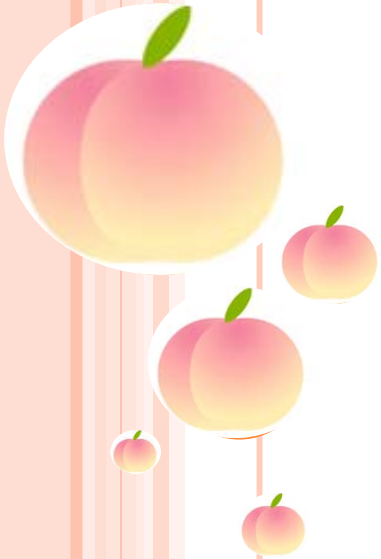


テスト設計コンテスト ’17

モモテツチーム

チーム紹介

- チーム名「モモテツ」の由来

  福島の名産である「モモ(桃)」と「テストせつけいコンテスト」を略して、名付けました。

- 仕事内容

  私たちは、グループ会社で開発している製品の品質保証部門の一員としてソフトウェアテストを担当しています。

- 参加への思い

  「**初めてテストする製品**」、「**自由な立ち位置の設定**」というフレッシュな状況でテスト設計することで、**普段の業務では得づらい**新しい発想や考え方、手法を学び、スキルを向上させたいと思いました。

テスト設計のコンセプト

ASTER通信カラオケシステム

外部ステークホルダー

ユーザー　サプライヤー　オーナー

内部ステークホルダー

企画部門　設計開発部門

結合テストチーム　システムテストチーム

品質保証部門　販売部門

ステークホルダーごとに・・・

いろいろな期待や不安

- みんなで盛り上がりたいなぁ
- 設置トラブルはイヤだなぁ
- 売れるといいなぁ
- 効率的にテストしたいなぁ

普段の業務では

外部ステークホルダーのみしか十分には意識できていない

テスコンでは

内部ステークホルダーを含むステークホルダー全員を安心させるためのテスト設計に挑戦したい

モモテツ

テスト設計のコンセプト

ASTER通信カラオケシステム

外部ステークホルダー

ユーザー　サプライヤー　オーナー

内部ステークホルダー

企画部門　設計開発部門

結合テストチーム　システムテストチーム

品質保証部門　販売部門

ステークホルダーごとに・・・

いろいろな期待や不安

- みんなで盛り上がりたいなぁ
- 設置トラブルはイヤだなぁ
- 売れるといいなぁ
- 効率的にテストしたいなぁ

ステークホルダー全員の

テスト設計によって・・・安心を実現

- ○みんなで盛り上がれます！
- ○設置トラブルはありません！
- ○売れます！
- ○効率的にテストしていきましょう！

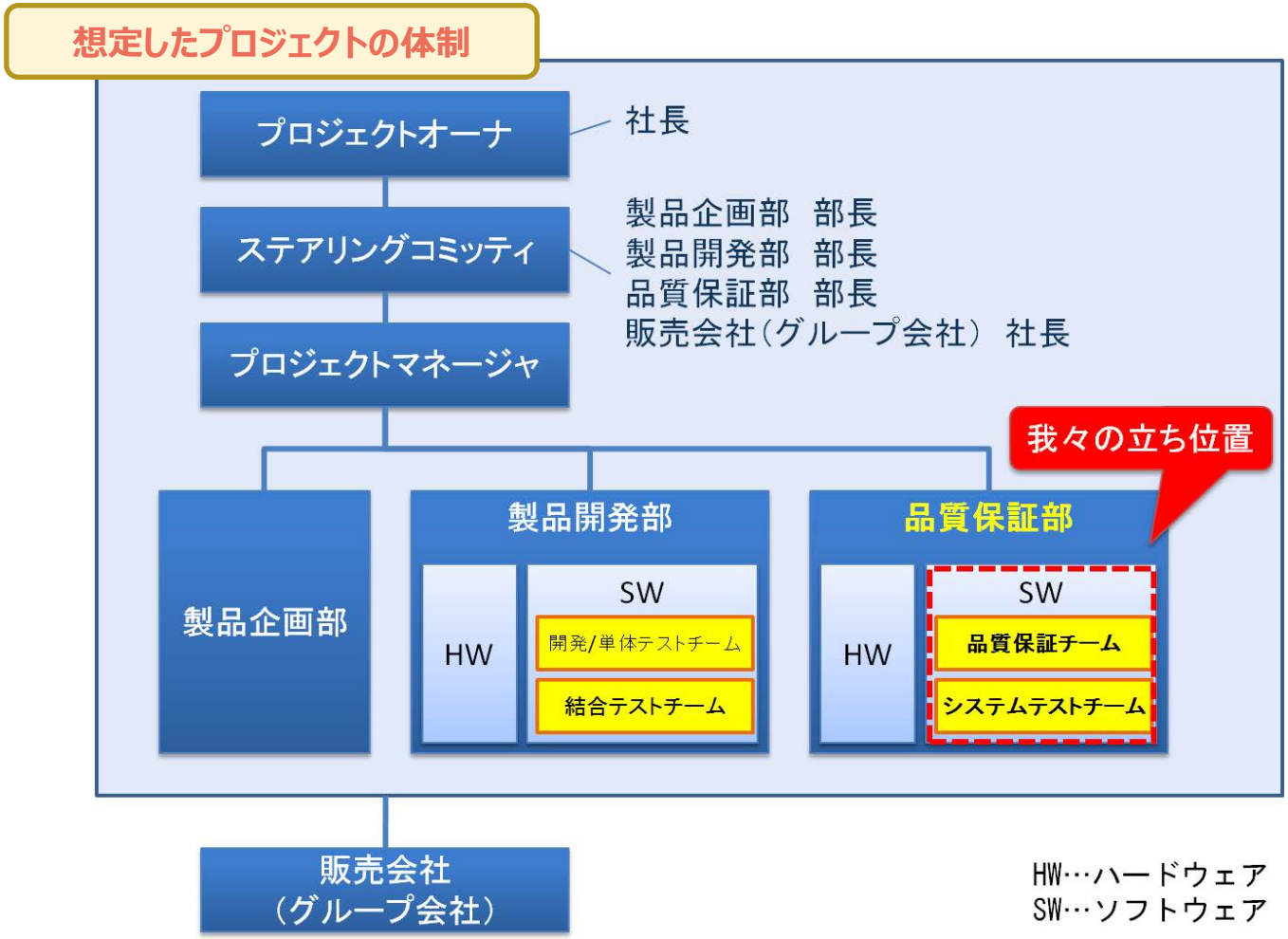
コンセプト

ステークホルダー全員（外部/内部 両方）を安心させるテスト設計をする。
特に、普段の業務で取り組めていない**内部ステークホルダーを意識した施策に重点的**に取り組む。

チームの位置づけと役割

■チームの位置づけ

品質を広い視点で捉えてみたいという思いから、
品質保証部門という立場でテスト設計することにしました。



モモテツ

チームの位置づけと役割

■品質保証部の役割

私たちは、大きくは下記5つの役割を担います。

Ⅰ	品質目標の設定および達成のための活動推進	市場ニーズの高い製品を開発するために、品質目標を設定し、達成するための活動をプロジェクト全体を巻き込み推進する。
Ⅱ	全体テスト計画の策定	テスト網羅を確保（漏れや意図しない重複を回避）するために、全体テスト計画（＝テスト戦略。実施するテストレベルと各テストレベルでのテスト内容）を高位レベルで策定する。
Ⅲ	開発プロセス全体の品質測定および是正勧告	品質をより上流で作りこむために、テストだけでなく開発プロセス全体の品質を測定し、必要に応じて是正勧告を行う。
Ⅳ	品質に応じたテスト内容の変更	プロジェクトの制約（コスト・スケジュール・スコープ）の中で品質を可能な限り高めるために、テスト設計時に「**品質状況に応じてテスト内容を変更できる仕組み**」を構築し、テスト実行時に制御する。
Ⅴ	システムテストの実施	品質の最終確認のために、自らシステムテストを実施する。

これらの役割を果たすべく、**品質保証計画を策定**し
品質保証活動および**システムテストを実施**していきます

モモテツ

チームの位置づけと役割

■品質保証部の役割

私たちは、大きくは下記5つの役割を担います。

I	品質目標の設定および達成のための活動推進	市場ニーズの高い製品を開発するために、品質目標を設定し、達成するための活動をプロジェクト全体を巻き込み推進する。
II	全体テ	するため トレベル 策定する。
III	開発プ 品質測	く開発プ を行う。
IV	品質に応 テスト内	ープ）の 時に「**品質** 構築し、テスト実行時に制御する。
V	システムテストの実施	品質の最終確認のために、自らシステムテストを実施する。

本プレゼンテーションでは、
品質保証計画を策定するプロセス
および
システムテストのテスト設計プロセス
についてご説明します。
（品質保証活動については割愛）

これらの役割を果たすべく、**品質保証計画を策定**し
品質保証活動および**システムテストを実施**していきます

モモテツ

テスト設計の流れ（概要）

テスト計画も
テスト設計の一部

TP テスト計画
品質保証やステークホルダーの安心のために
必要な施策を盛り込んだ計画を策定する。

- 品質保証計画書
- システムテスト計画書

TRA テスト要求分析
ステークホルダーを安心させるための
テスト要求を漏れなく導出する。

- テスト要求一覧

TAD テストアーキテクチャ設計
テスト要求を確認し易いように整理し、
効率的・効果的に確認するための
実施順序や工数配分を決定する。

主要成果物

- テストフレーム
- 因子水準一覧
- テストアーキテクチャ表

TDD テスト詳細設計
テストケースを具体的かつ機械的に
導出できるようにする。

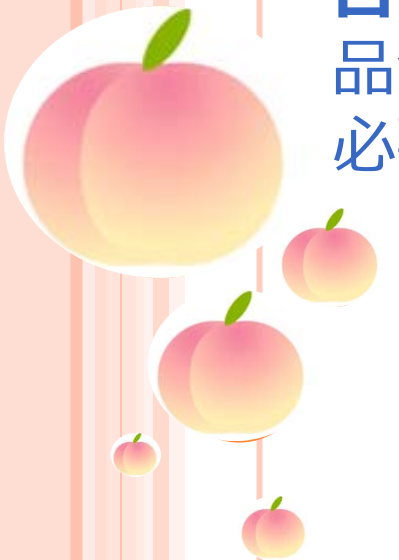
- テスト詳細設計書

モモテツ

TP

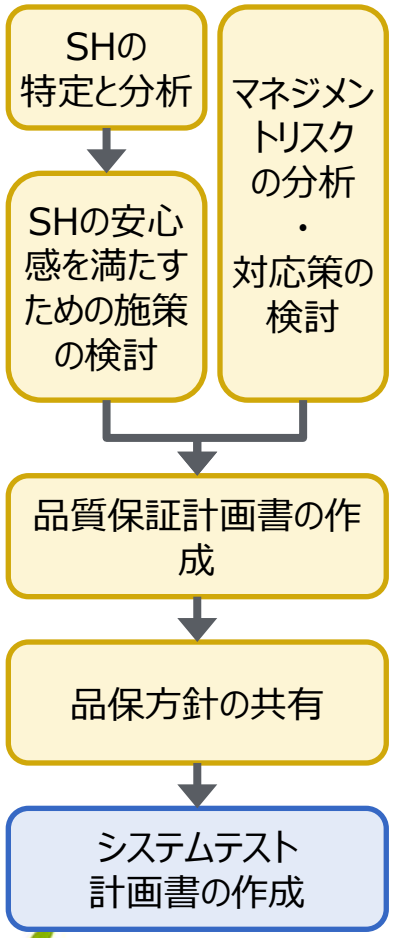
テスト計画

目的：
品質保証やステークホルダーの安心のために
必要な施策を盛り込んだ計画を策定する



テスト計画の概略

▼プロセスフロー



下記を踏まえ、品質保証計画を策定する。
- ソフトウェア品質目標
- リスクと対応策
- 各ステークホルダーを安心させるための施策

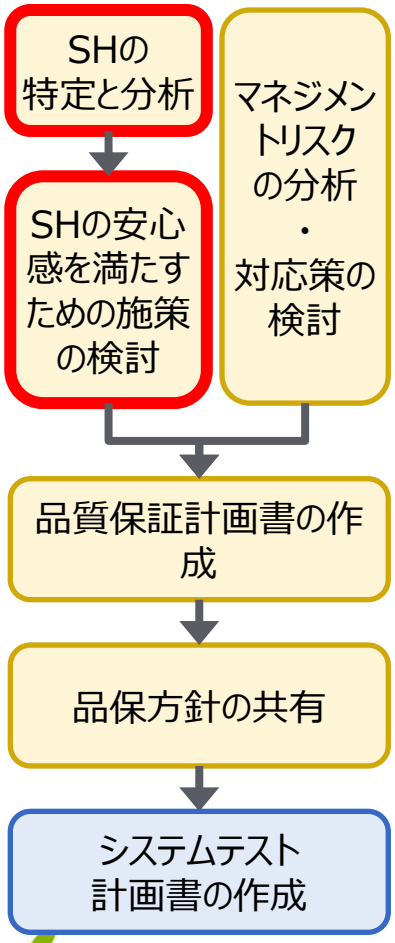
品質保証計画には、全体テスト計画も含む。

各テストレベルの担当チームと
品質保証計画の内容を共有する。

→各テストレベル（システムテストを含む）
の担当チームは、品質保証計画に準拠した
テスト計画を策定する。

ステークホルダーの安心を満たす施策の検討

▼プロセスフロー



各ステークホルダーが
「どのような製品だったら安心できるか」
「どのようなテストが実行されたら安心されるか」
分析し、必要な施策を検討した。

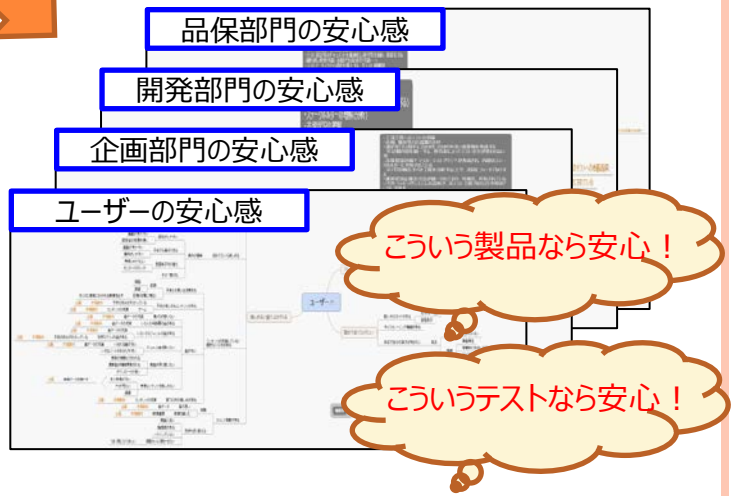
カラオケシステムのステークホルダーを特定後、
ツテを頼ってヒアリングや調査を実施


カラオケ店従業員へのヒアリング


ネットで口コミ調査


開発経験者

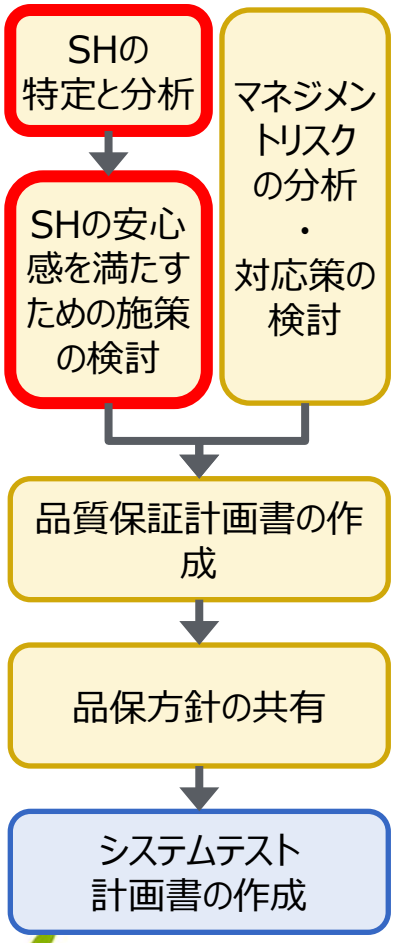
ヒアリング/調査結果を基に
マインドマップで発想を拡大



モモテツ

ステークホルダーの安心を満たす施策の検討

▼プロセスフロー



各ステークホルダーが
「どのような製品だったら安心できるか」
「どのようなテストが実行されたら安心されるか」
分析し、必要な施策を検討した。

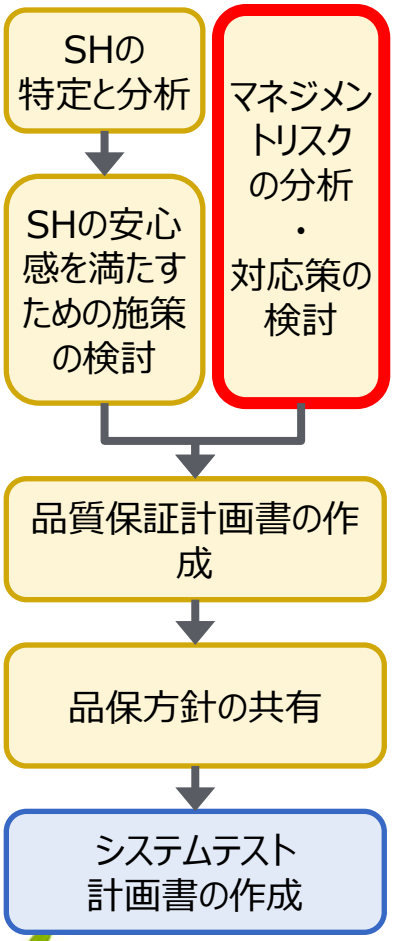
重点施策

- 使用する人視点での観点の盛り込み
- タイムリーなリスク分析と
  ピンポイントなテストの実施
- 品質を判定できる仕組みの導入
- 判定した品質状況に応じ、
  テストの戦略を柔軟に変更できる
  テストアーキテクチャ設計

マインドマップ
のリファインを
重ね導出

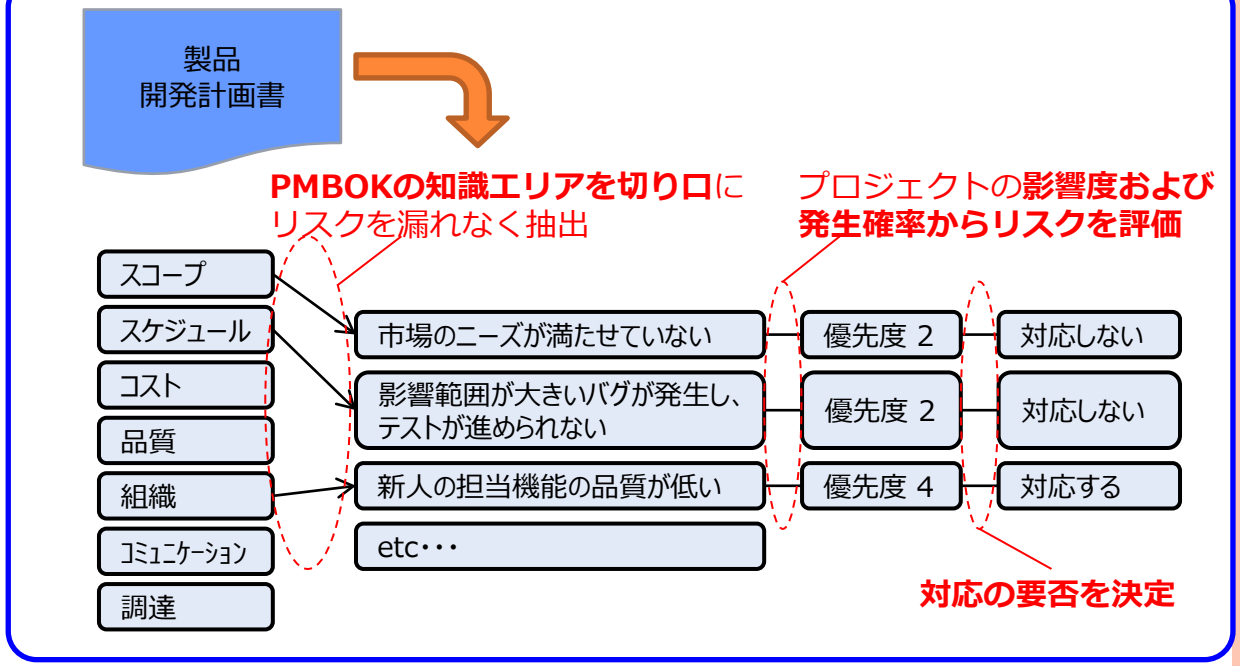
マネジメントリスクの分析と対応策の検討

▼プロセスフロー



プロジェクトの遂行を阻害するリスクに対して
未然に策を講ずるため、
テスト計画時におけるリスクの分析を行った。

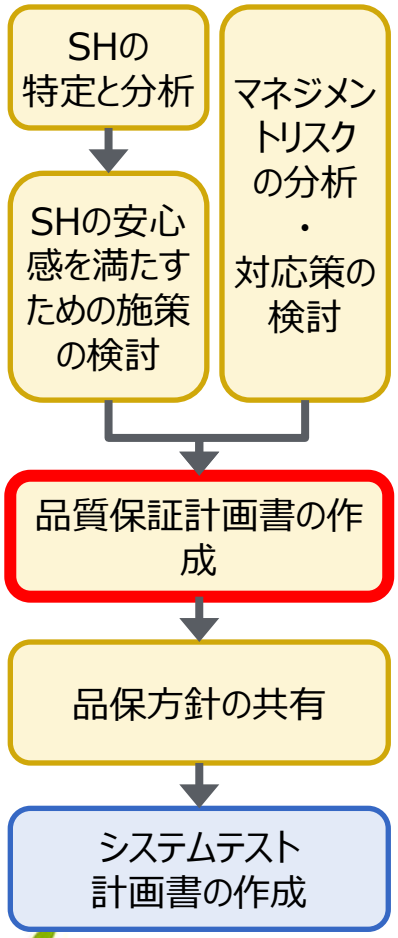
マネジメントリスク
プロジェクト全体のマネジメントへ悪影響を及ぼし得るリスク



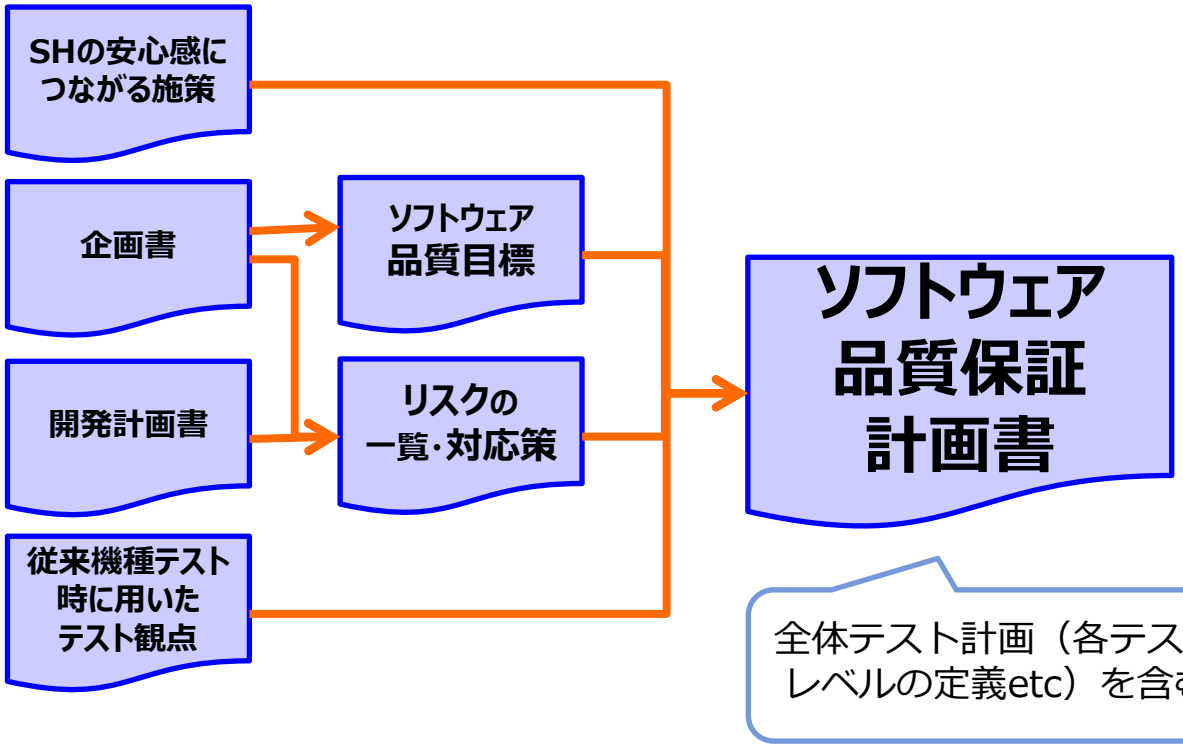
対応が必要なリスクは、対応策を**品質保証計画書に反映させる**

ソフトウェア品質保証計画書の作成

▼プロセスフロー



ステークホルダーの安心感につながる施策や
リスクへの対応策
などのインプットを基に、品質保証計画を策定した。



策定内容の詳細は、次スライド以降で説明する

モモテツ

ソフトウェア品質保証計画の詳細（1/6）

ｉ）品質目標の設定

→ **市場ニーズが高い製品を開発する**狙いで、品質目標を設定。
確実に達成できるよう、目標に応じた施策を策定した。

品質保証部の役割

- I 品質目標の設定および達成のための活動推進
- II 全体テスト計画の策定
- III 開発プロセス全体の品質測定および是正勧告
- IV 品質に応じたテスト内容の変更
- V システムテストの実施

品質目標

性能目標以外

- 楽曲データ、外部機器への完全互換性を保証すること。
- 従来機種に搭載されていたコンテンツを100%搭載すること。
- 従来からのOS変更（オリジナルのリアルタイムOS→LinuxOS）に伴い、LinuxOS環境における全機能の動作を保証すること。
- センター⇔店舗間の通信や各ストレージのセキュリティが強固に保たれていること。

→ 確認漏れが生じないよう、確認担当のテストレベルを定義

性能目標

- 「ハードウェア性能の大幅な向上」や「通信回線の増強」を十分に生かしたソフトウェア性能を実現し、前回のフラグシップモデルを全性能で上回ること。

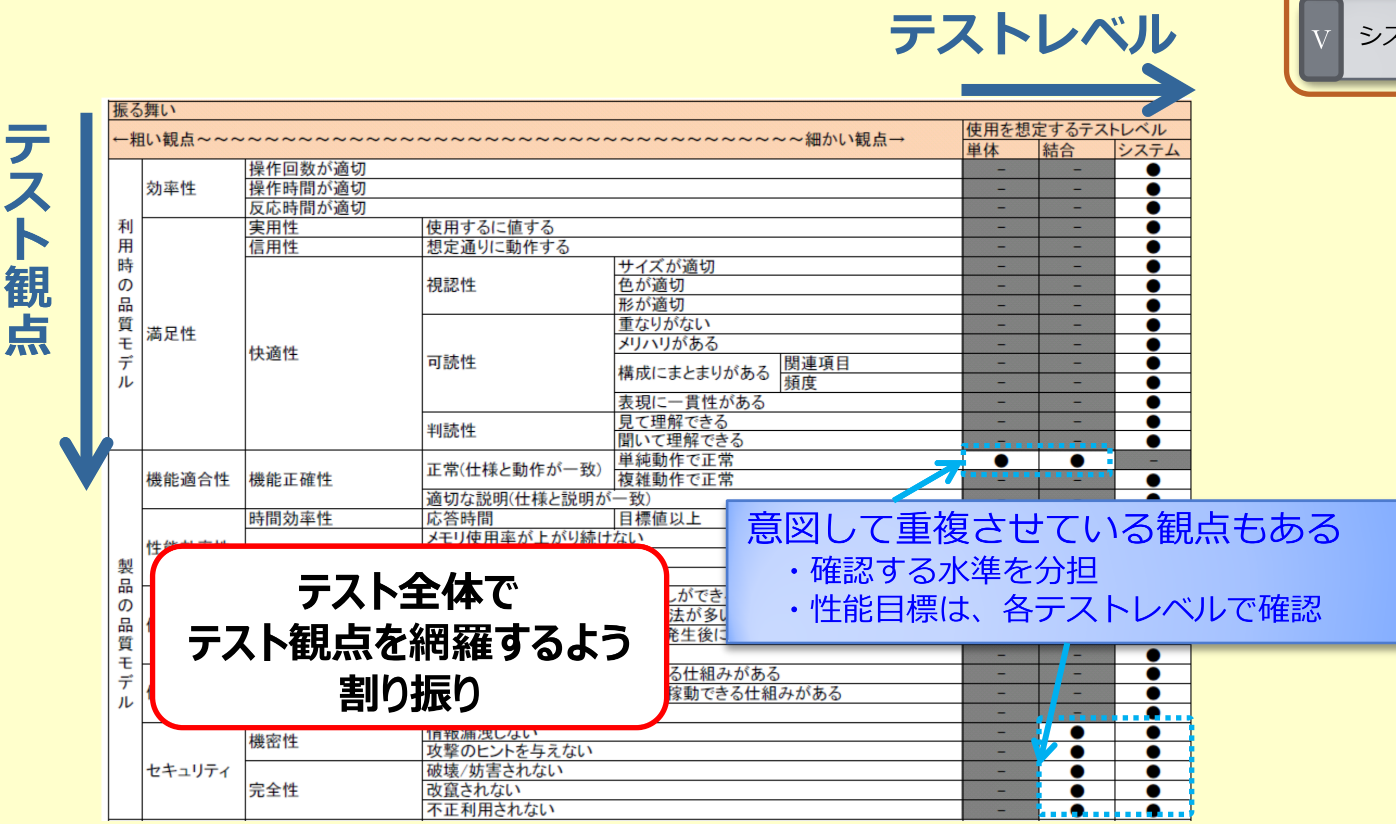
→ 大きな開発手戻りが生じないよう、テストレベル毎に目標値を定め、次のテストレベルへの移行条件に設定

モモテツ

ソフトウェア品質保証計画の詳細（2/6）

ⅱ）各テストレベルにおけるテスト観点のすみ分け

→ **テスト網羅を確保（漏れや 意図しない重複を回避）する**
狙いで、全体テスト計画を策定。
各テストレベルで確認するテスト観点を定めた。
（この時点では、従来機種テスト時に使用したテスト観点を使用）



振る舞い							
←粗い観点～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～細かい観点→					使用を想定するテストレベル		
					単体	結合	システム
利用時の品質モデル	効率性	操作回数が適切			−	−	●
		操作時間が適切			−	−	●
		反応時間が適切			−	−	●
	満足性	実用性	使用するに値する		−	−	●
		信用性	想定通りに動作する		−	−	●
		快適性	視認性	サイズが適切	−	−	●
				色が適切	−	−	●
				形が適切	−	−	●
			可読性	重なりがない	−	−	●
				メリハリがある	−	−	●
				構成にまとまりがある 関連項目	−	−	●
				構成にまとまりがある 頻度	−	−	●
				表現に一貫性がある	−	−	●
			判読性	見て理解できる	−	−	●
				聞いて理解できる	−	−	●
製品の品質モデル	機能適合性	機能正確性	正常(仕様と動作が一致)	単純動作で正常	●	●	−
				複雑動作で正常	−	−	●
			適切な説明(仕様と説明が一致)				●
	性能効率性	時間効率性	応答時間	目標値以上			
			メモリ使用率が上がり続けない				
			…しができ…				
			…法が多し…				
			…発生後に…				
					−	−	●
			…る仕組みがある		−	−	●
			…稼動できる仕組みがある		−	−	●
					−	−	●
	セキュリティ	機密性	情報漏洩しない		−	●	●
			攻撃のヒントを与えない		−	●	●
		完全性	破壊/妨害されない		−	●	●
			改竄されない		−	●	●
			不正利用されない		−	●	●

品質保証部の役割

- Ⅰ 品質目標の設定および達成のための活動推進
- Ⅱ 全体テスト計画の策定
- Ⅲ 開発プロセス全体の品質測定および是正勧告
- Ⅳ 品質に応じたテスト内容の変更
- Ⅴ システムテストの実施

ソフトウェア品質保証計画の詳細（3/6）

iii）品質メトリクスの取得

→ **品質をより上流で作りこむ**狙いで、開発プロセス全体の品質を測定し、必要に応じて是正勧告を行うよう施策を定めた。
プロセスの品質を測定するために、**プロダクトメトリクスだけでなくプロセスメトリクスも取得**し、監視することとした。

品質保証部の役割

- I 品質目標の設定および達成のための活動推進
- II 全体テスト計画の策定
- III 開発プロセス全体の品質測定および是正勧告
- IV 品質に応じたテスト内容の変更
- V システムテストの実施

品質測定の対象は **開発プロセス全体**

要件定義 → 基本設計 → 詳細設計 → 実装 → 単体テスト → 結合テスト → システムテスト

実装工程の例

プロダクトメトリクス	プロセスメトリクス
FP規模	工数の予実比率（実績工数、見積工数）
SLOC	レビュー指摘率（レビュー工数、指摘数）
コメント率	
ネストの深さ（最大）	
サイクロマティック複雑度	

- ●特定の担当者のソースコードが複雑だ
  → 担当者の教育を勧告
- ●レビュー工数が著しく少ない
  → レビュー実施を勧告
- ●レビュー指摘率が低い
  → レビュー体制や運営を確認し、是正勧告

モモテツ

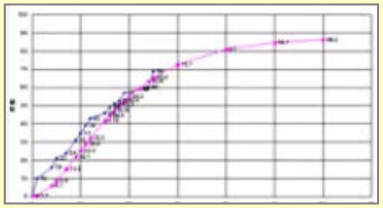
ソフトウェア品質保証計画の詳細（4/6）

1/2

ⅳ）ODC 分析による「テスト成熟度」「テスト網羅度」の判定

品質状況に応じて、テスト内容(テストアーキテクチャ)を柔軟に変更するために、まずは品質状況の見極めに適した分析手法を検討した。

本プロジェクトで検出されるバグ数
⇒561件の想定



信頼度成長曲線(定量的分析)
→バグの特徴や内容はつかめない

品質状況を見極めるためには、発生しているバグの件数だけではなく、バグの内容まで分析する必要がある

しかし・・・

原因分析手法(定性的分析)
→時間がかかる

定量的分析と定性的分析の両方の性質を併せ持つODC分析を採用

品質保証部の役割

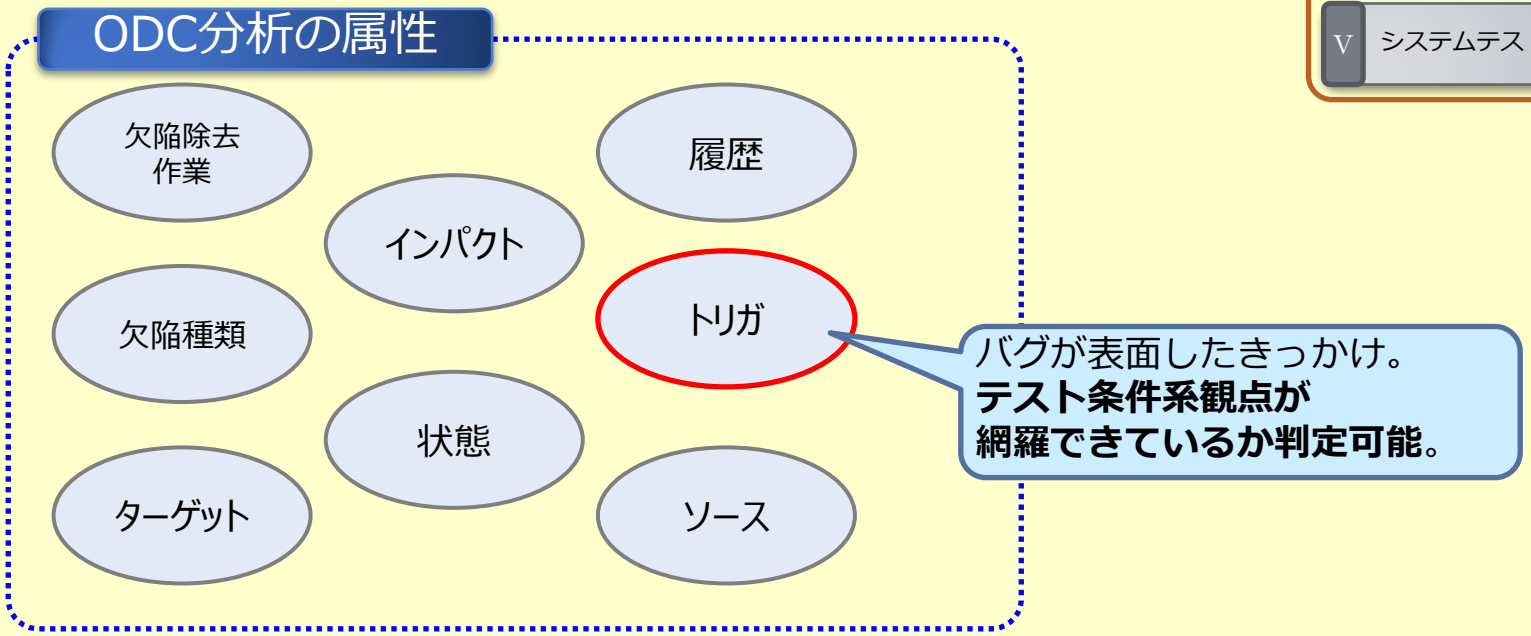
- Ⅰ 品質目標の設定および達成のための活動推進
- Ⅱ 全体テスト計画の策定
- Ⅲ 開発プロセス全体の品質測定および是正勧告
- Ⅳ 品質に応じたテスト内容の変更
- Ⅴ システムテストの実施

ソフトウェア品質保証計画の詳細（5/6）

2/2

ⅳ）ODC 分析による「テスト成熟度」「テスト網羅度」の判定

ソフトウェアの品質状況だけでなく、**テストの成熟度**や**網羅度**を判定するために、**ODC分析のトリガ属性を用いる**こととした。



テストの成熟度と**テストの網羅度**を分析できる「**トリガ**」属性を採用

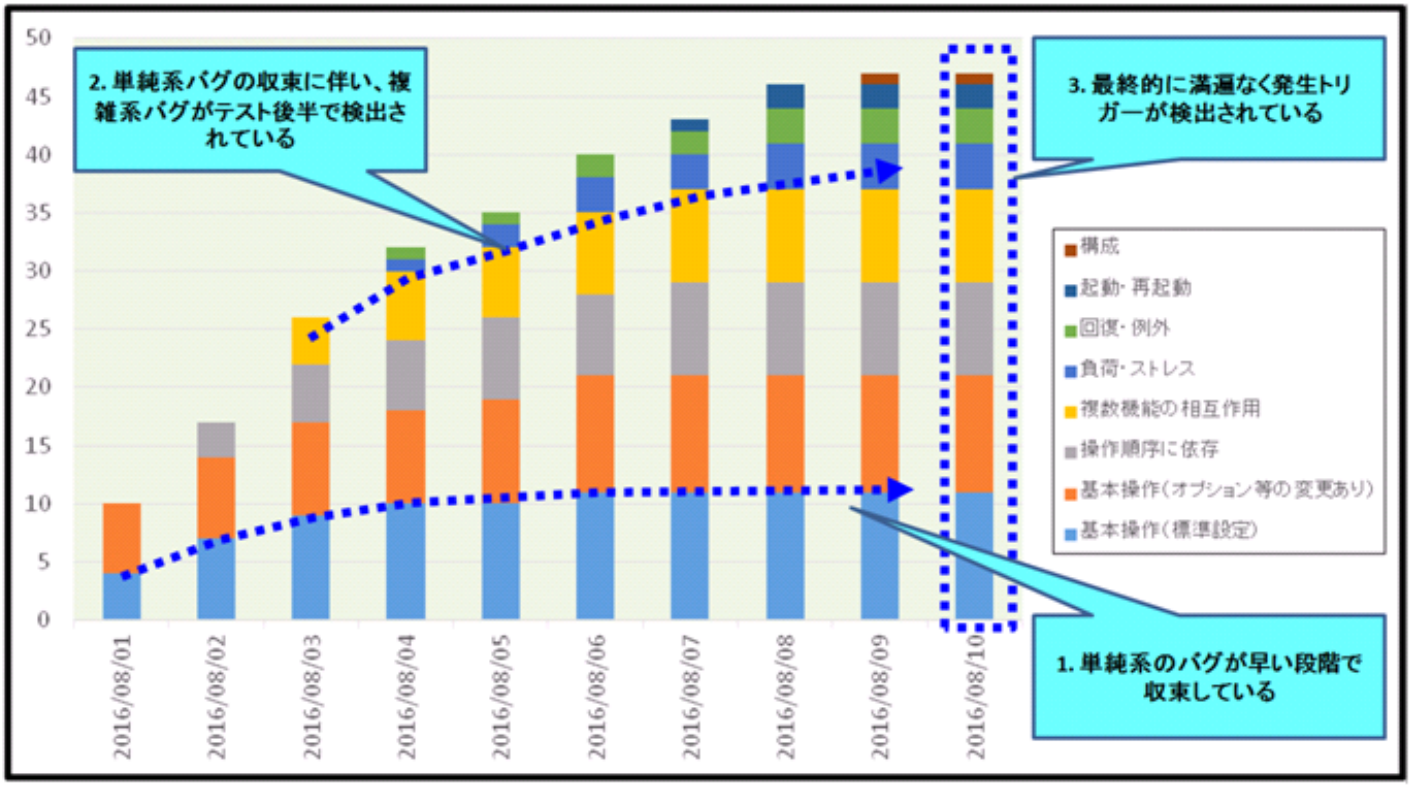
品質保証部の役割

- Ⅰ 品質目標の設定および達成のための活動推進
- Ⅱ 全体テスト計画の策定
- Ⅲ 開発プロセス全体の品質測定および是正勧告
- Ⅳ 品質に応じたテスト内容の変更
- Ⅴ システムテストの実施

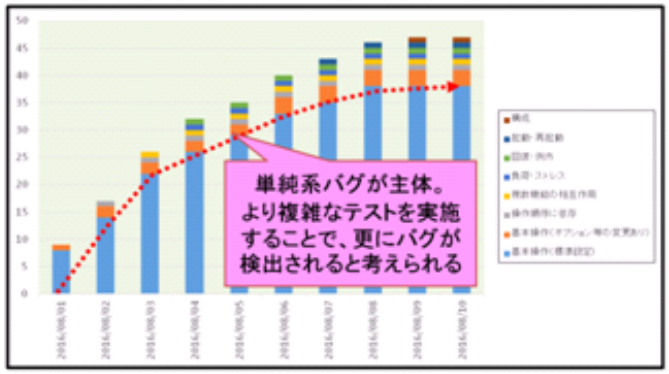
モモテツ

トリガ属性を用いた分析の例

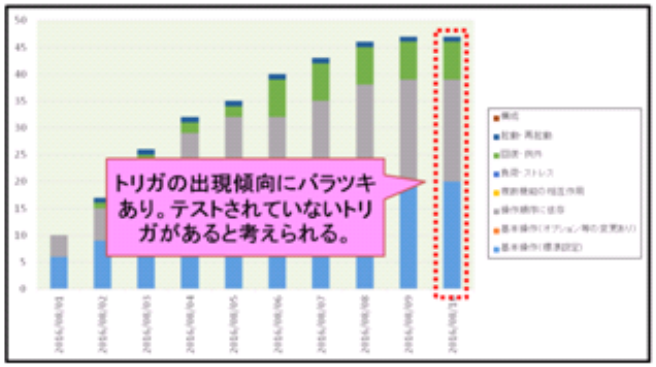
あるべき姿



問題事例①



問題事例②

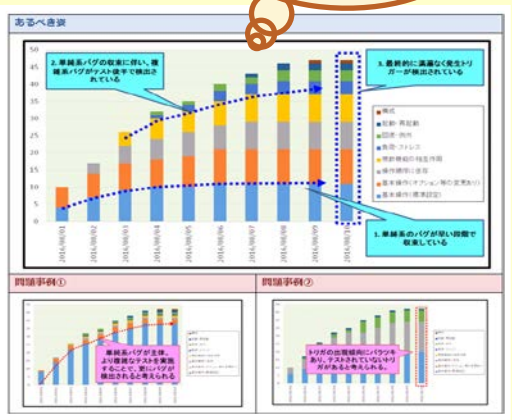


ソフトウェア品質保証計画の詳細（6/6）

ⅳ）品質状況により変更が可能なテストアーキテクチャ

品質状況(バグの収束具合やテストの成熟具合)によって必要なテストは異なるため、必要分を**事前に準備**しておく。

ODC分析の結果によって、
・バグの収束度合い
・テストの網羅/成熟度合い
を把握することができる。

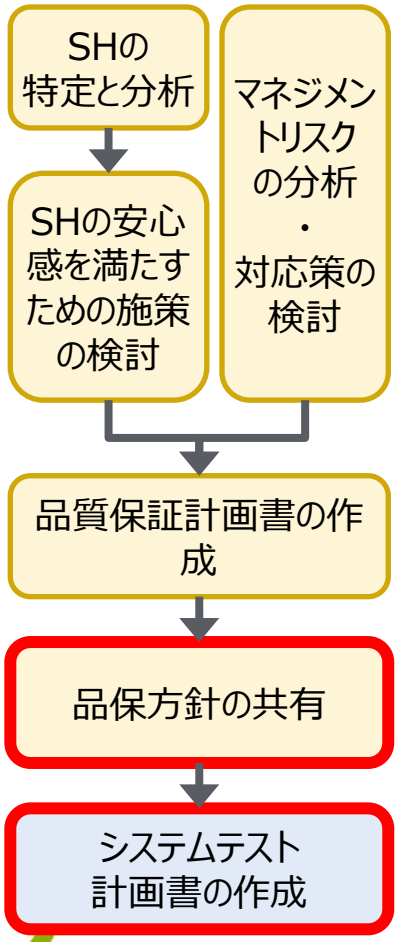


		テストの質	
		優良	不良
バグ収束度合い	収束	テストアーキ①	テストアーキ②
	非収束	テストアーキ③	テストアーキ④

テストアーキテクチャを4通り準備しておき、品質状況に応じて使い分ける

ステークホルダーの安心を満たす施策の検討

▼プロセスフロー

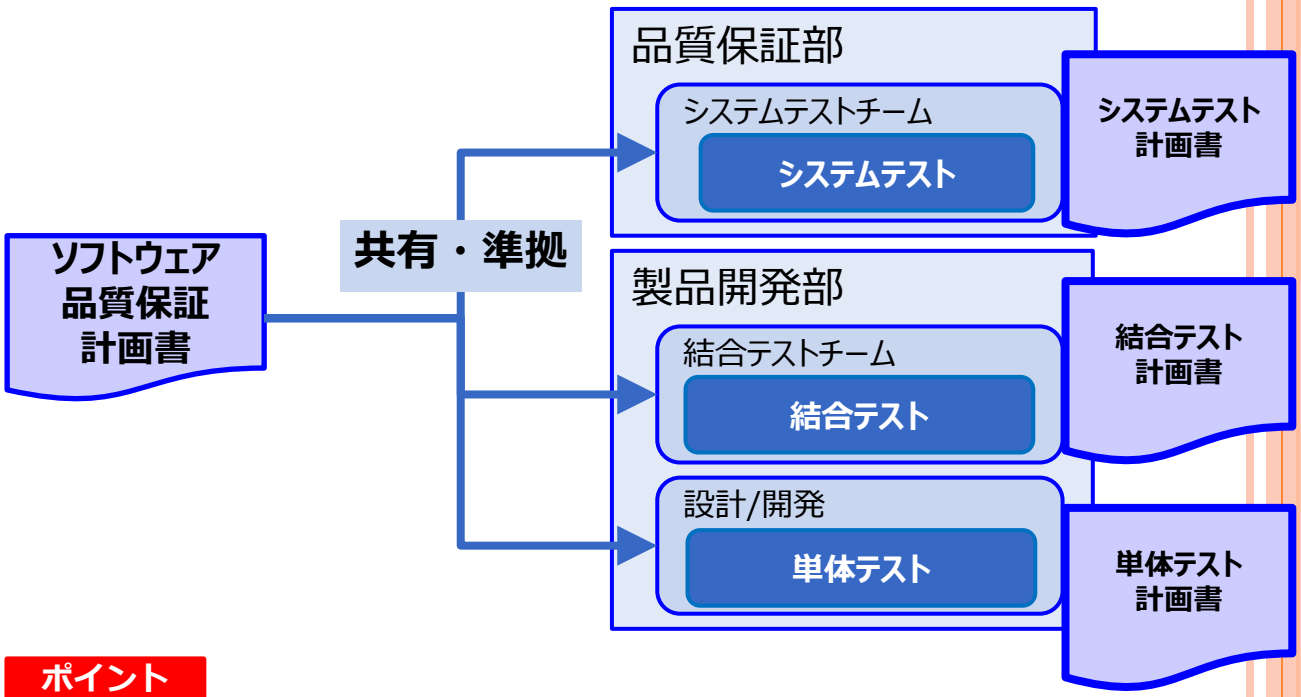


品質保証チームが担当

システムテストチームが担当

品質保証計画を
各テストレベルの実施担当チームと共有した。

システムテストチームなどの各テストレベルの担当チームは
テスト計画書を**品質保証計画に準拠するよう**策定した。



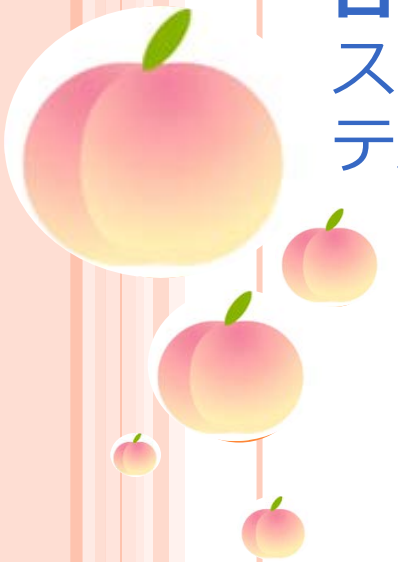
ポイント

・テスト全体での**テスト観点の網羅性を確保**した
・**共通手法による品質分析を可能に**した

モモテツ

TRA テスト要求分析

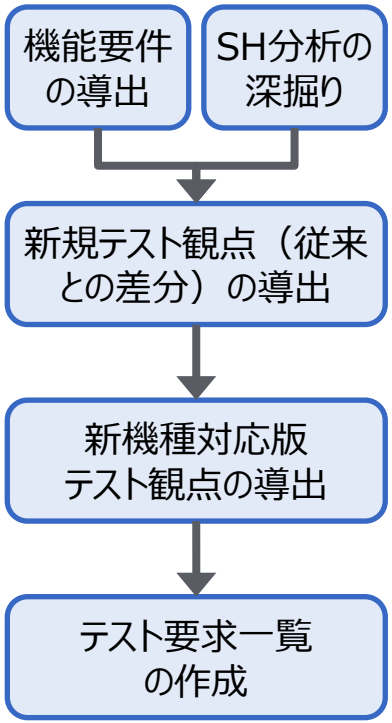
目的：
ステークホルダーを安心させるための
テスト要求を漏れなく導出する



テスト要求分析の概略

システムテストチームが担当

▼プロセスフロー



新機種対応版のテスト観点を導出する。

従来機種のテストで使用したテスト観点に下記の観点を追加する。

- 機種差分（新規要件）から導出した観点
- ステークホルダーの安心感に繋がるテスト観点

テスト観点同士を組み合わせて
テスト要求を導出し、一覧化する。

モモテツ

機能要件の導出

システムテストチームが担当

▼プロセスフロー

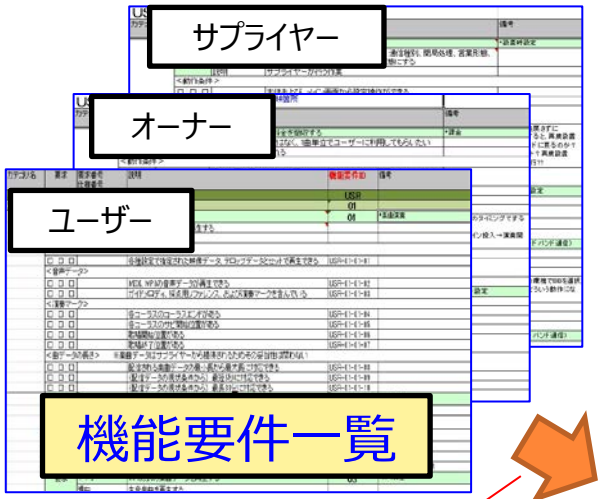
機能要件の導出
SH分析の深掘り
↓
新規テスト観点（従来との差分）の導出
↓
新機種対応版テスト観点の導出
↓
テスト要求一覧の作成

テストベースから機能要件をUSDM形式で抽出した。
（仕様不明点はQAリストにより都度解決）

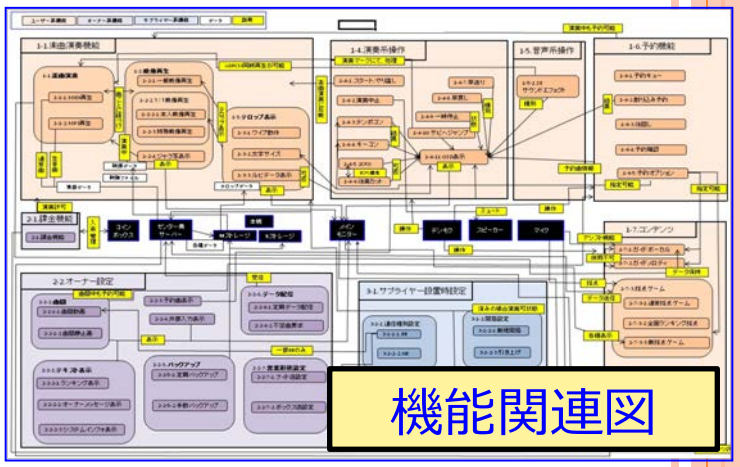
また、機能要件一覧を基に機能関連図を作成した。

テストベース
QAリスト

外部ステークホルダーごとにUSDMで整理（仕様理解も兼ねる）



機能間の関連性を図解



探索的テスト実施時やバグ改修確認時の参考に使用することで、**影響範囲の確認漏れを防ぐ**ことができる

モモテツ

新機種対応版テスト観点の導出

システムテストチームが担当

▼プロセスフロー

機能要件の導出
SH分析の深掘り
↓
新規テスト観点（従来との差分）の導出
↓
新機種対応版テスト観点の導出
↓
テスト要求一覧の作成

今回の対象ソフトウェアは、従来機種のバージョンアップ版であるため、従来機種テスト時に使用したテスト観点一覧を流用した。
⇒欠如していた下記①②の観点を導出しマージすることで
今回の機種に対応したテスト観点一覧を作成した。

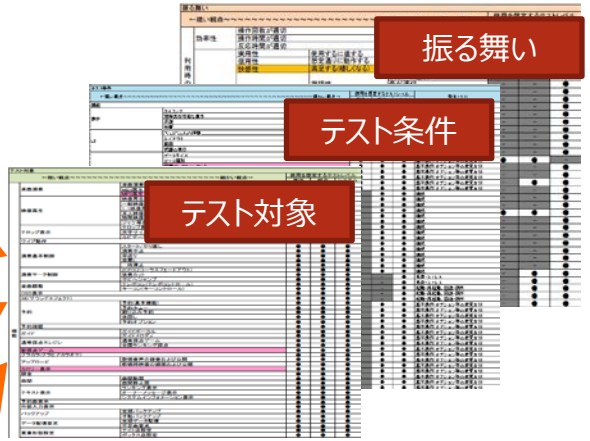
流用元

従来機種テスト時に用いたテスト観点

従来機種との差分

① ステークホルダーの安心感に繋がるテスト観点

② 機種差分（新規要件）から導出したテスト観点



①②合計で　41個の観点を追加した

品質保証計画を策定時は、これらの観点は**考慮していなかった**

テストスコープ変更に伴い、**各テストレベル担当チームの確認担当をすみ分けを見直した**

モモテツ

テスト要求一覧の作成

システムテストチームが担当

▼プロセスフロー

機能要件の導出
SH分析の深掘り
↓
新規テスト観点（従来との差分）の導出
↓
新機種対応版テスト観点の導出
↓
テスト要求一覧の作成

テスト観点（テスト条件/テスト対象/振る舞い）同士の関連付けを行うことで、テストフレームの基盤となるテスト要求を一覧化した。

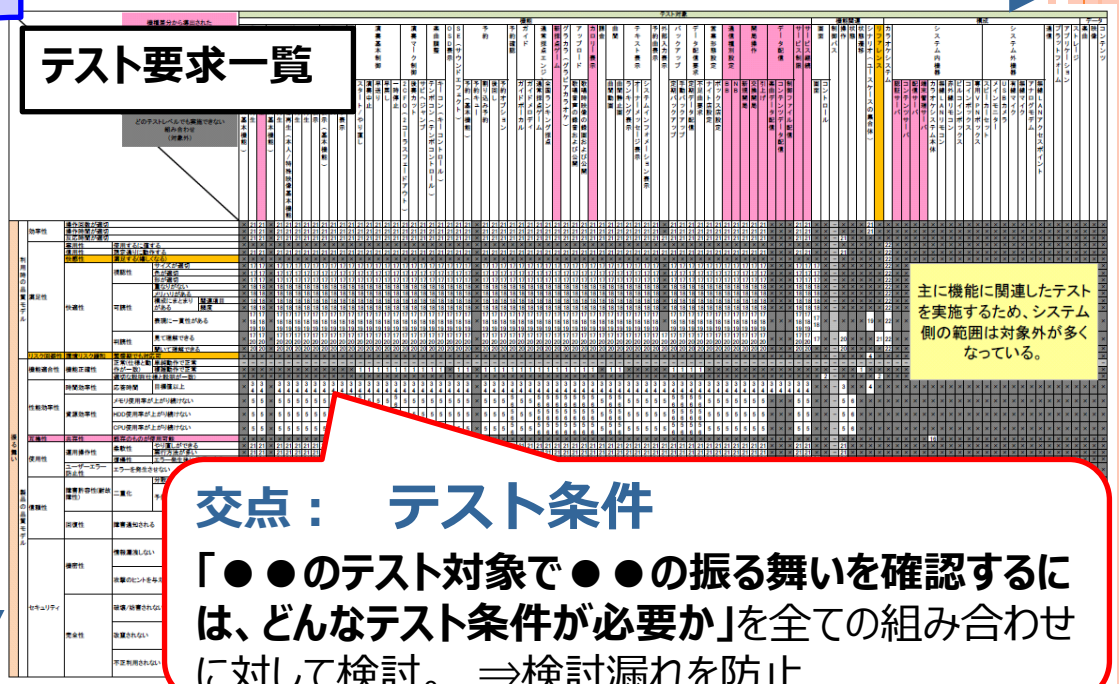
※テスト要求…●●に対して●●したときに●●になる

新機種対応テスト観点

横軸： テスト対象

テスト要求一覧

縦軸：振る舞い



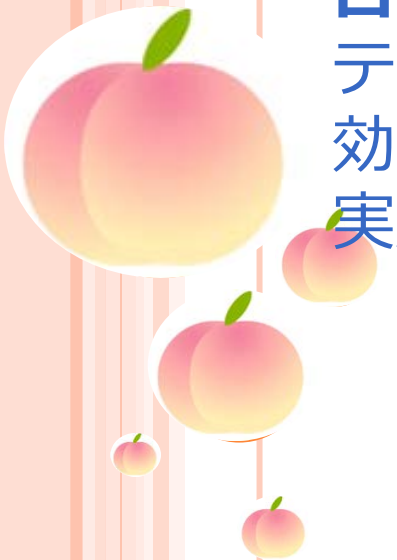
交点： テスト条件

「●●のテスト対象で●●の振る舞いを確認するには、どんなテスト条件が必要か」を全ての組み合わせに対して検討。 ⇒検討漏れを防止

モモテツ

TAD テストアーキテクチャ設計

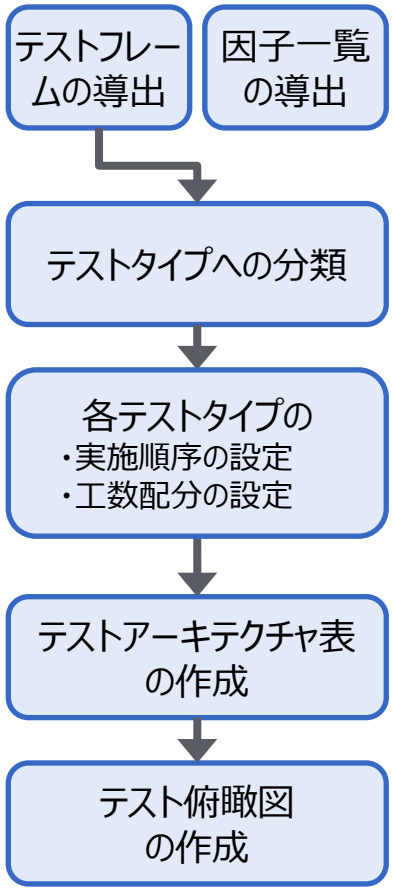
目的：
テスト要求を確認し易いように整理し、
効率的・効果的に確認するための
実施順序や工数配分を決定する



テストアーキテクチャ設計の概略

システムテストチームが担当

▼プロセスフロー



テスト要求一覧を、
テスト詳細設計しやすくするために分解する。

全体像の把握し易さ・管理のし易さUPを
目的に、テストフレームをグルーピングする。

各テストタイプの優先度や工数配分を決める。

テストアーキテクチャの内容を図示し、
テスト内容をプロジェクト内で共有しやすくする。

モモテツ

テストフレーム一覧の作成
テストフレームをテストタイプに分類

システムテストチームが担当

▼プロセスフロー

テストフレームの導出

因子一覧の導出

テストタイプへの分類

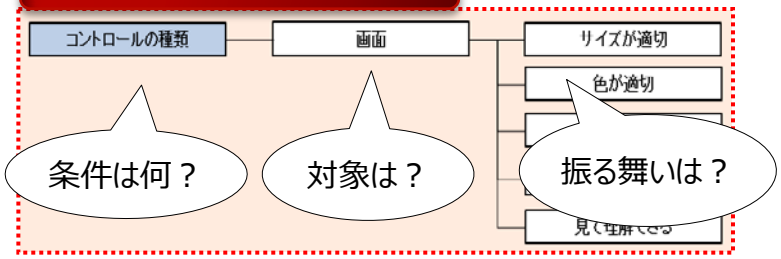
各テストタイプの
・実施順序の設定
・工数配分の設定

テストアーキテクチャ表の作成

テスト俯瞰図の作成

テストアーキテクチャ設計やテスト詳細設計をしやすくするために、テスト要求一覧からテストフレームを作成した。

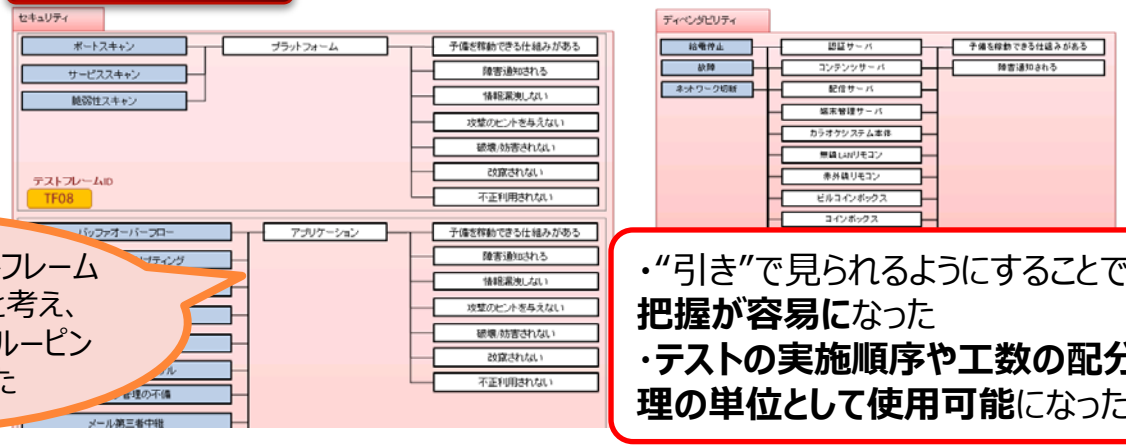
テストフレームの一例



テスト観点（因子）の使われ方を定義することで、**テスト詳細設計担当者によって異なるテストケースが導出されることを防止（一意性を確保）**

さらに
テスト全体像の把握やテストの管理をし易くするために、テストフレームを「テストタイプ」にグループ化した。

テストタイプ



「振る舞い」が同じテストフレームはテストの目的が同じと考え、「振る舞い」を基準にグルーピングし粒度を粗くした

・“引き”で見られるようにすることで**全体像の把握が容易に**なった
・**テストの実施順序や工数の配分、進捗管理の単位として使用可能**になった

モモテツ

因子・水準一覧の導出

システムテストチームが担当

▼プロセスフロー

テストフレームの導出
因子一覧の導出
↓
テストタイプへの分類
↓
各テストタイプの
・実施順序の設定
・工数配分の設定
↓
テストアーキテクチャ表の作成
↓
テスト俯瞰図の作成

因子（「テスト対象」「テスト条件」「振る舞い」）に対する水準を洗い出した。
担当者の思い込み等によるテスト漏れをレビューで防ぐことができるよう、実際にはテストしないであろう水準も含めて一覧化した。

因子・水準一覧

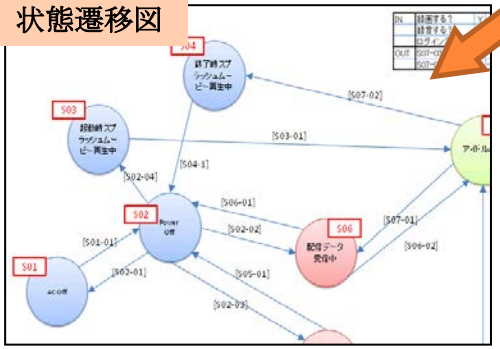
観点分類	テスト観点 ←粗い観点〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜細かい観点→			水準一覧	
テスト条件	機能			《機能一覧を参照》	
	操作	タイミング		遷移中（遷移先に遷移完了しないタイミング）	遷移直後（遷移後、02内のタイミング）
		常時実行可能な操作		背面電源スイッチ	フロント電源スイッチ
		反復		テスト日程に収まる最多回数	10,000回
		放置		テスト日程に収まる最長時間	2週間
	UI	コントロールの種類		フォーム	ボタン
		レイアウト		配置	
		画面		画面全体	
		状態の表示		スタンバイ中	処理中
	機能関連	画面	画面	《画面一覧を参照》	
			コントロール	ボタンやテキストボックス等の全コントロール	
		操作		《操作一覧を参照》	
		状態		《状態・状態遷移一覧を参照》	
		状態遷移		《状態・状態遷移一覧を参照》	

《状態・状態遷移一覧を参照》

実際に確認対象とするか否かは問わず、確認対象となりうる水準を全て列挙。

確認対象の明確化はテスト詳細設計で行う。

状態遷移図



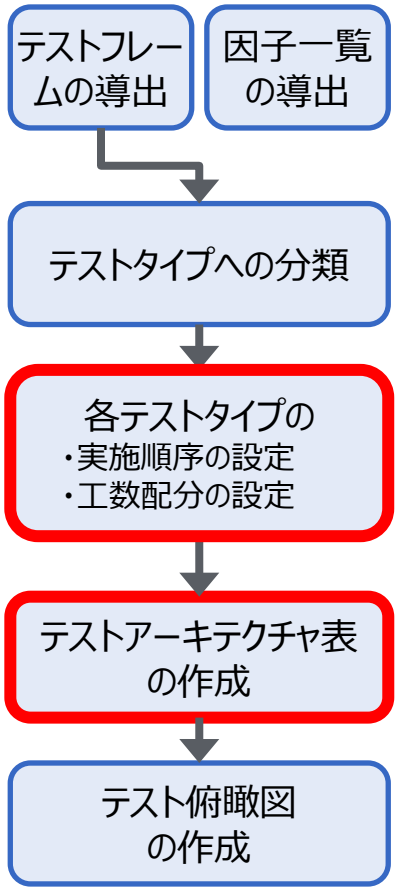
遷移の状態などは、一覧化よりも図を使用したほうがわかりやすい。
目的は水準の洗い出しであるため、**参照のし易さや管理のし易さも考慮し**つつ、その**対象に適した表現方法**を選択した。

モモテツ

テストアーキテクチャの決定

システムテストチームが担当

▼プロセスフロー



下記の情報を基に各テストタイプの実施順序や工数配分を決定した。

- プロダクトリスクの分析結果
- 当該テストタイプで検出されるバグの傾向予測
  （改修コストや、件数etc）
- 当該テストタイプで確認する機能要件数

テストアーキテクチャ表

を決定

			理由
1	工数配分	大きいテストレベルを優先	工数配分が大きいテストレベルは重視しているため、優先度を上げる
2	テスト不良型	探索的 が優先	テストの漏れを効果的に見つけるために探索的テストの優先度を上げる
3	バグ非収束型	1サイクル目回帰 が優先	システムの品質が低いため、優先度の高いテストからもバグが再度検出される可能性を考え、優先度を上げる

サブテストレベル毎のテストケース実施の方針

サブテストレベル	テストケース実施の方針
優先度高	Mustになっているテストケースを全て実施する。
優先度低	Optionになっているテストケースを、テストアーキテクチャのパターンに応じた工数配分(割合)で実施する。(優先度低項目の20% or 70%) ※対象項目を実施する際の予定工数の合計が工数配分(人月)を上回る場合、工数配分(人月)に収まるよう対象項目を絞り実施する。 対象項目を実施する際の予定工数の合計が工数配分(人月)を下回る場合、余った工数はサブテストレベルの実施順序が早いものから順に充てる。
1サイクル目回帰	Mustになっているテストケースを、テストアーキテクチャのパターンに応じた工数配分(割合)で実施する。(優先度高項目の10% or 30% or 40% or 50%) ※対象項目を実施する際の予定工数の合計が工数配分(人月)を上回る場合、工数配分(人月)に収まるよう対象項目を絞り実施する。 対象項目を実施する際の予定工数の合計が工数配分(人月)を下回る場合、余った工数はサブテストレベルの実施順序が早いものから順に充てる。
探索的	テストアーキテクチャのパターンに応じた工数配分(割合)で実施する。(20% or 40% or 50% or 70%)

		1サイクル目	2サイクル目										
テストレベル	工数(人月)	7.00	5.20										
	テストアーキテクチャ名	-	テスト優良・バグ収束型			テスト優良・バグ非収束型			テスト不良・バグ収束型			テスト不良・バグ非収束型		
サブテストレベル		優先度高	探索的	優先度低	1サイクル目回帰	探索的	優先度低	1サイクル目回帰	探索的	優先度低	1サイクル目回帰	探索的	優先度低	1サイクル目回帰
	工数配分(割合)	100%	20%	70%	10%	40%	20%	40%	70%	0%	30%	50%	0%	50%
	工数配分(人月)	7.00	1.04	3.64	0.52	2.08	1.04	2.08	3.64	0.00	1.56	2.60	0.00	2.60
	実施順序	1	2	1	3	2	3	1	1	-	2	1	-	2
テストタイプ工数配分(人月)	ディペンダビリティ	0.85	-	0.53	0.06	-	0.15	0.25	-	-	0.19	-	-	0.32
	ユーザビリティ	1.70	-	1.06	0.13	-	0.30	0.50	-	-	0.38	-	-	0.63
	ストレス	0.41	-	0.25	0.03	-	0.07	0.12	-	-	0.09	-	-	0.15
	タイミング	0.85	-	0.53	0.06	-	0.15	0.25	-	-	0.19	-	-	0.32
	運用互換	0.81	-	-	0.06	-	-	0.24	-	-	0.18	-	-	0.30
	リソースリーク	1.70	-	1.06	0.13	-	0.30	0.50	-	-	0.38	-	-	0.63
	セキュリティ	0.35	-	0.22	0.03	-	0.06	0.10	-	-	0.08	-	-	0.13
	パフォーマンス	0.34	-	-	0.03	-	-	0.10	-	-	0.08	-	-	0.13
	探索的	-	1.04	-	-	2.08	-	-	3.64	-	-	2.60	-	-

実施順序

モモテツ

テスト俯瞰図の作成

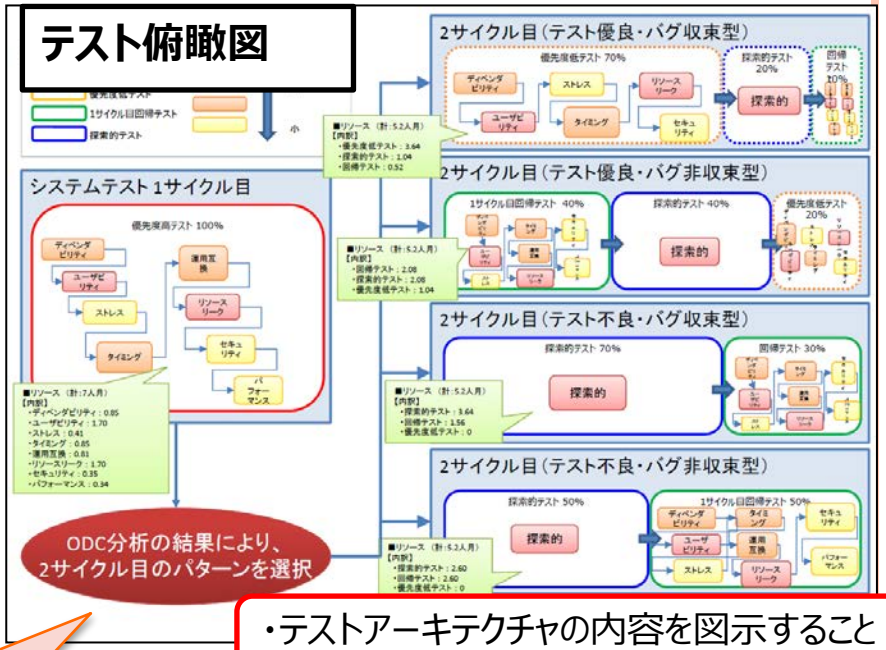
システムテストチームが担当

▼プロセスフロー

テストフレームの導出
因子一覧の導出
↓
テストタイプへの分類
↓
各テストタイプの
・実施順序の設定
・工数配分の設定
↓
テストアーキテクチャ表の作成
↓
テスト俯瞰図の作成

テスト内容についてプロジェクト関係者（特に他チーム・他部門のメンバー）と認識合わせしやすくするため、テストアーキテクチャの内容を一目で把握できるようなテスト俯瞰図を作成した。

テスト
アーキテクチャ
表



品質状況に応じた
分岐(4パターン)を実現

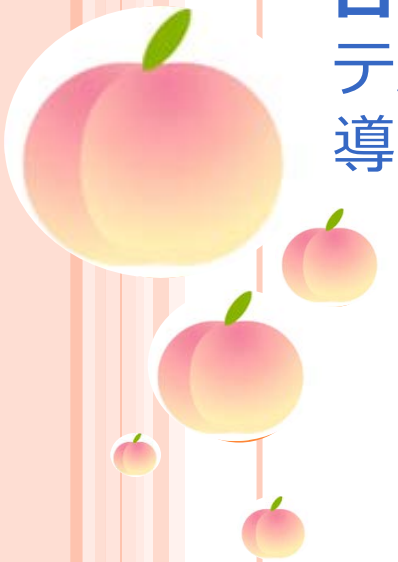
・テストアーキテクチャの内容を図示することで**一目で把握できるようになり、認識合わせをし易く**なった。
・実施内容や順序に加え、**工数も併記し、各テストタイプのボリューム感も把握できる**ようにした。

モモテツ

TDD

テスト詳細設計
（探索的テスト以外）

目的：
テストケースを具体的かつ機械的に
導出できるようにする



テスト対象水準の選定基準の設定

テストアーキ設計で導出した
テストフレーム＋因子・水準一覧
を基に、**機械的に**入力

しかし、全水準を
確認するのは
現実的ではない

選定基準により
テストする水準を絞り込み

テスト詳細設計書

率(使いやすさ)のテスト ／ sf (satisfaction) :満足度のテスト

マスター基準
・従来機種に対するテストで確認実績がない水準を
Must、確認実績がある水準をOptionとする。

マスター基準と同じ基準を採用する場合は「-」を、
異なる基準を採用する場合には具体的な基準を記述する。

テストケース	対応テストフレームID	分類	因子		全水準	テスト対象水準		
						選定基準	Must	Option
USR_ev03 ユーザーが操作するにあたり、各所に備えられたコントロールに一貫性があるか確認する	TF17	テスト条件	UI	コントロールの種類	・フォーム ・ボタン ・テキストボックス ・テキストラベル ・アイコン ・プログレスバー	-	・なし	・フォーム ・ボタン ・テキストボックス ・テキストラベル ・アイコン ・プログレスバー ※全ての水準を確認するが、機能内で共通で使用されているコントロールについては機能内で1回の確認とする。
			ユーザー視点テストの方式	システムテスト担当者によるOK/NG判定方式	・システムテスト担当者が1名で実施する ・システムテスト担当者が複数名で実施する	定量的な評価ができずテスト担当者の主観的な判断になってしまうリスクがあるため、複数名の議論による判断結果を採用する。 尚、この水準は従来機種テストで確認実績があるためOptionに設定する。	・なし	・システムテスト担当者が複数名で実施する
		テスト対象振る舞い	機能関連	画面	《画面一覧を参照》	-	・従来機種のテストで確認実績のない画面	・従来機種のテストで確認実績のある画面
			利用時の品質モデル	満足性	・機能/画面内で表現が統一されていること。 ・製品内で表現が統一されていること。 ・シリーズ内で表現が統一されていること。 ・全製品内で表現が統一されていること。 ・客観的に見て、見やすく統一されていること。	-	・なし	・機能/画面内で表現が統一されていること。 ・製品内で表現が統一されていること。 ・シリーズ内で表現が統一されていること。 ・全製品内で表現が統一されていること。 ※「客観的に」が含まれている水準については、「USR_sf01」のみで確認する為、対象外とする。
			快適性	画面				
			可読性	画面				
			表現に一貫性がある	画面				

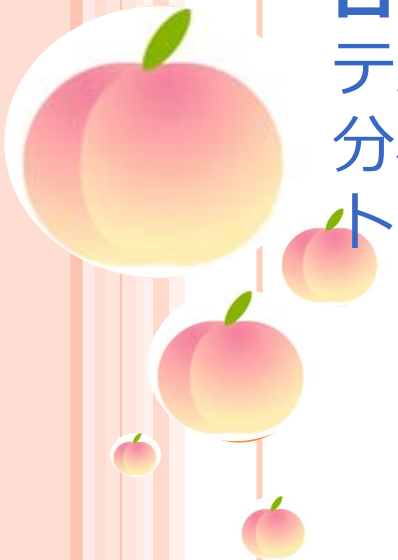
テスト
フレーム一覧

因子水準
一覧

選定基準により
- **テストケースの爆発を防ぎ、効果の高いテストケースの実施**を実現できた
- その水準を**実施する/しない理由を明確**にできた

モモテツ

TDD´探索的テスト方針



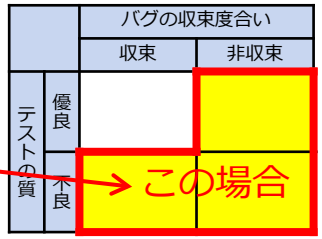
目的：
テスト実行結果等からタイムリーにリスクを分析し、リスクに対してピンポイントなテストを実施する

探索的テスト導入の経緯

テストの実行期間中に想定されるケース…

品質状況（ソフトの質・テストの質のいずれかまたは両方）が劣悪

スケジュールの変更が困難



市場への投入時期を逃すと
製品価値が下がり、
売上に結び付かない懸念

下記のいずれか、
または複数を組み合わせて対応

【コストの変更】
要員を追加投入する

【スコープの変更】
品質が伴わない機能を削除する

リスクベース且つピンポイントな
テストを実施する
（制約の中で、品質の最大化を狙う）

探索的テスト導入の経緯

テストの実行期間中に想定されるケース…

品質状況（ソフトの質・テストの質のいずれかまたは両方）が劣悪

スケジュールの変更が困難

下記のいずれか、
または複数を組み合わせて対応

【コストの変更】
要員を追加投入する

【スコープの変更】
品質が伴わない機能を削除する

リスクベースかつピンポイントなテストを実施する
（制約の中で、品質の最大化を狙う）

リスクをタイムリーに分析でき、
直後のテストで活用できる必要がある

**「短いスパンで
テスト設計とテスト実行を繰り返す」
特徴を持つ探索的テストを導入することに**

探索的テスト導入の経緯

テストの実行期間中に想定されるケース…

品質状況（ソフトの質・テストの質のいずれかまたは両方）が劣悪

スケジュールの変更が困難

下記のいずれか、
または複数を組み合わせて対応

【コストの変更】
要員を追加投入する

【スコープの変更】
品質が伴わない機能を削除する

リスクベースかつピンポイントな
テストを実施する
（制約の中で、品質の最大化を狙う）

リスクをタイムリーに分析でき、
直後のテストで活用できる必要がある

**「短いスパンで
テスト設計とテスト実行を繰り返す」
特徴を持つ探索的テストを導入することに**

リーダーがテスターに
「テスト条件系観点×テスト対象系観点」
の単位で指示が出せる
制御された探索的テスト
を実現させる方針を策定する

探索的テストの実施方針の設計

探索的テストの方針を、以下の通り定めた。

1	工数ボリュームを、テストアーキ設計で定めたパターンに応じて決定 目的：品質状況に応じた探索的テストの使い分け
2	リスクベース且つピンポイント性の高いテストを実行 目的：市場バグのすり抜けを低減
3	30分から2時間程度の細かなセッションに分割して、テスト実行 目的：テスト進捗の管理/効率的な進行
4	重点項目は、複数名で、同じ「テスト条件×機能」を確認する 目的：発生頻度が少ないバグの流出防止 目的：個人差によるテスト漏れリスクの低減
5	各自が行ったテストの内容は一覧化し、確認できるようにする 目的：テスト履歴の管理、 目的：テスト内容を共有による探索箇所の発想拡大

探索的テストの実施方針の設計

探索的テストの方針

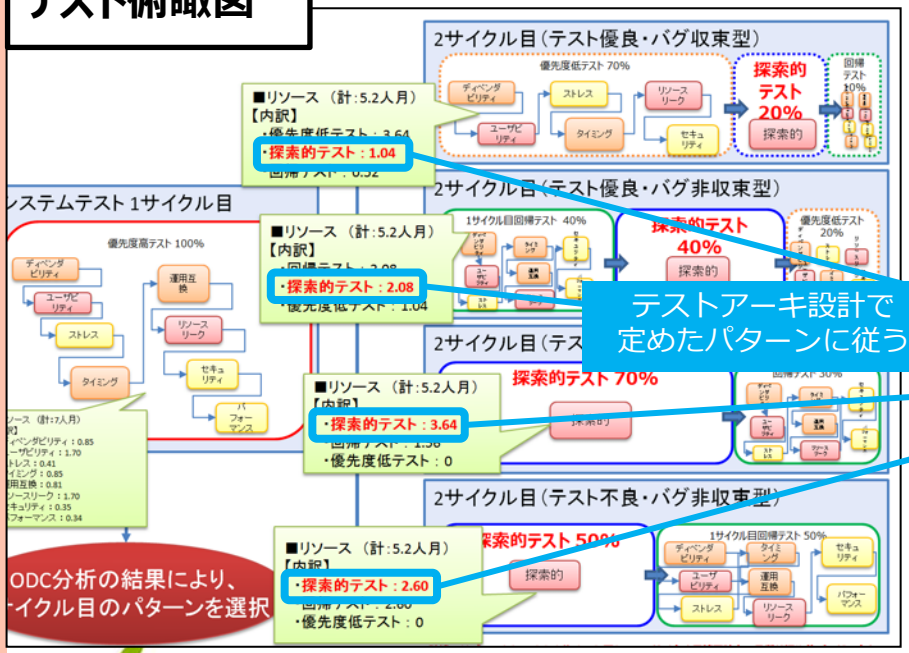
1	工数ボリュームを、テストアーキ設計で定めたパターンに応じて決定 目的：品質状況に応じた探索的テストの使い分け
2	リスクベース且つピンポイント性の高いテストを実行 目的：市場バグのすり抜けを低減
3	30分から2時間程度の細かなセッションに分割して、テスト実行 目的：テスト進捗の管理/効率的な進行
4	重点項目は、複数名で、同じ「テスト条件×機能」を確認する 目的：発生頻度が少ないバグの流出防止 目的：個人差によるテスト漏れリスクの低減
5	各自が行ったテストの内容は一覧化し、確認できるようにする 目的：テスト履歴の管理、 目的：テスト内容を共有による探索箇所の発想拡大

探索的テストの方針１

1	工数ボリュームを、テストアーキ設計で定めたパターンに応じて決定 目的：品質状況に応じた探索的テストの使い分け

品質状況によってリスクの大きさ（残存バグ数、デグレード数）は異なるため
探索的テストに割り当てるリソースを変動させる。

テスト俯瞰図



テストアーキ設計で
定めたパターンに従う

探索的テストに割りあてるリソース

		バグ収束度合い	
		収束	非収束
テストの質	優良	テストアーキ① 1.04人月	テストアーキ② 2.08人月
	不良	テストアーキ③ 3.64人月	テストアーキ④ 2.60人月

テストアーキ①では、
ステアリングコミッティによる審議により「実施不要」
と判断された場合には実施しない。

モモテツ

探索的テストの実施方針の設計

探索的テストの方針

1	工数ボリュームを、テストアーキ設計で定めたパターンに応じて決定 目的：品質状況に応じた探索的テストの使い分け
2	リスクベース且つピンポイント性の高いテストを実行 目的：市場バグのすり抜けを低減
3	30分から2時間程度の細かなセッションに分割して、テスト実行 目的：テスト進捗の管理/効率的な進行
4	重点項目は、複数名で、同じ「テスト条件×機能」を確認する 目的：発生頻度が少ないバグの流出防止 目的：個人差によるテスト漏れリスクの低減
5	各自が行ったテストの内容は一覧化し、確認できるようにする 目的：テスト履歴の管理、 目的：テスト内容を共有による探索箇所の発想拡大

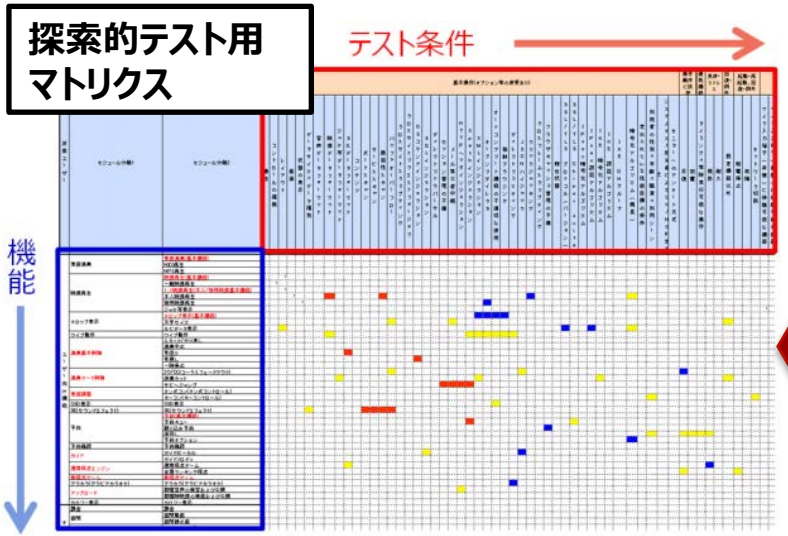
探索的テストの方針２

2	リスクベース且つピンポイント性の高いテストを実行 目的：市場バグのすり抜けを低減

各情報を基にリスク分析を行うことで、
リスクの高い「テスト条件」「機能」を割り出し、効率的なバグ検出を狙う。

①機能×テスト条件をマトリクス化する。

②リーダーが、リスクを分析し、
リスクが高い「機能×テスト条件」について
マトリクスのセルに色を付ける。



色付け

リスク分析

*バグが検出されていない**機能**はないか
*バグが検出されていない**トリガ**はないか

*バグの改修情報から影響を受けそうな機能や修正規模の大きな箇所がないか

*ソースコードレビューが十分ではないなど、開発が不安視している機能はないか

ODC 分析結果

バグ 改修履歴

開発担当者の声

モモテツ

探索的テストの実施方針の設計

探索的テストの方針

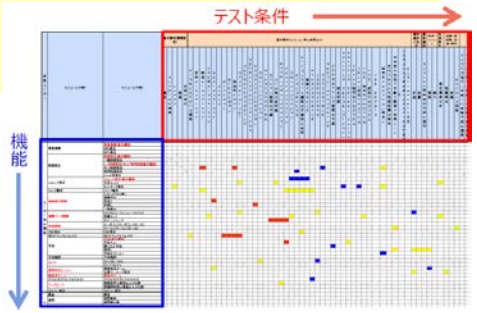
1	工数ボリュームを、テストアーキ設計で定めたパターンに応じて決定 目的：品質状況に応じた探索的テストの使い分け
2	リスクベース且つピンポイント性の高いテストを実行 目的：市場バグのすり抜けを低減
3	30分から2時間程度の細かなセッションに分割して、テスト実行 目的：テスト進捗の管理/効率的な進行
4	重点項目は、複数名で、同じ「テスト条件×機能」を確認する 目的：発生頻度が少ないバグの流出防止 目的：個人差によるテスト漏れリスクの低減
5	各自が行ったテストの内容は一覧化し、確認できるようにする 目的：テスト履歴の管理、 目的：テスト内容を共有による探索箇所の発想拡大

探索的テストの方針３

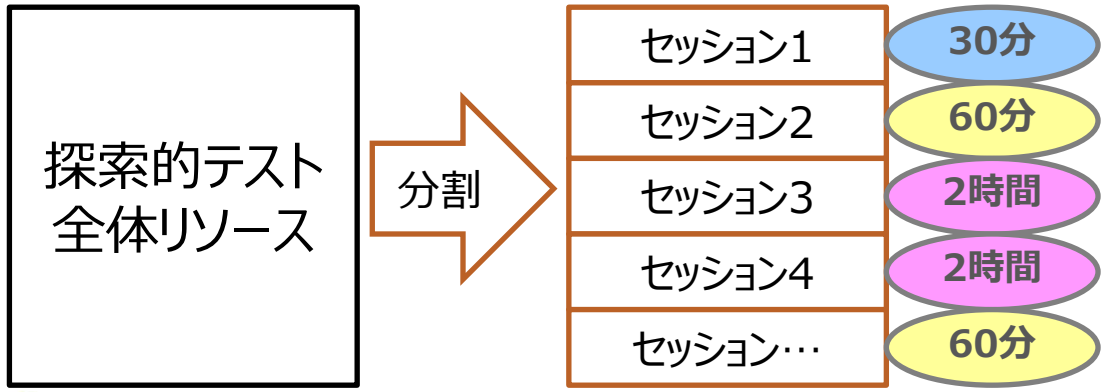
3	30分から2時間程度の細かなセッションに分割して、テスト実行 目的：テスト進捗の管理/効率的な進行

探索的テストに割り当てられたリソースを「セッション」という単位に分割することで
以下の効果を狙う。

- （テスター） 1つの「テスト条件×機能」の**テストの冗長化を防ぐ**
- （テスター） テストに**メリハリをつけさせる**
- （リーダー） **テスト進捗を管理しやすく**する



■ **色の定義（優先、重点）**



- 各リスク分析結果を基に**総合的なリスク**を考慮し、3段階で定義
- リスクに応じ、**1セッションあたりに掛ける工数**を変動させる（あくまで目安）。
  - 重点箇所(赤)　…2時間
  - それ以外(黄青) …30分～60分

モモテツ

探索的テストの実施方針の設計

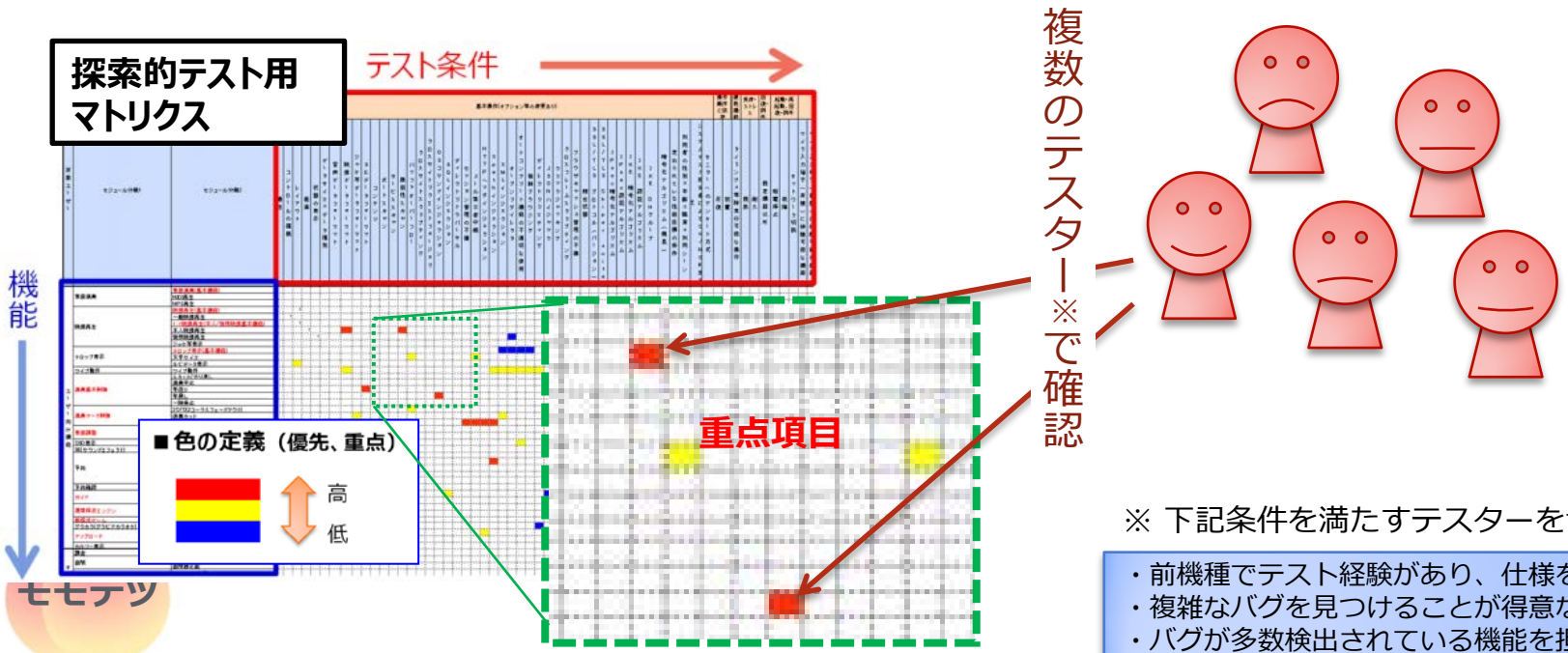
探索的テストの方針

1	工数ボリュームを、テストアーキ設計で定めたパターンに応じて決定 目的：品質状況に応じた探索的テストの使い分け
2	リスクベース且つピンポイント性の高いテストを実行 目的：市場バグのすり抜けを低減
3	30分から2時間程度の細かなセッションに分割して、テスト実行 目的：テスト進捗の管理/効率的な進行
4	重点項目は、複数名で、同じ「テスト条件×機能」を確認する 目的：発生頻度が少ないバグの流出防止 目的：個人差によるテスト漏れリスクの低減
5	各自が行ったテストの内容は一覧化し、確認できるようにする 目的：テスト履歴の管理、 目的：テスト内容を共有による探索箇所の発想拡大

探索的テストの方針４

4	重点項目は、複数名で、同じ「テスト条件×機能」を確認する 目的：発生頻度が少ないバグの流出防止 目的：個人差によるテスト漏れリスクの低減

- 同様のテストを複数回行うことで、**再現性の低いバグの流出リスクを低減**させる。
- 探索的テストは確認内容に**個人差が生じやすいテスト**であるため、**複数名で確認することにより信頼度を向上**させる。



※ 下記条件を満たすテスターを含む複数名

・前機種でテスト経験があり、仕様を熟知している人
・複雑なバグを見つけることが得意な人
・バグが多数検出されている機能を担当していた人

探索的テストの実施方針の設計

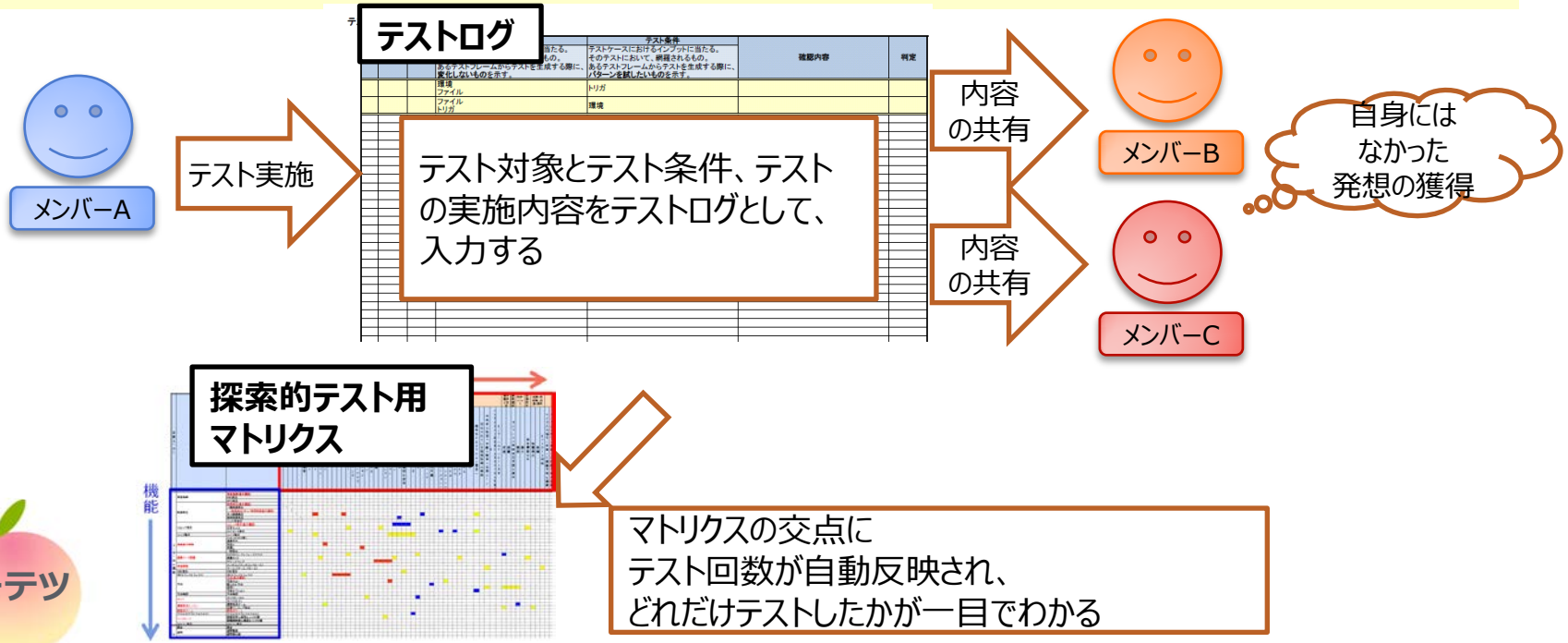
探索的テストの方針

1	工数ボリュームを、テストアーキ設計で定めたパターンに応じて決定 目的：品質状況に応じた探索的テストの使い分け
2	リスクベース且つピンポイント性の高いテストを実行 目的：市場バグのすり抜けを低減
3	30分から2時間程度の細かなセッションに分割して、テスト実行 目的：テスト進捗の管理/効率的な進行
4	重点項目は、複数名で、同じ「テスト条件×機能」を確認する 目的：発生頻度が少ないバグの流出防止 目的：個人差によるテスト漏れリスクの低減
5	各自が行ったテストの内容は一覧化し、確認できるようにする 目的：テスト履歴の管理、 目的：テスト内容を共有による探索箇所の発想拡大

探索的テストの方針 5

5	各自が行ったテストの内容は一覧化し、確認できるようにする 目的：テスト履歴の管理、 目的：テスト内容を共有による探索箇所の発想拡大

- 他メンバーのテスト内容を共有することで、**探索視点の発想拡大**。
- 探索的テストの履歴を残すことで、**バグが検出されなかった場合でも**、「テストしてないからバグが出なかった」のではなく**「テストしたけどバグが出なかった」ことの裏付け**とできる。



探索的テストの実施フロー

探索的テスト期間中は、①～⑥を日々繰り返し実施する

リーダー作業
メンバー作業
手順

バグDB

ODC
分析結果

バグ
改修履歴

開発担当者
の声

①各入力ファイルから
リスク分析を行う

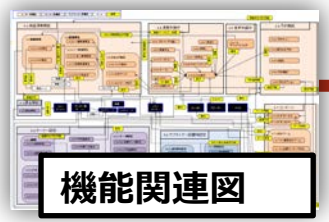
リーダー

②分析結果を反映
マトリクスの交点
(テスト条件×機能)
を色付け

探索的テスト用
マトリクス

③メンバーへ
指示出し

⑥実施したセッション数
がカウントされる



参考

機能関連図

④**他のメンバーのテスト**
内容を参考に、**指示され**
た「テスト条件×機能」の
テストを行う

メンバー

自動反映

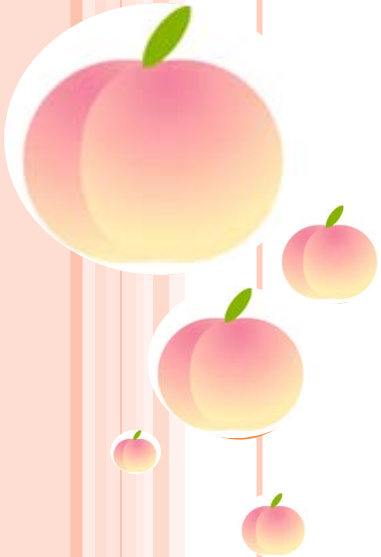
テストログ

テスト対象
（カラオケシステム）

⑤**有識者（リーダー、ベテラン）とテスト内容をレビュー**した上で
テストを実施し、**内容・結果をテストログ**に入力する。

モモテツ

まとめ

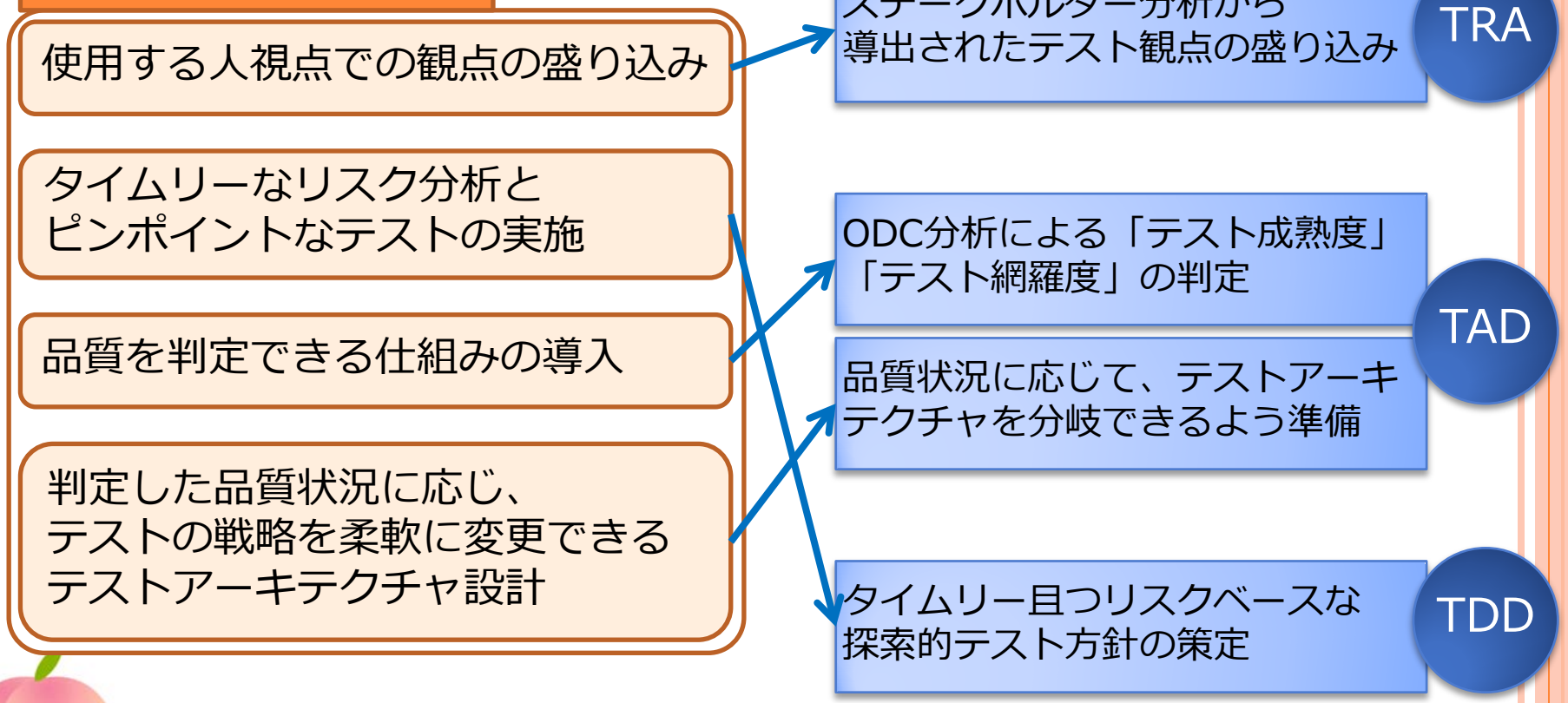


まとめ

ステークホルダーの安心感に繋がる施策を盛り込むことができた

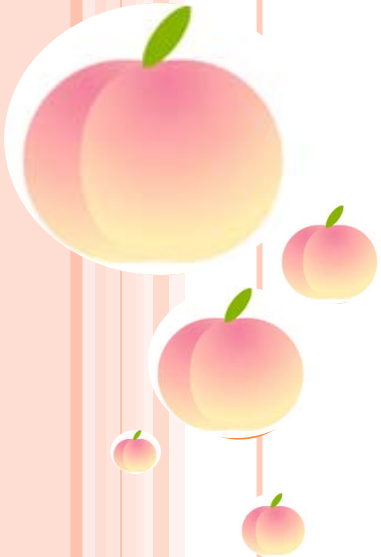
重点施策

- 使用する人視点での観点の盛り込み
- タイムリーなリスク分析と
  ピンポイントなテストの実施
- 品質を判定できる仕組みの導入
- 判定した品質状況に応じ、
  テストの戦略を柔軟に変更できる
  テストアーキテクチャ設計



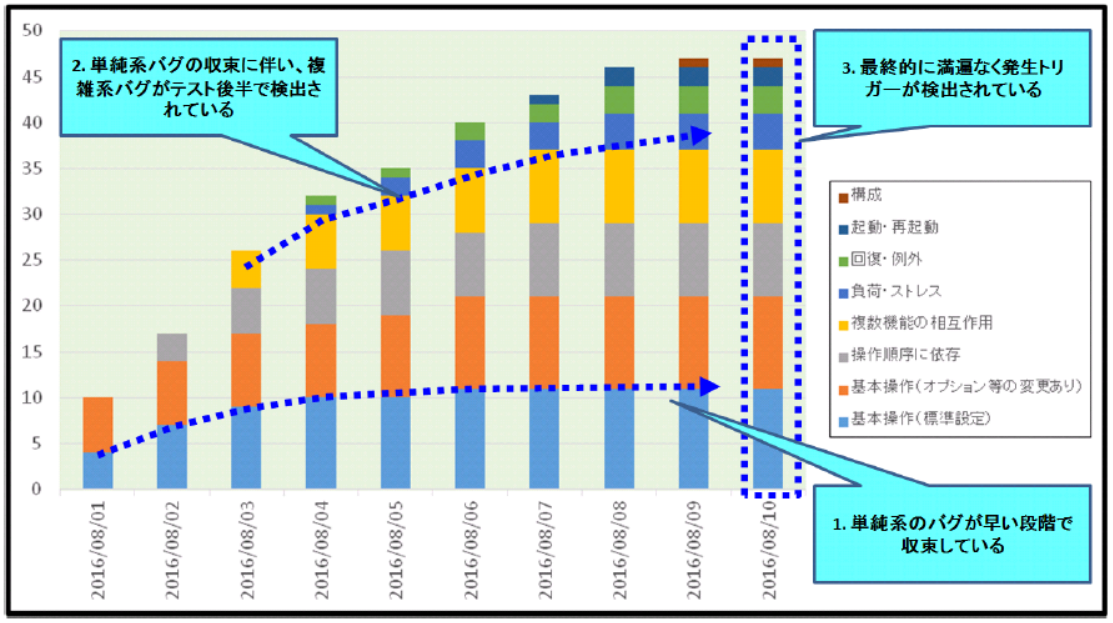
ご清聴ありがとうございました

質疑応答用

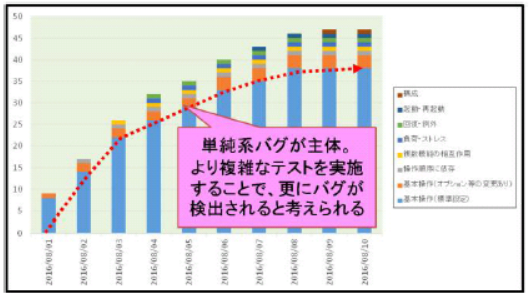


ODC分析により、テストの成熟度を測る

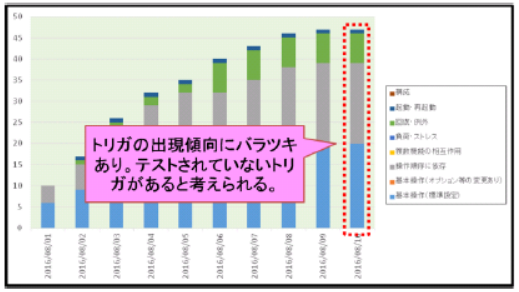
あるべき姿



問題事例①



問題事例②



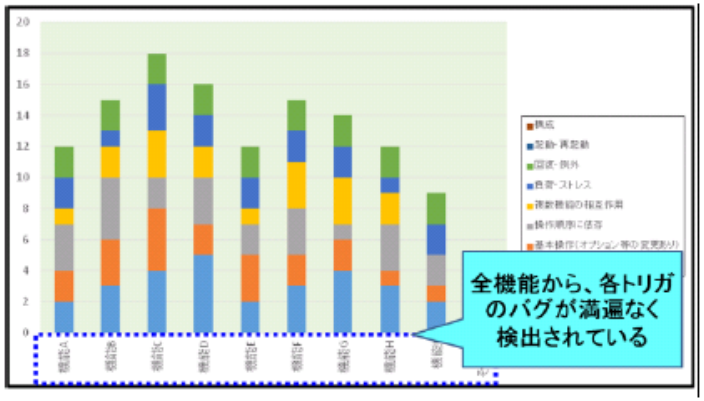
着眼点：
- 件数の推移
- 発生トリガーの偏り具合

モモンガ

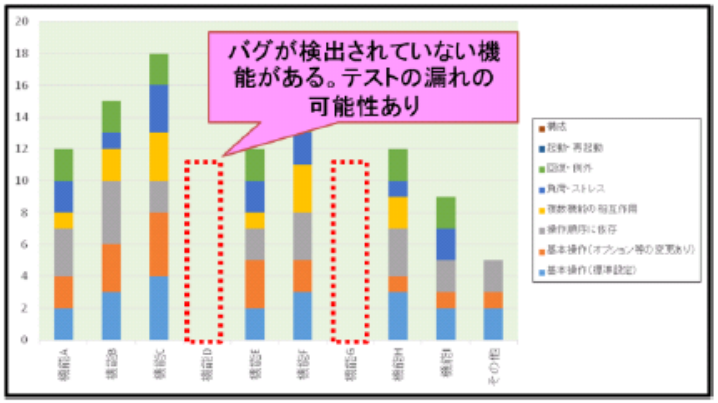
ODC分析により、テストの網羅度を測る

着眼点：　機能毎の発生トリガーの内訳

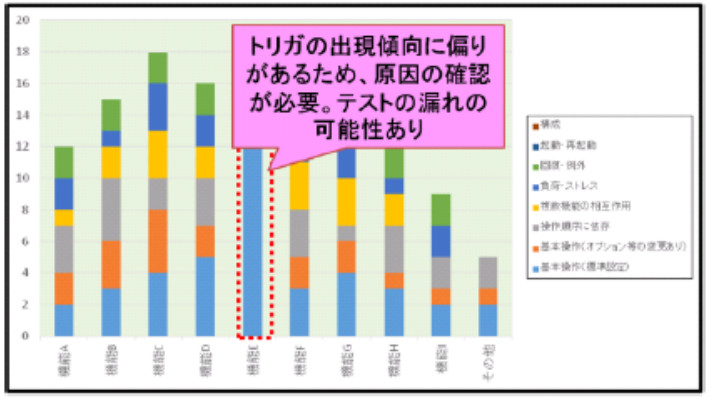
あるべき姿



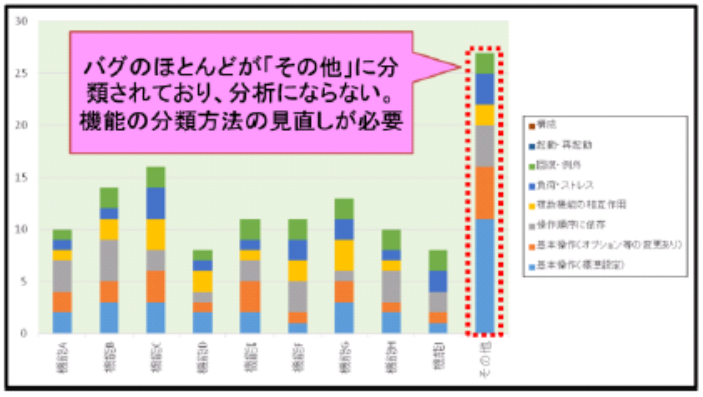
問題事例①



問題事例②



問題事例③



テスト条件系観点と ODC発生トリガーの対応付け

テスト条件						
←粗い観点～～細かい観点→			使用を想定するテストレベル			発生トリガ
			単体	結合	システム	
機能			–	●	●	基本操作(標準設定)
操作	タイミング		–	–	●	複数機能の相互作用、操作順序に依存
	常時実行可能な操作		–	–	●	複数機能の相互作用、操作順序に依存
	反復		–	–	●	操作順序に依存
	放置		–	–	●	操作順序に依存
UI	コントロールの種類		–	–	●	基本操作(標準設定)
	レイアウト		–	–	●	基本操作(標準設定)
	画面		–	–	●	基本操作(標準設定)
	状態の表示		–	–	●	基本操作(標準設定)
データ	データサイズ		●	●	●	基本操作(オプション等の変更あり)
	データ種別		●	●	●	基本操作(オプション等の変更あり)
	音声データフォーマット		●	●	●	基本操作(オプション等の変更あり)
	映像データフォーマット		●	●	●	基本操作(オプション等の変更あり)
	ジャケ写データフォーマット		●	●	●	基本操作(オプション等の変更あり)
	SEデータフォーマット		●	●	●	基本操作(オプション等の変更あり)
	コンテンツ		●	●	●	基本…
ハードウェアI/F	マイク入力端子(有線)に接続可能な機器		–	●	●	構成…
	マイク入力端子(無線)に接続可能な機器		–	●	●	構成…
	音声入力端子に接続可能な機器		–	●	●	構成…
	音声出力端子に接続可能な機器		–	●	●	構成…
	サービスコンセント(連動)に接続可能な機器		–	●	●	構成…
	サービスコンセント(非連動)に接続可能な機器		–	●	●	構成…
	映像入力端子に接続可能な機器		–	●	●	構成…
	映像出力端子に接続可能な機器		–	●	●	構成…
	HDMI出力端子に接続可能な機器		–	●	●	構成…
	ネットワーク端子に接続可能な機器		–	●	●	構成…
	コントロール端子に接続可能な機器		–	●	●	構成…
	ビルコインボックス端子に接続可能な機器		–	●	●	構成…
	外部リモコン端子に接続可能な機器		–	●	●	構成…
	シリアル端子に接続可能な機器		–	●	●	構成…
	USB端子に接続可能な機器		–	●	●	構成…
負荷	限界		–	–	●	負荷…
	耐久		–	–	●	負荷…
動作不能	給電停止		–	–	●	起動…
	故障		–	–	●	起動…
	ネットワーク切断		–	–	●	起動…
プラットフォームの脆弱性	ポートスキャン		–	●	●	基本…
	サービススキャン		–	●	●	基本…
	脆弱性スキャン		–	●	●	基本…
アプリケーションの脆弱性	パラメータ改竄により顕在化される脆弱性	バッファオーバーフロー	–	●	●	基本…
		クロスサイトスクリプティング	–	●	●	基本…
		クロスサイトリクエストフォージェリ	–	●	●	基本…
		OSコマンドインジェクション	–	●	●	基本…
		SQLインジェクション	–	●	●	基本…
		ディレクトリトラバーサル	–	●	●	基本…
		セッション管理の不備	–	●	●	基本…
		メール第三者中継	–	●	●	基本…
		HTTPヘッダインジェクション(レスポンス分割)	–	●	●	基本…
		Xpathインジェクション	–	●	●	基本…
		XMLインジェクション	–	●	●	基本…
		オープンリダイレクタ	–	●	●	基本…
	UIの構造に依存した脆弱性	オートコンプリート機能の不適切な使用	–	●	●	基本…
	サーバの設定により顕在化する脆弱性	強制ブラウジング	–	●	●	基本操作(オプション等の変更あり)
		ディレクトリリスティング	–	●	●	基本操作(オプション等の変更あり)
		JSONハイジャック	–	●	●	基本操作(オプション等の変更あり)
		クリックジャッキング	–	●	●	基本操作(オプション等の変更あり)
		クロスフレームスクリプティング	–	●	●	基本操作(オプション等の変更あり)
		ブラウザキャッシュ管理の不備	–	●	●	基本操作(オプション等の変更あり)
	マルチスレッド処理時に顕在化する脆弱性	競合状態	–	●	●	基本操作(オプション等の変更あり)
ネットワークの暗号化強度	SSL/TLS	SSL/TLS プロトコル(バージョン)	–	●	●	基本操作(オプション等の変更あり)
		SSL/TLS Chiper suite	–	●	●	基本操作(オプション等の変更あり)
	IPsec (トンネルモード/トランスポートモード共通)	IPsec暗号化アルゴリズム	–	●	●	基本操作(オプション等の変更あり)
		IPsec認証アルゴリズム	–	●	●	基本操作(オプション等の変更あり)
		IKE 暗号化アルゴリズム	–	●	●	基本操作(オプション等の変更あり)
		IKE 認証アルゴリズム	–	●	●	基本操作(オプション等の変更あり)
		IKE DHグループ	–	●	●	基本操作(オプション等の変更あり)
ストレージ/データの暗号化強度	暗号化アルゴリズム(鍵長)		–	●	●	基本操作(オプション等の変更あり)
不正利用	既定機器以外		–	–	●	回復・例外
営業形態に応じた構成			–	–	●	構成
定められている性能目標の条件			–	●	●	基本操作(オプション等の変更あり)
ユーザー環境	利用者の性別		–	–	●	基本操作(オプション等の変更あり)
	年齢		–	–	●	基本操作(オプション等の変更あり)
	職業		–	–	●	基本操作(オプション等の変更あり)
	利用シーン		–	–	●	基本操作(オプション等の変更あり)
ユーザー視点テストの方式	システムテスト担当者によるOK/NG判定方式		–	–	●	基本操作(オプション等の変更あり)
	モニターへのアンケート方式		–	–	●	基本操作(オプション等の変更あり)