操作性	s1_1	各種エラー発生時にも復帰しやすいか	C5	通信エラー後、再		
	s1_2	操作を実施したことが分かりやすいか	C1	ボタン押下したこ 性】		
	s1_3	飲み会・食事会の場で初めてインストールして利用する場合、目的達成に支障はないか	-	-		
各種表示	s2_1	OSの設定値を変更してもUI崩れなく目的を達成できるか	C7			
			D8		時 / ポップアップメッセージ表示時	
			D9	端末を横向きにした時、表示は変化しない	設定例：画面の向き 確認するUI：アプリアイコン / 画面表示	UIに欠け、乱れなく画面が表示されること

テストチャーターの項目

※テストアーキテクチャ設計で検討済

- テストの概要・目的
- どんな欠陥を見つけたいか
- そのためにどんなデバイスを使うか
- 確認観点及び期待結果
- セッション時間の目安

テスト結果に差がでないように**条件・期待を明記**

その他検討内容

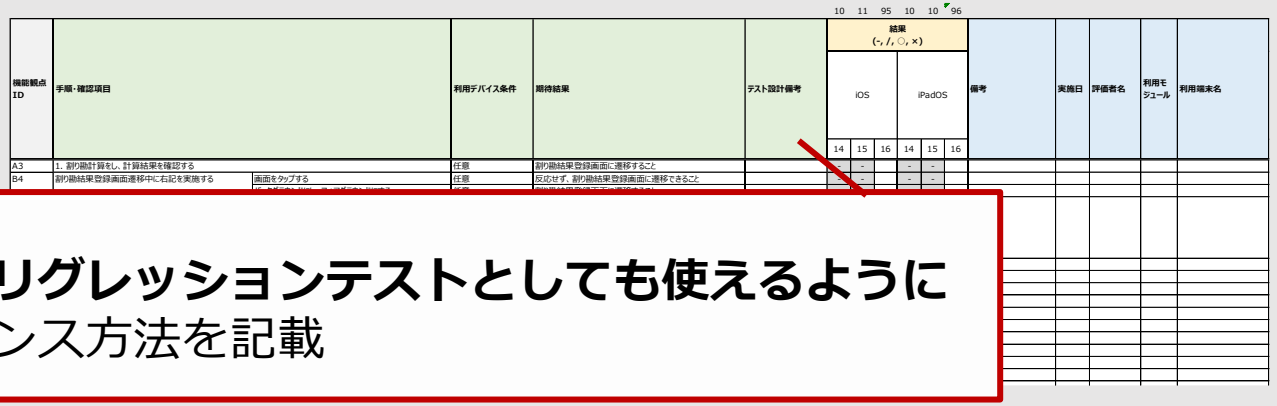
- 実施順の反映
- 仕様変更への対応
- トレーサビリティ
- 機種決定

3-4. テスト実装


テスト
要求分析
テスト
アーキテクチャ
設計
テスト
詳細設計
テスト
実装
利用分析
テスト方針
の検討
機能と観点の
洗い出し
仕様書への
フィードバック
テスト方法の
検討
構造化による観点
漏れの洗い出し
機能テストの
詳細設計
探索的テストの
詳細設計
機能テストの
手順書作成
探索的テストの
手順書作成

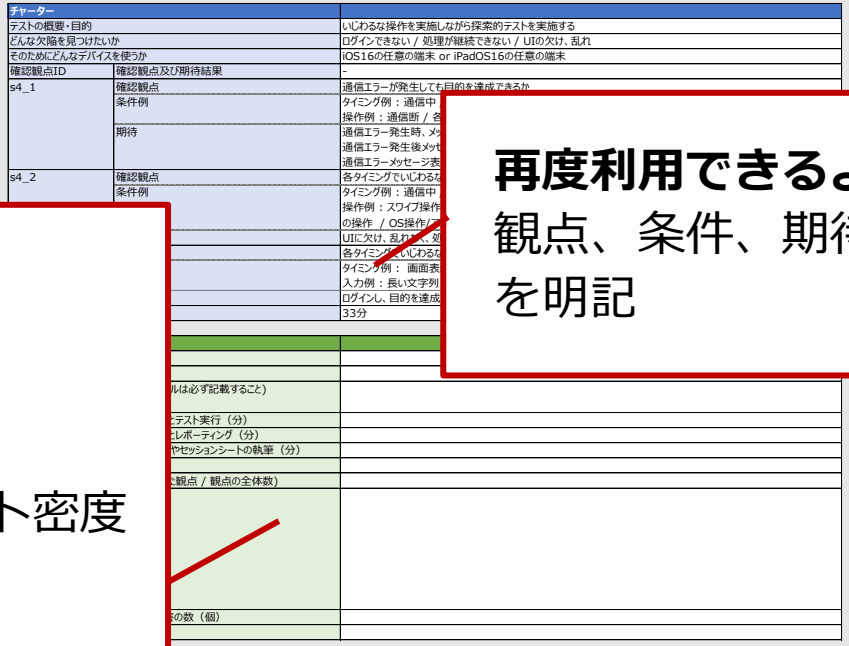
3-4. テスト実装 – 手順書作成 -

機能テスト



次回以降にリグレッションテストとしても使えるように
メンテナンス方法を記載

探索的テスト



再度利用できるような形式
観点、条件、期待結果
を明記

次回のテスト設計で
メンテナンスに活用できる項目を記載

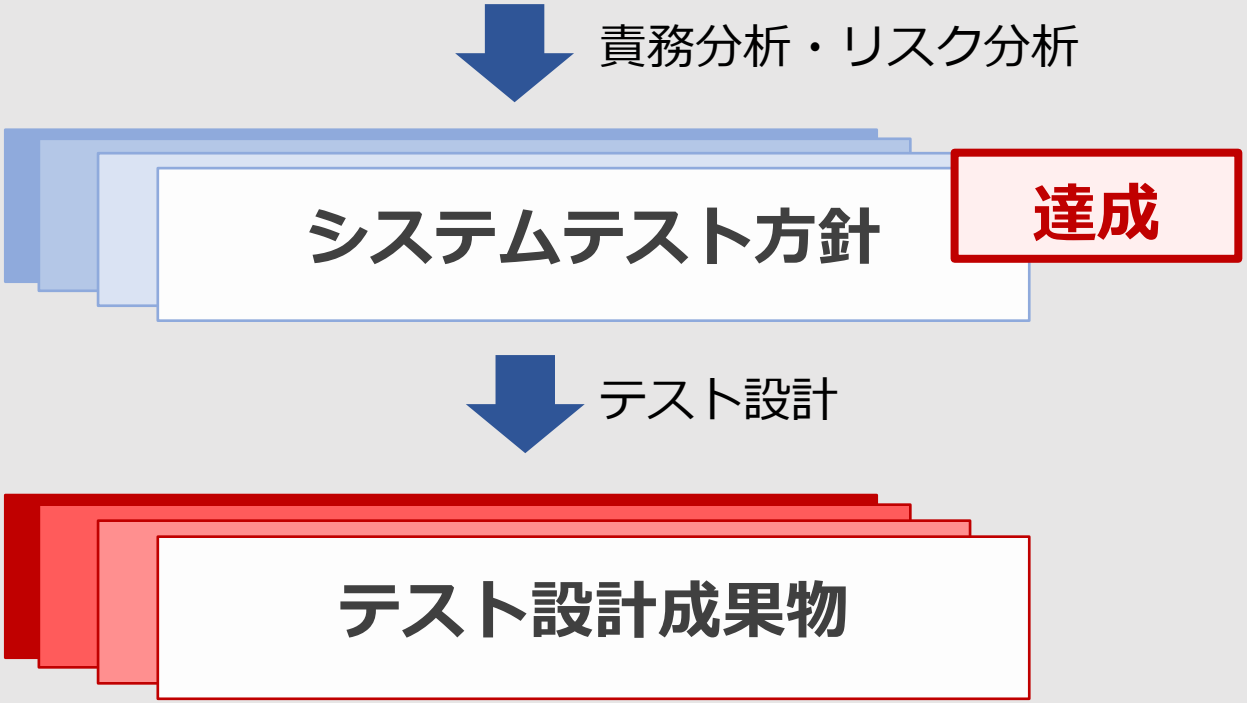
- 想定実行時間に対するテスト密度
- 障害の傾向　など

4. まとめ

テスト依頼元の要求を満たすために

達成

コンセプト **テストの目的、チームの責務を明確にして、テスト設計する**



ご清聴ありがとうございました