TEF道『聡美塾』Presents
ユーザー視点とテストの素敵なコラボ
～魅力あるソフトウェアを創り出すのはテストから～

TEF道 are

小楠　聡美（聡美塾・塾長）
上田　和樹
小楠　貴紀
佐々木　誠司
高木　進也
中岫　信
根本　紀之
藤田　将志
村上　寛
安達　賢二（お世話係）

はじめに

- 「ワークショップ」と銘打っていますが、時間と場所の都合上「チュートリアル」に近い内容になることを、あらかじめご了承ください

アジェンダ

- イントロダクション(10分)
- Step1：既存テストケース作成方法確認【演習】(40分)
- Step2：今回のテストプロセス（TEF道の提案内容）概説【レクチャー】(20分)
- Step3：テスト分析～テスト設計～テストケース作成【演習】(40分)
- Step4：まとめ(5分)
- 質疑応答(5分)

TEF道のご紹介

- 正式名称：TEF北海道ソフトウェアテスト勉強会
  - Testing Engineer's Forum HOKKAIDO
- 愛称「TEF道（てふ-どう）」
- 隔週で一回2時間程度の開催。参加メンバーは10名程度で、常時5～6名のメンバーが参加。
- 2006年～2008年
  - **JSTQBテスト勉強会**として発足。断続的な活動。
- 2009年
  - **JaSST '09北海道にて成果発表**
- 2010年
  - **JaSST '10東京に事例発表の応募**

ワークショップの目的

- テストプロセスの実践によってテストへの理解を深めて頂く
- 他人の成果物と自分の成果物を比較することによって、新たな「気づき」を得る
- プロセスの違いによって、「テスト仕様」がどのように変化するかを体験して頂く

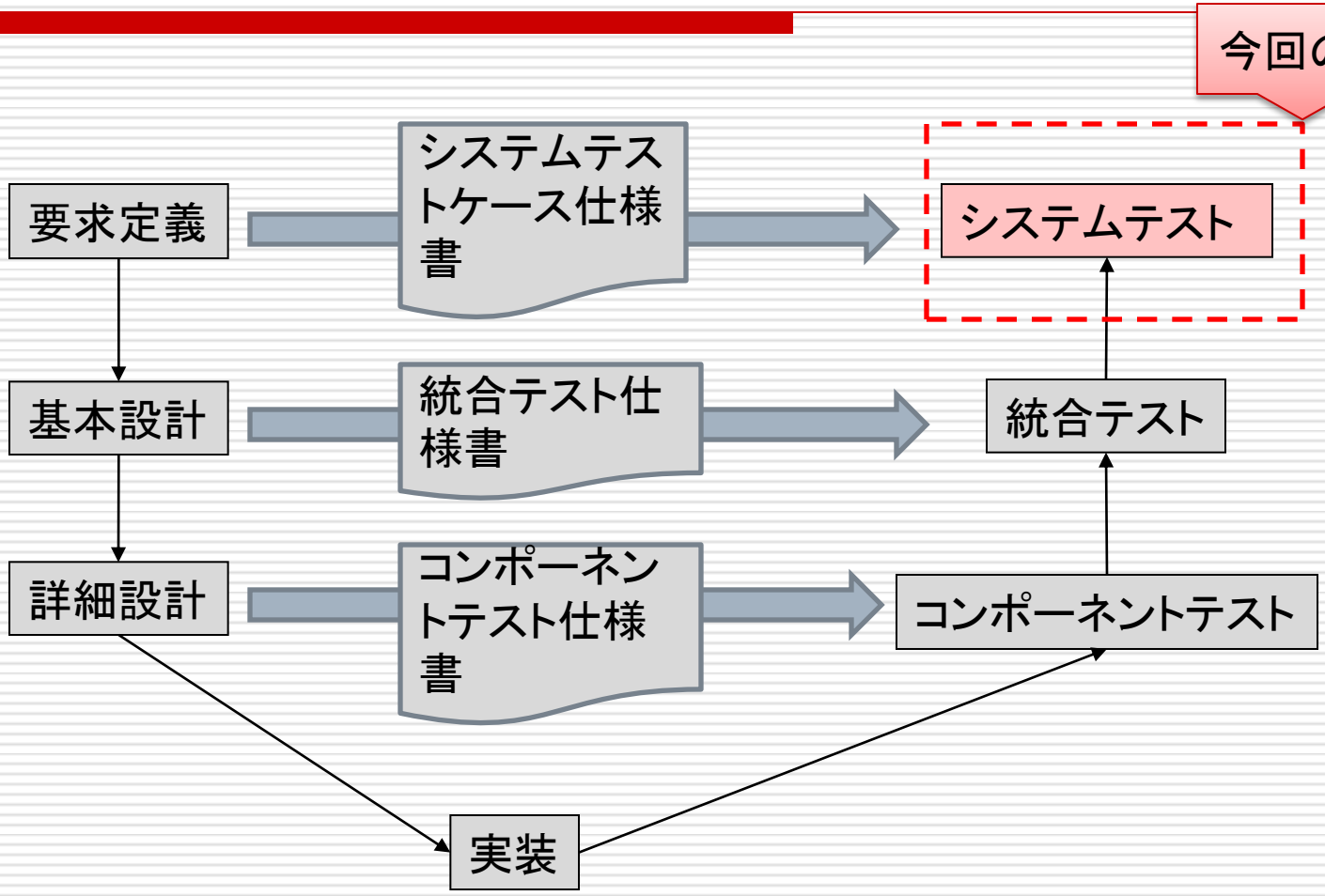
本ワークショップの実施概要

- テスト対象は「英語読書日記システム」
- あなたの役割は、システムテストの仕様書作成とテストを実施すること
- 要求仕様書は作成済み

要求定義(携帯アプリ)

- 配布してある<要求仕様書>をご確認ください

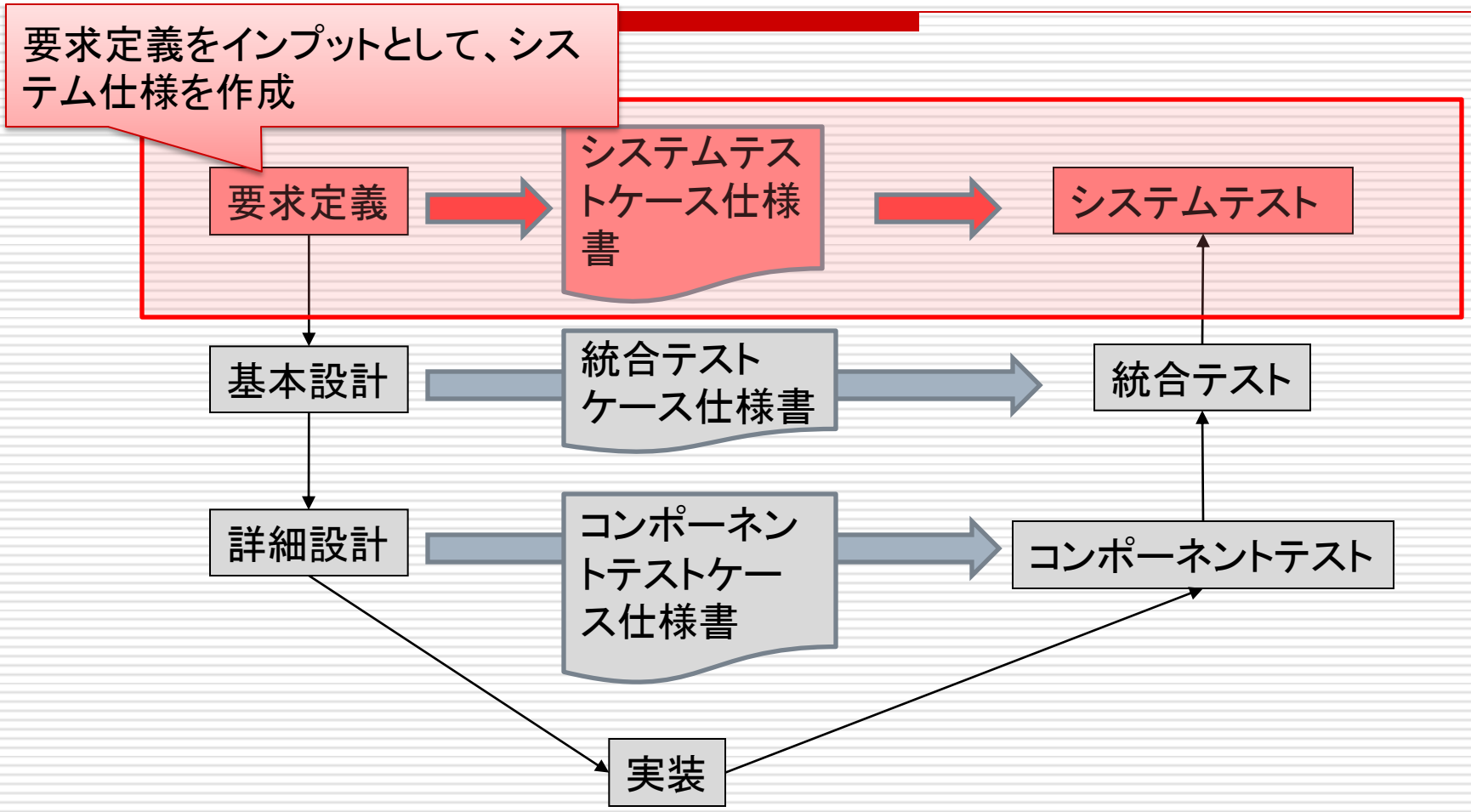
開発プロセスの全体像（概略）



STEP1：既存テストケース作成方法確認【演習】

- 演習1-1：要求仕様書からテストケースを作成
- 演習1-2：テストケース仕様書から「テストケース分析表」を作成
- 演習1-3：意見交換

STEP1：既存テストケース作成方法確認【演習】



テストケース仕様書とは？

「入力値、実行事前条件、期待結果、そして、実行事後条件の組み合わせで、特定のプログラムパスを用いることや指定された要件の遵守を検証することのような、特定の目的またはテスト条件のために開発されたもの」(JSTQB用語集より)

今回使用するテストケース仕様書

テストケースNo.	テスト事前条件	テスト手順	期待値

テストを実行する前に満たす必要がある条件

テストの手順テストのインプットとなる

指定の条件下で期待されるシステムの動作

演習1-1：要求仕様書からテストケースを作成してみてください

- 普段業務で行なっているやり方で作成してみてください
- テストケースを書いたことのない方も、「たぶんこうじゃないか？」という方法でやってみてください
- テストレベルは**「システムテスト」**です
- 「重要」と思う機能から作成を行なってください

作業時間：今から15分間

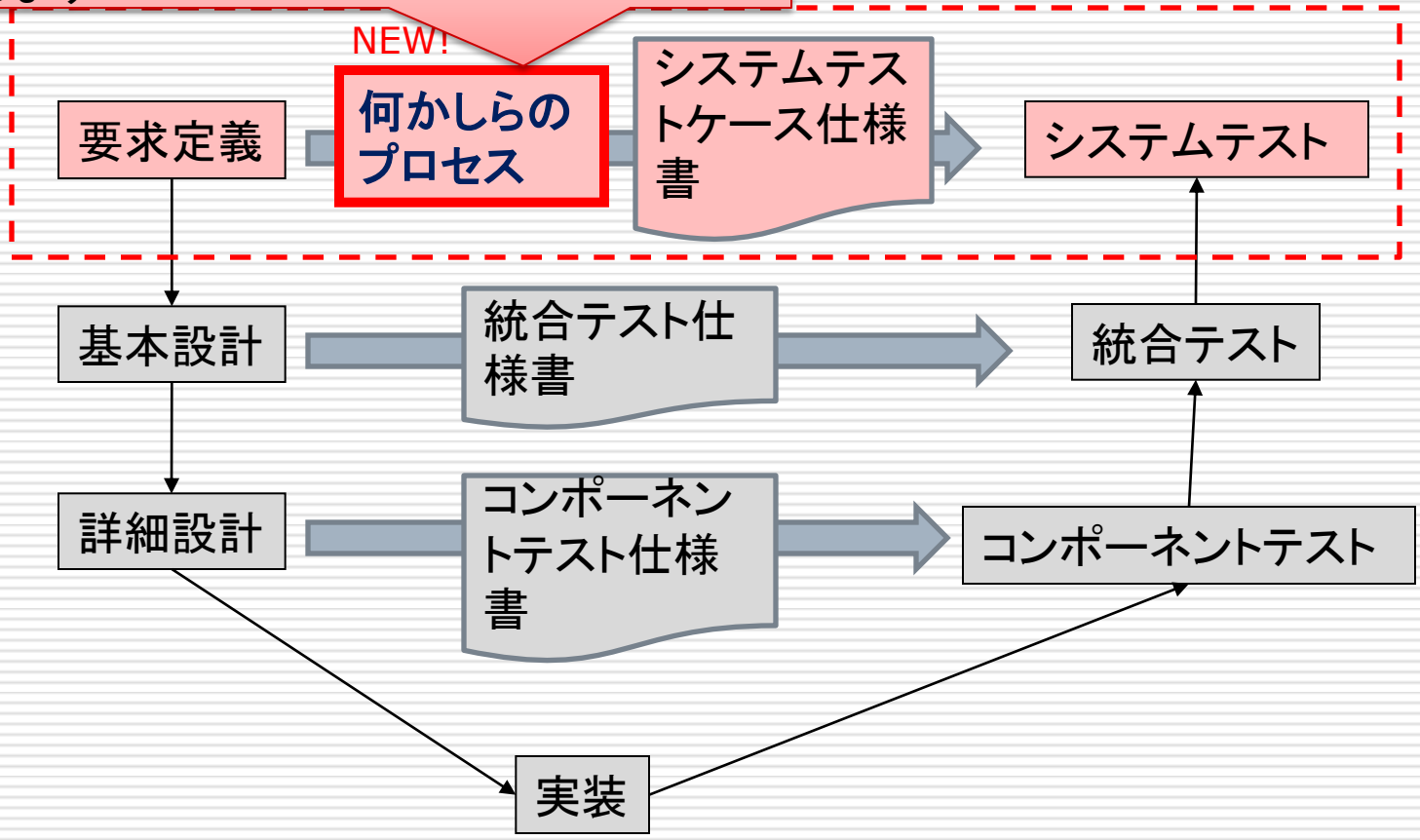
演習1-1：終了

良いテストケースが作成できましたか？

- 「システムテスト」の仕様書になっていますか？
- 作成したテストケースが「どのように導き出された」をこれから明らかにします
- これから配布する「テストケース分析表」にて、テストケースの『リバースエンジニアリング』を行ないます

演習1-2：作成したテストケースが、どのように作られたかを解明します

テスト仕様書が「どのように導き出されたかの解明をします



演習1-2:テストケース分析表

テストケースNo.	このテストケースの目的	使用したテスト技法	想定したユーザー（立ち位置）	想定した非機能

このテストで「何を」確認したいのか？

何かのテスト技法を使ってケースを作成した場合は記入

どこの立ち位置からテストケースを作成したのか？どういうユーザーを想定したテストか？

想定した『非機能』を記入
（使いやすさやレスポンス時間など）

演習1-2：テストケース仕様書から「テストケース分析表」を記入してください

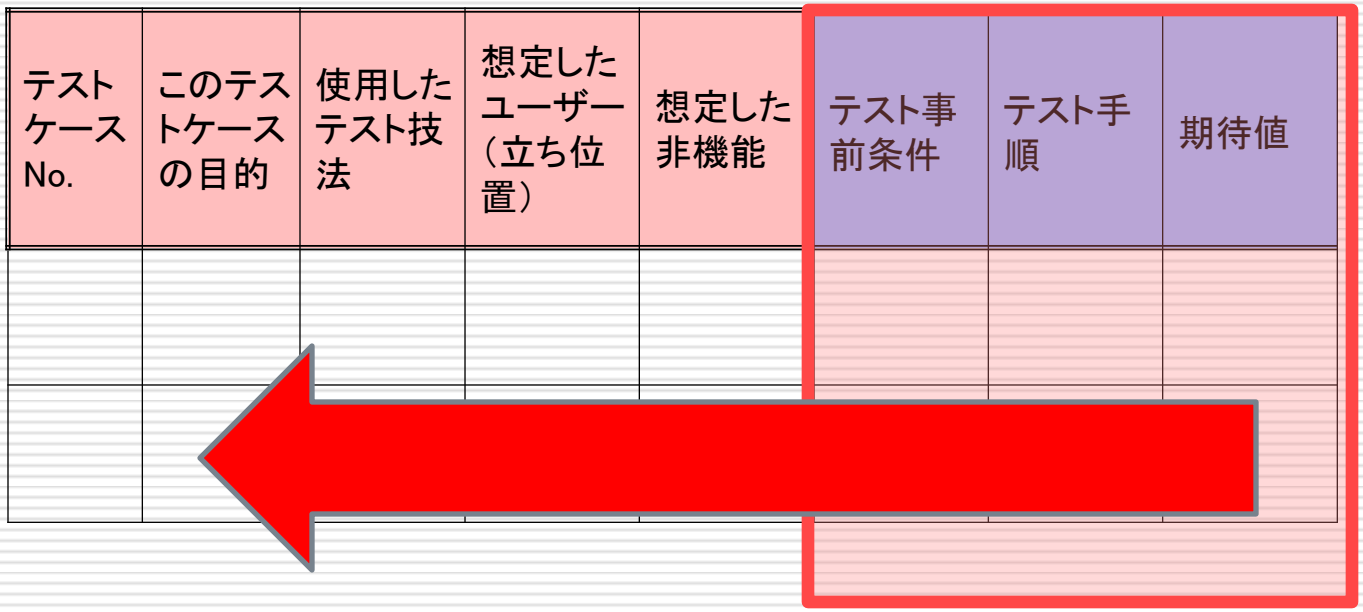
- 作成したテストケースは**「何を確認する目的」**で作りましたか？
- 作成したテストケースは**「何かのテスト技法」**を使用して作りましたか？
- 作成したテストケースは**「誰の立ち位置」**で作りましたか？
- 作成したテストケースで考慮した**「非機能」**はありましたか？

作業時間：今から10分間

演習1-2：終了

自分がどういう「方針」でテストケースを作成したか、明確になったと思います。

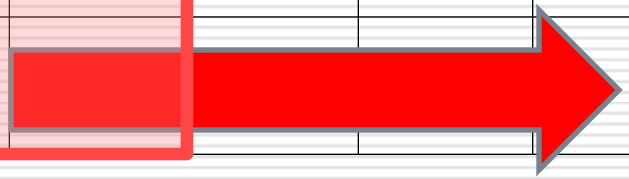
テストケース分析表⇒テストケース仕様書の順に横に並べてください



テストケース仕様書（右）から左の表を考えたように思えるが・・・・・

テストケースからテスト設計へのリバースエンジニアリング

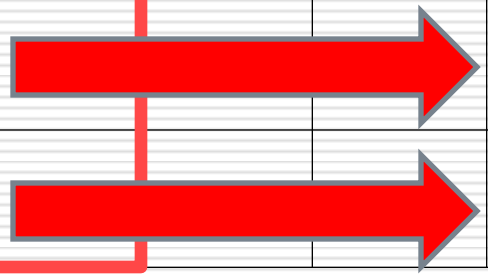
テストケースNo.	このテストケースの目的	使用したテスト技法	想定したユーザー（立ち位置）	想定した非機能	テスト事前条件	テスト手順	期待値



実は、左の表は、右のテストケースを作成するために、「アタマの中」で考えていた筈の内容。

本来は、左の表（テスト設計仕様書）から、テストケースへブレイクダウンする

テストNo.	確認すべきテスト対象	使用するテスト技法	想定すべきユーザー（立ち位置）	想定すべき非機能	テスト事前条件	テスト手順	期待値	テスト事後条件



語尾を変えると、ちょっとテスト設計仕様書っぽくなりましたね

ポイント

テストNo.	確認すべきテスト対象	使用するテスト技法	想定すべきユーザー（立ち位置）	想定すべき非機能	テスト事前条件	テスト手順	期待値	テスト事後条件

【テスト設計仕様書のポイント】
・設計仕様書：テストケース⇒1：nになることもある
・業務では明示的に作成しないことも多いかも（？）
・普段書かない方も、アタマの中では作成している
・「テスト戦術」から実際のテストケース（手順など）へブレイクダウンしていく

作成例

テストケースNo.	このテストケースの目的	使用したテスト技法	想定したユーザー（立ち位置）	想定した非機能	テスト事前条件	テスト手順	期待値
1	検索ボタン押下で画面遷移することの確認	特になし	携帯に使い慣れた若者	レスポンス性	ログイン⇒読書登録画面に遷移	検索ボタンを押す	検索画面に素早く遷移することを確認
2	検索画面の結果（本のタイトル）が反映されることを確認	特になし	特になし	特になし	検索画面で本を特定して当画面に戻った後	「名前」の欄を確認	「名前」の欄に検索した本の名称が表示されていることを確認
3	検索画面の結果（レベル）が反映されることを確認	境界値テスト	特になし	特になし	検索画面に遷移	LV1の本を選択	検索したレベルが表示されることを確認
4					検索画面に遷移	LV99（最大値）の本を選択	検索したレベルが表示されることを確認

演習1-3：隣の方と意見交換してください

- 意見交換のポイント
  - なぜ、そのテストを選んだのか？（重要だと思ったのか？）
  - 自分との違いは？
  - 左の表から右の表へブレイクダウンしているように見えますか？
  - 本当にシステムテストになっていますか？

作業時間：今から10分間

演習1-3：終了

- テストの「指針」が明確になっていますか？
  - 普段テスト設計仕様書を書いてる方は、自分のやっている作業の再認識をしてみて下さい
  - 普段テスト設計仕様書を意識してない方は、「実はアタマの中でやってること」なので、意識的に書き出すと、自分の「テスト指針」が明確になり、他メンバーと「共有」が出来ます

さて、「良いシステムテスト設計書」を作るにはどうすれば良いのでしょうか？

STEP2：今回のテストプロセス（TEF道の提案内容）概説【レクチャー】

- 5W1H分析
- スープカレー表：縦軸
- スープカレー表：横軸
  - ユーザー分析
  - 非機能要求
- スープカレー表：交点を埋める
- テスト設計の例

システムテスト仕様設計書作成の為の「5W1H」分析

テストケース分析表

テストNo.	確認すべきテスト対象	使用するテスト技法	想定すべきユーザー（立ち位置）	想定すべき非機能	テスト事前条件	テスト手順	期待値
	テスト対象の分析（何を何のためにテストするのか？）	どのようにテストするか？	想定ユーザー（誰がどの状況で使うのか？）	非機能（どのように動くのか？）			
	What Why	How	Who Where When	How			

システムテストの設計は「5W1H」で表すことができる

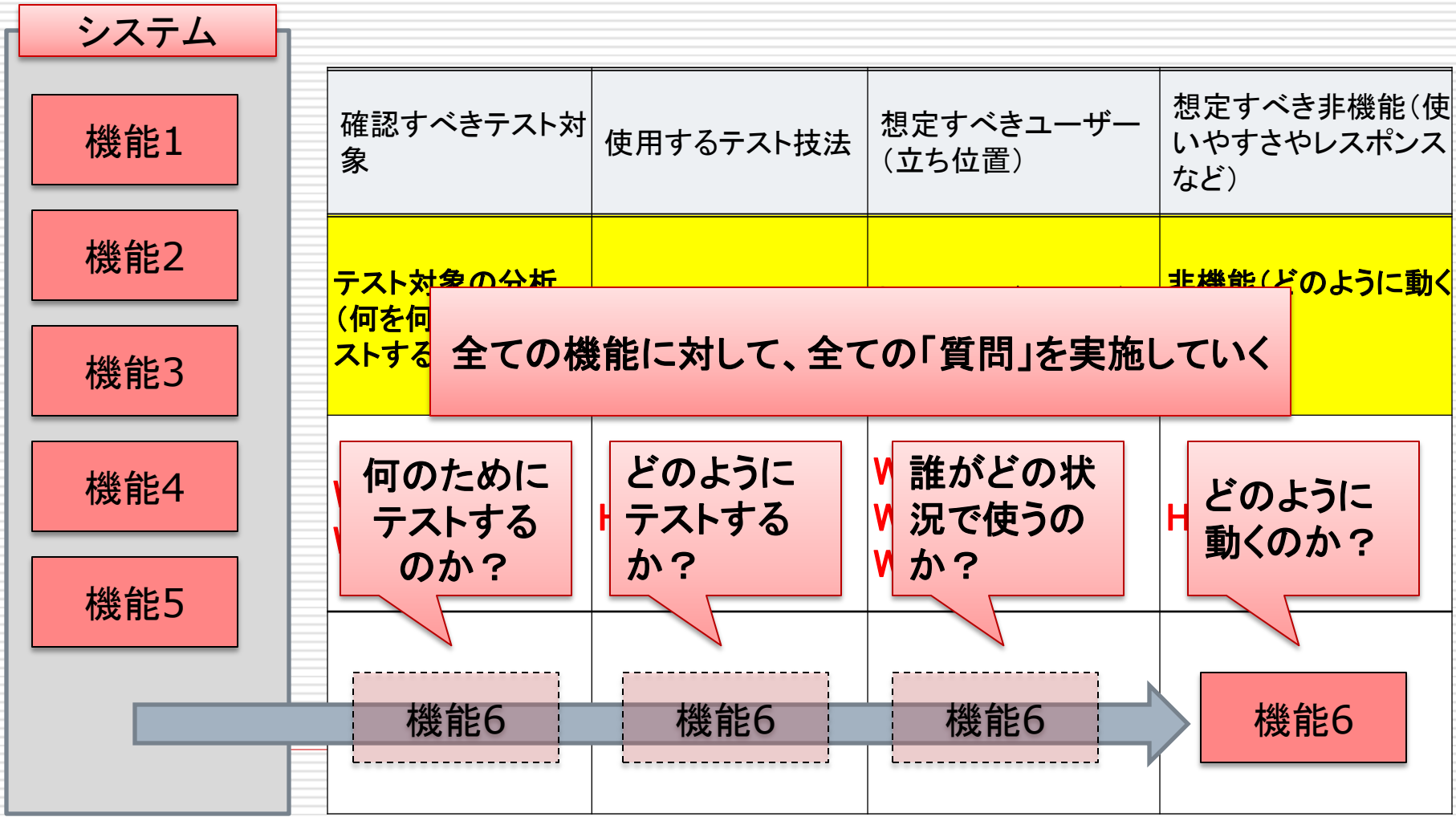
システムを機能に分割して、一つの機能ごとに各項目を「質問」していく

システム

- 機能1
- 機能2
- 機能3
- 機能4
- 機能5
- 機能6

確認すべきテスト対象	使用するテスト技法	想定すべきユーザー（立ち位置）	想定すべき非機能（使いやすさやレスポンスなど）
テスト対象の分析（何を何のためにテストするのか？）	**どのようにテストするか？**	**想定ユーザー（誰がどの状況で使うのか？）**	**非機能（どのように動くのか？）**
What Why	How	Who Where When	How

システムを機能に分割して、一つの機能ごとに各項目を「質問」していく

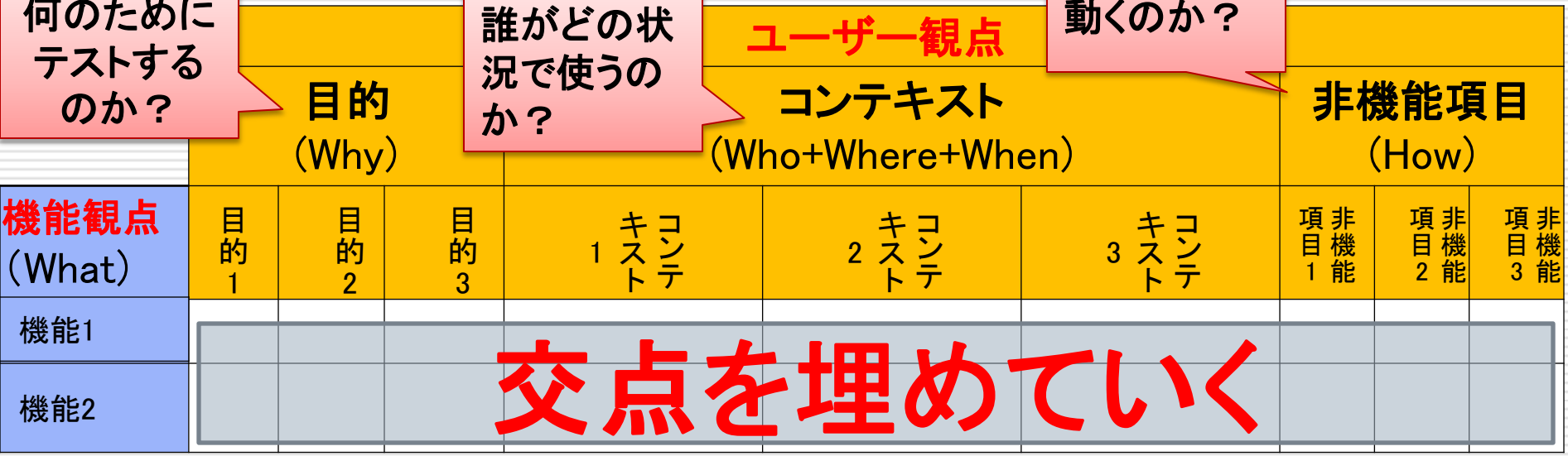


そこで、このようなマトリクスを考えてみました

何のためにテストするのか？

誰がどの状況で使うのか？

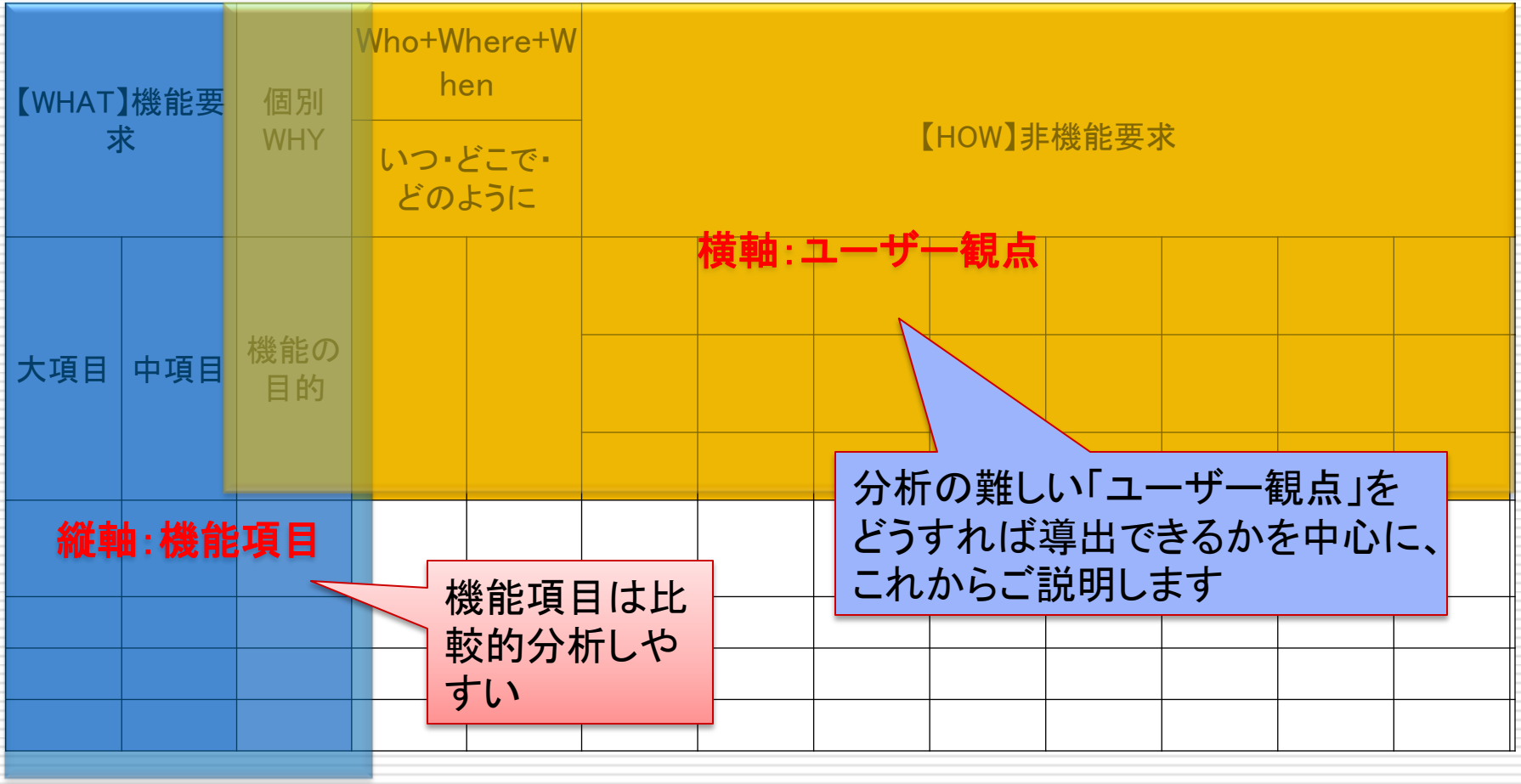
どのように動くのか？



- 「スープカレー表」と命名しました
  - JaSST'09 Hokkaidoにて最初の発表
  - システムの「5W1H」を表にまとめる
  - 縦に「機能観点」、横に「ユーザー観点」を記入
  - その「機能観点」×「ユーザー観点」の交点を埋めていく
  - 「機能」を【ライス】、ユーザー観点を【スープと具】と見立てて、「機能を次々とユーザー観点に潜らせる」イメージから

今回使用するスープカレー表

- 配布資料の「スープカレー表」をご覧下さい



スープカレー表①：縦軸に機能(What)の記入

- 要求仕様書から、適した粒度で機能を抽出

【WHAT】機能要求	
大項目	中項目
読書データ登録	本のタイトル表示
	レベル表示
	検索画面遷移
	日付入力
	ストップウォッチ
	読書時間入力／表示
	速度計測（表示）
	データ登録

スープカレー表②：機能の目的(Why)の記入

- その機能が「どういう目的で作られたか？」を考える
  - インプットは想定したユーザー像やシナリオや「想像力」
  - 機能は、その動作を確認するだけでは「正しい機能」とはいえない

    ⇒「目的」の無い機能は無い

    ⇒「目的」を果たさない動作は、いくら仕様通りでも正しいとは言えない

    （例）

    機能⇒本のタイトル表示

    機能の目的⇒**「自分の読んでいる本を登録する際に、タイトル確認するため」**

スープカレー表③:
システムの利用状況
(Who+Where+When)の記入

- システムの利用状況(だれが、いつ、どこで)を考える
  - システムや機能はその利用状況で、求められる振る舞いや目的が大きく変わるため
- 想定ユーザーから基本的なユーザーシナリオを考え、使用順に当てはめていく
  - シナリオの流れを確認できる
  - 複数のシナリオを記入することで、「重要な機能」が抽出できる

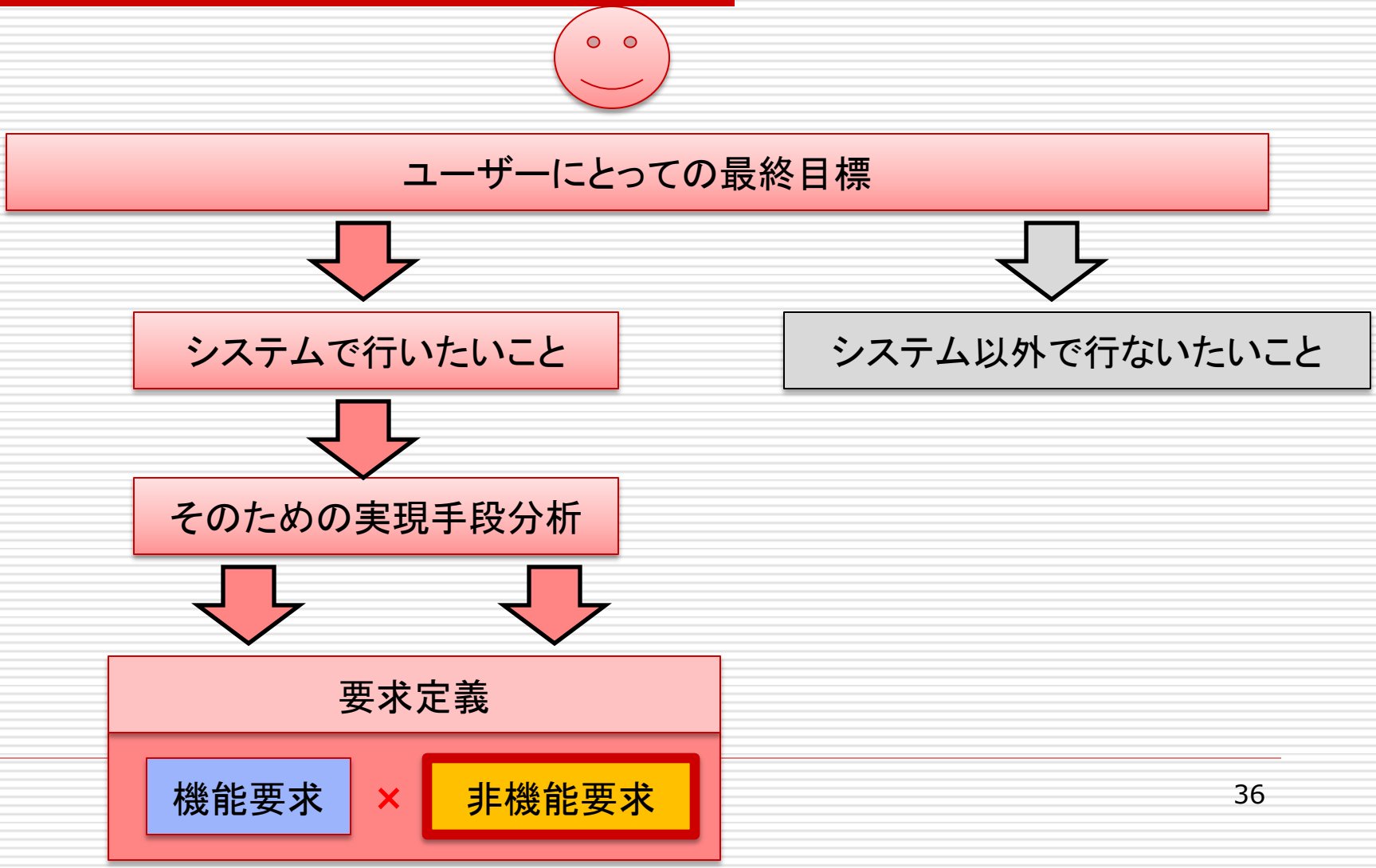
※配布資料「中山 由信ペルソナ」「中山 由信シナリオ」を参照ください

スープカレー表④非機能(How)の必要事項を記入

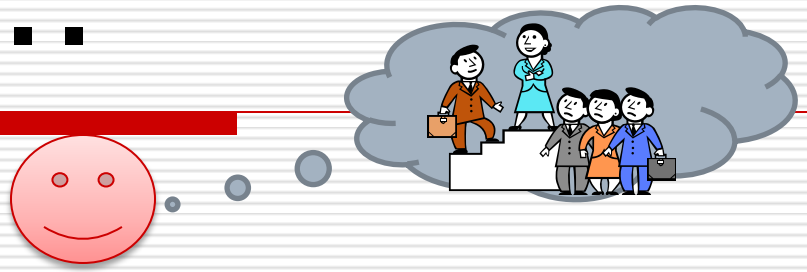
- 横軸に、非機能要求（プログラムが「どのように」動作する事を求めているか）を分析して記入
- 非機能要求・・・「レスポンス性」「使いやすさ」「信頼性」などの機能以外の要求

この要求を分析するには「想定ユーザー像」を知る必要がある

中山さんの「やりたいこと」をシステムで実現するには・・・



中山さんの「やりたいこと」をシステムで実現するには・・・



英語ができないと昇進できない！
⇒「英語の本を沢山読めば英語が上達するのでは？」

読書日記をシステムで記録することによって、学習意欲が沸く

システム以外で行ないたいこと

そのための実現手段分析

スープカレー表の縦軸

要求定義

機能要求 × 非機能要求

スープカレー表の横軸

中山さんの「非機能要求」を洗い出す

英語ができないと昇進できない！
⇒「英語の本を沢山読めば英語が上達するのでは？」

読書日記をシステムで記録することによって、学習意欲が沸く

手書きの記録等手間を削減できる	学習経過・上達度合等の状況が手間なく把握できる	飽きずに、楽しく取り組みを続けられる

（ユーザー視点からの）非機能要求

愛着がわく画面表示	パッと見で分かるグラフィカルな表示	わかりやすい操作	サクサク動く
いつでも、どこでも	最小限の手間と時間で対応できる	他の人にデータをみられたくない	

しかし、非機能要求を洗い出してみたものの・・・

非機能要求
愛着がわく画面表示
パッと見で分かるグラフィカルな表示
わかりやすい操作
サクサク動く
いつでも、どこでも
最小限の手間と時間で対応できる
他の人にデータをみられたくない

システムとして扱うには、まだ粗すぎる？

「品質特性」を使用して分析できるのでは？

品質特性とは？

- **ISO 9126** で定められた、ソフトウェア品質の評価に関する国際規格
- 「ソフトウェア品質」を構造的に定義

機能性	信頼性	使用性	効率性	保守性	移植性
• 合目的性 • 正確性 • 相互運用性 • 機密性 • 標準適合性	• 成熟性 • 障害許容性 • 回復性 • 標準適合性	• 理解性 • 習得性 • 運用性 • 注目性 • 標準適合性	• 時間効率性 • 資源効率性 • 標準適合性	• 解析性 • 変更性 • 安定性 • 試験性 • 標準適合性	• 環境適応性 • 設置性 • 共存性 • 置換性 • 標準適合性

非機能要求分析表

①ISO9126の品質特性より

②中山さんから導き出される非機能要求

			対象	愛着がわく画面表示	わかりやすい操作	サクサク動く	いつでも、どこでもできる	最小限の手間・時間で対応できる	パッと見で分かるグラフィカルな表示
使用性	理解性	どうしたらよいかが分かりやすい	◎		●			△	●
	習得性	使い方を覚えやすい	◎		●			△	
	運用性	導入・操作・対話の手間が少ない	◎		●			●	
	魅力性	かっこいい、かわいい等	◎	●					●
効率性	時間効率性	作業時間が短くて済む	◎			●	●	●	
	資源効率性	容量等が少なくて済む	△			△			
保守性	解析性	ソフトの内容が分かりやすい							
	変更性	あとからソフト やすい							
	安定性	ソフトを直して しくなりにくい							
	試験性	ソフトを直した 認しやすい							
移植性	環境適応性	いろんなところで使える	（●）				（●）		

④今回のシステムに必要な品質特性の決定

③その非機能項目が、品質特性のどの特性に当てはまるかをプロット

品質特性での分析結果

ここの内容は、品質特性とシステムによって、個別に分析する

ユーザーとしての非機能要求	品質特性	システムとしての非機能要求
愛着がわく画面表示	魅力性	見た目がかわいい
パッと見で分かるグラフィカルな表示	理解性	情報が見やすい・わかりやすい
わかりやすい操作	理解性	どうしたらよいかが分かりやすい
	習得性	使い方を覚えやすい
サクサク動く	時間効率性	レスポンスがよい
いつでもどこでも	環境適用性	
最小限の手間と時間で対応できる	運用性	
	時間効率性	作業時間が短くて済む
他の人にデータをみられたくない	セキュリティ	許された人しか使えない

非機能要求が少しシステム寄りになり、扱いやすくなった

スープカレー表⑥：交点を埋めていく

- 機能ごとに横軸の非機能要求を「くぐらせ」て、機能に非機能観点を「パッケージング」する。

「本のタイトル表示」機能が「見た目がかわいい」という要求を満たすにはどうすれば良いか？

「本のタイトル表示」機能が「レスポンスが良い」という要求を満たすにはどうすれば良いか？

機能	わかりやすい操					最小限の手間と時間で対応できる		他の人にデータをみられたくない
	性	理解性	理解性	習得性	性	適用性	運用性	時間効率性	セキュリティ
	見た目がかわいい	**情報が見やすい・わかりやすい**	**どうしたらよいかが分かりやすい**	**使い方を覚えやすい**	**レスポンスがよい**	**いろんなところで使える**	**導入・操作・対話の手間が少ない**	**作業時間が短くて済む**	**許された人しか使えない**
本のタイトル表示	**字体・配色等で画面やボタンがかわいい**	大きい文字、見やすい字体、グラフ等で見やすい、意味を捉えやすい	書名が表示される部分だと分かる	一度使うと、左記のようなもので、こういう使い方だと分かる	**検索結果の表示が速い**		（理解性：表示が見やすい→意味を把握しやすい）で代替	（理解性：表示が見やすい→意味を把握しやすい）で代替	

スープカレー表：例

【WHAT】機能要求		個別WHY	いつ・どこで・どのように		【HOW】非機能要求								
大項目	中項目	機能の目的	シナリオ1 中山さん朝通勤時、（学園都市線）電車内で	シナリオ2〇〇〇	愛着がわく画面表示	パッと見で分かるグラフィカルな表示	わかりやすい操作		サクサク動く	いつでもどこでも	最小限の手間と時間で対応できる		他の人にデータをみられたくない
					魅力性	理解性	理解性	習得性	時間効率性	環境適用性	運用性	時間効率性	セキュリティ
					見た目がかわいい	情報が見やすい・わかりやすい	どうしたらよいかが分かりやすい	使い方を覚えやすい	レスポンスがよい	いろんなところで使える	導入・操作・対話の手間が少ない	作業時間が短くて済む	許された人しか使えない
読書データ登録画面	学習した経過と習熟度をあとで確認するために、 1.読書時間を計測する 2.今回の学習結果を登録する									シナリオの状況で他の非機能要件を満たしつつ動作する			同上
	本のタイトル表示	どの本を読んだか登録するため／確認するため	■S12：「読書データ登録画面」が表示され、書籍名”肉取物語”、レベル”YL3”、単語数”735”、そして読書速度”27.7”が表示されました。（表示のみ）		字体・配色等で画面やボタンがかわいい	大きい文字見やすい字体、グラフ等で見やすい、意味を捉えやすい	書名が表示される部分だと分かる	一度使うと、左記のようなもので、こういう使い方だと分かる	検索結果の表示（切り替わり）が速い	同上	（理解性：表示が見やすい→意味を把握しやすい）で代替	（理解性：表示が見やすい→意味を把握しやすい）で代替	同上
	レベル表示	読んだ本の難易度を登録するため／確認するため	■S12：「読書データ登録画面」が表示され、書籍名”肉取物語”、レベル”YL3”、単語数”735”、そして読書速度”27.7”が表示されました。（表示のみ）		字体・配色等で画面やボタンがかわいい	大きい文字見やすい字体、グラフ等で見やすい、意味を捉えやすい	難易度・総語数が表示されている部分だと分かる／表示だけされるものだと分かる	一度使うと、左記のようなもので、こういう使い方だと分かる	検索結果の表示（切り替わり）が速い	同上	（理解性：表示が見やすい→意味を把握しやすい）で代替	（理解性：表示が見やすい→意味を把握しやすい）で代替	同上
	検索画面遷移	読んだ本の情報を登録するため検索画面に遷移する	■S9：次に〇〇により「本検索画面」を表示させ、→【本検索画面】		字体・配色等で画面やボタンがかわいい	大きい文字見やすい字体、グラフ等で見やすい、意味を捉えやすい	検索画面にいけることが分かる／どうしたら検索できるかが分かる	一度使うと、左記のようなもので、こういう使い方だと分かる	画面遷移が速い	同上	検索に関わる手間が少ない	検索に関わる手間が少ない	同上

この表が意味するものは？

- システムテスト全体像が見渡せる
- ユーザ観点の「網羅」を確認できる
- 機能と非機能を包括した大きな**「要求仕様書」**
- ステークホルダ（開発者とテスト担当者とユーザー）と、共通の認識を持つことができる

「開発」「テスト」「ユーザー」
三つの観点から「システム」を表現したもの

スープカレー表からテスト設計書を作成した例（FV表およびFL表：HAYST法®より）

Function-Verification Table				Factor-Level Table	
No.	Function（目的機能）	Verification（検証方法）	Test（テスト技法）	Factor(因子)	Level(水準)
2.7	速度計測（表示）				
2.7.1	自分の読書速度（語／分）を簡単に分かりやすく確認するため	・リアルタイムで、分かりやすく読書速度が表示されることを確認	語数と分数の桁数の組合せテスト実施とレスポンステスト実施。ユーザー観点での画面デザインの検証	語数	・0 ・1 ・最大値
				分数	・0 ・1 ・最大値
2.8	データ登録				
2.8.1	データを登録して保管するため（→あとでメイン画面で経過を見る）	・データ登録は分かりやすく、押下後に1秒以内に登録完了画面に遷移することを確認 ・片手で操作できる感じを確認する ・登録データはメイン画面にて自分の成果として確認が出来ることを確認	更新項目を直交表で組合せて登録し、メイン画面で実績確認とレスポンステスト実施。ユーザー観点での画面デザインの検証	名前（タイトル）	・1文字 ・最大文字 ・英字 ・数字 ・ひらがな ・カタカナ ・漢字 ・半角ｶﾀｶﾅ ・記号
				レベル	・1 ・最大
				語数	・1 ・最大
				日付	・1/1 ・2/29 ・12/31
				読書時間	46 ・1 ・最大

「機能」と「非機能」をパッケージングすることによって、同様の粒度で扱えるようになる

STEP3：テスト分析～テスト設計～テストケース作成【演習】

- 演習2-1：スープカレー表を埋める（15分）
- 演習2-2：スープカレー表より、「テストケース仕様書」を作成する（10分）
- 演習2-3：最初に作った「テストケース仕様書」と比較して、隣同士で意見交換（5分）
- 演習2-4：最初に作った「テストケース分析表」とスープカレー表を比較して、隣同士で意見交換（5分）

演習2-1：スープカレー表を埋める

- 「機能」に必要な非機能項目を表に埋めてみてください
  - 例えば、「ログイン」機能に対し、「どうすれば分かりやすく使えるか？」という非機能要求を考える
  - 最初に作った「テストケース仕様書」と同じ機能の箇所から埋めていってください（後で比較します）
  - 必要だと思う欄を埋めてください

作業時間：今から15分間

演習2-1：例

- 機能ごとに横軸の非機能要求を「くぐらせ」て、機能に非機能観点を「パッケージング」する。

例

「本のタイトル表示」機能が「見た目がかわいい」という要求を満たすにはどうすれば良いか？

「本のタイトル表示」機能が「レスポンスが良い」という要求を満たすにはどうすれば良いか？

機能	わかりやすい操					最小限の手間と時間で対応できる		他の人にデータをみられたくない
	性	理解性	理解性	習得性	性 適用性	運用性	時間効率性	セキュリティ
	見た目がかわいい	**情報が見やすい・わかりやすい**	**どうしたらよいかが分かりやすい**	**使い方を覚えやすい**	**レスポンスがよい**	**いろんなところで使える**	**導入・操作・対話の手間が少ない**	**作業時間が短くて済む**	**許された人しか使えない**
本のタイトル表示	**字体・配色等で画面やボタンがかわいい**	大きい文字、見やすい字体、グラフ等で見やすい、意味を捉えやすい	書名が表示される部分だと分かる	一度使うと、左記のようなもので、こういう使い方だと分かる	**検索結果の表示が速い**		（理解性：表示が見やすい→意味を把握しやすい）で代替	（理解性：表示が見やすい→意味を把握しやすい）で代替	

演習2-1：終了

うまく埋まりましたか？

- 埋める際にユーザーが使うときのイメージが浮かびましたか？
- 本来は、この表からテスト設計仕様書を作成しますが、今回はいきなりテストケースを作成します

演習2-2：テストケースを作成する

- 「機能の目的」と「スープカレーの交点：機能に対する非機能要求」を確認できるようなテストを記入してください
- なるべく、少ない手順でテストできるように工夫してみてください
- テストに使用する入力値も一緒に考えてみてください

作業時間：今から10分間

演習2-2：例

中項目	機能の目的	見た目がかわいい	情報が見やすい・わかりやすい	どうしたらよいかが分かりやすい	使い方を覚えやすい	レスポンスがよい	いろんなところで使える			
							少ない			
日付入力	本を読んだ日を登録する／入力するため	–	日付はわかりやすく表示されている	・日付が手入力できることが分かる ・日付の入力操作は迷わずに行える	-	デフォルト表示は現在日付を1秒以内に表示する	-	-	-	

コツ①：テストの目的を考える

※テストの目的とは、「機能の目的」と「非機能要求」が満たされていることを確認すること！

コツ②：テストの目的を確認できる「事前条件」と「手順」を考える！

コツ③：期待値に、「機能」と「非機能」の両方を確認できる結果を盛り込む

例

テスト事前条件	テスト手順	期待値
本の検索が行なわれ、登録画面に戻ってくる	1、日付欄のデフォルト表示の確認 2、日付にフォーカスを当てる 3、「2010/01/01」を入力する 4、登録ボタンを押下する	1、検索画面から戻ってきた際に、1秒以内にシステム日付をデフォルト表示する 2、日付が手入力できることが理解できるような表示になっている（色や動作など） 3、日付に「2010/1/1」が表示される 4、登録ボタン押下後にメイン画面に遷移する

一つの手順で、機能と非機能の両方が効率的に確認できる

演習2-2：終了

良いテストケースが作成できましたか？

- 「システムテスト」の仕様書になっていますか？
- 機能だけではなく、非機能も考慮されていますか？

演習2-3：最初に作ったテストケースと比較

- 最初に作ったテストケースと比較して、隣同士で意見交換してください
  - 内容にどのような違いがありますか？
  - 違いの理由はどこにあると思いますか？
  - それぞれのテストを実施していったときに、どのような違いが生まれると思いますか？

作業時間：今から5分間

演習2-3：終了

テストケースに違いがありましたか？

- テストケースに「非機能」が考慮されていると思います
- テストケースに「想定したユーザーの実利用時要求」が考慮されていると思います
- つまり「ユーザー」を想定したテスト（⇒システムテストの要素）になっていると思います

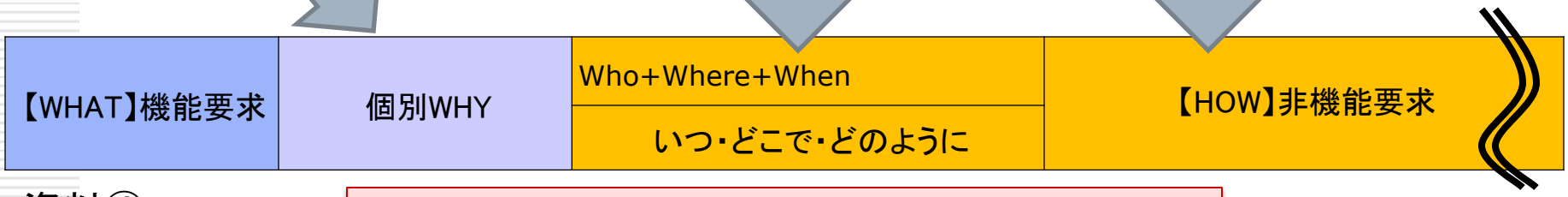
演習2-4：最初に作った「テストケース分析表」とスープカレー表を比較

- 「テストケース分析表」と、スープカレー表に記入した内容を比較し、隣同士で意見交換してください
  - 違いがありますか？
  - 違いの理由はどこにあると思いますか？

比較方法

資料③

このテストケースの目的	使用したテスト技法	想定したユーザー（立ち位置）	想定した非機能



資料④
スープカレー表

作業時間：今から5分間

演習2-4：終了

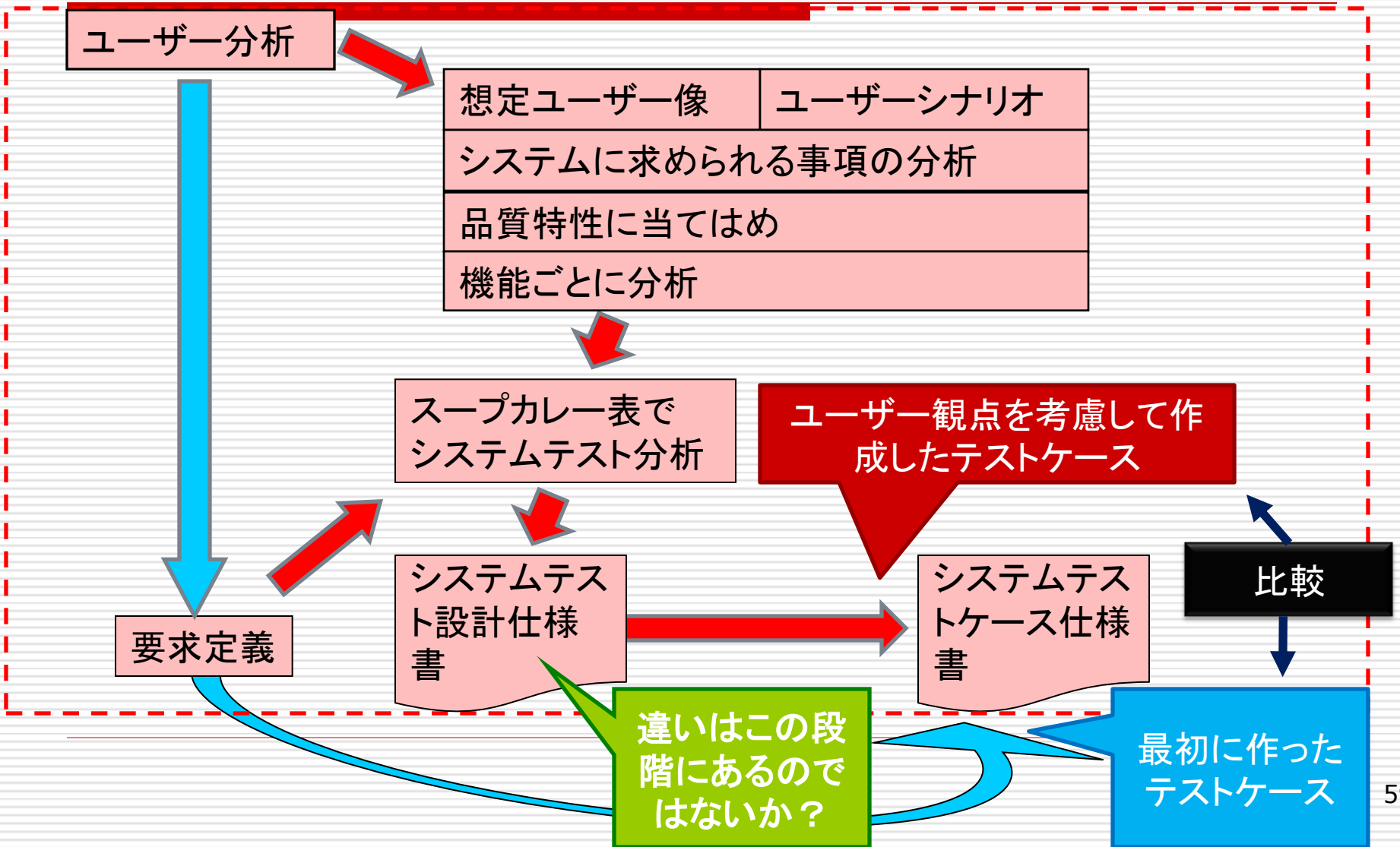
どのような違いがありましたか？

- 両方の内容の違いがテストケースの違いになっていると思います
- テスト設計書の段階で、背景も含めた具体的な「ユーザー」を想定することによって、システムテスト仕様書になったと思います

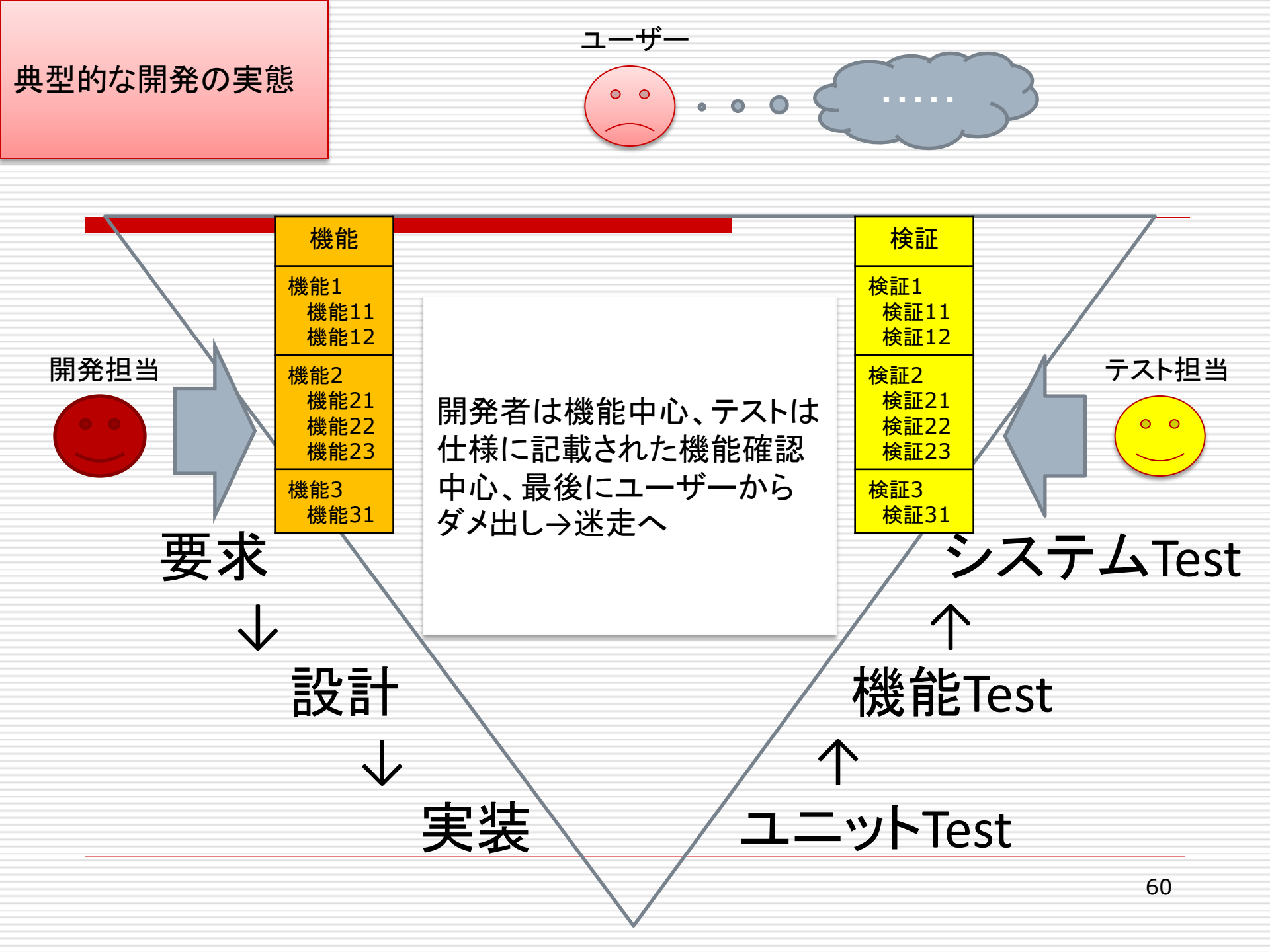
STEP4：まとめ

- 演習結果サマリ
- スープカレー表の意義
- オプション
- まとめ
- 最後に

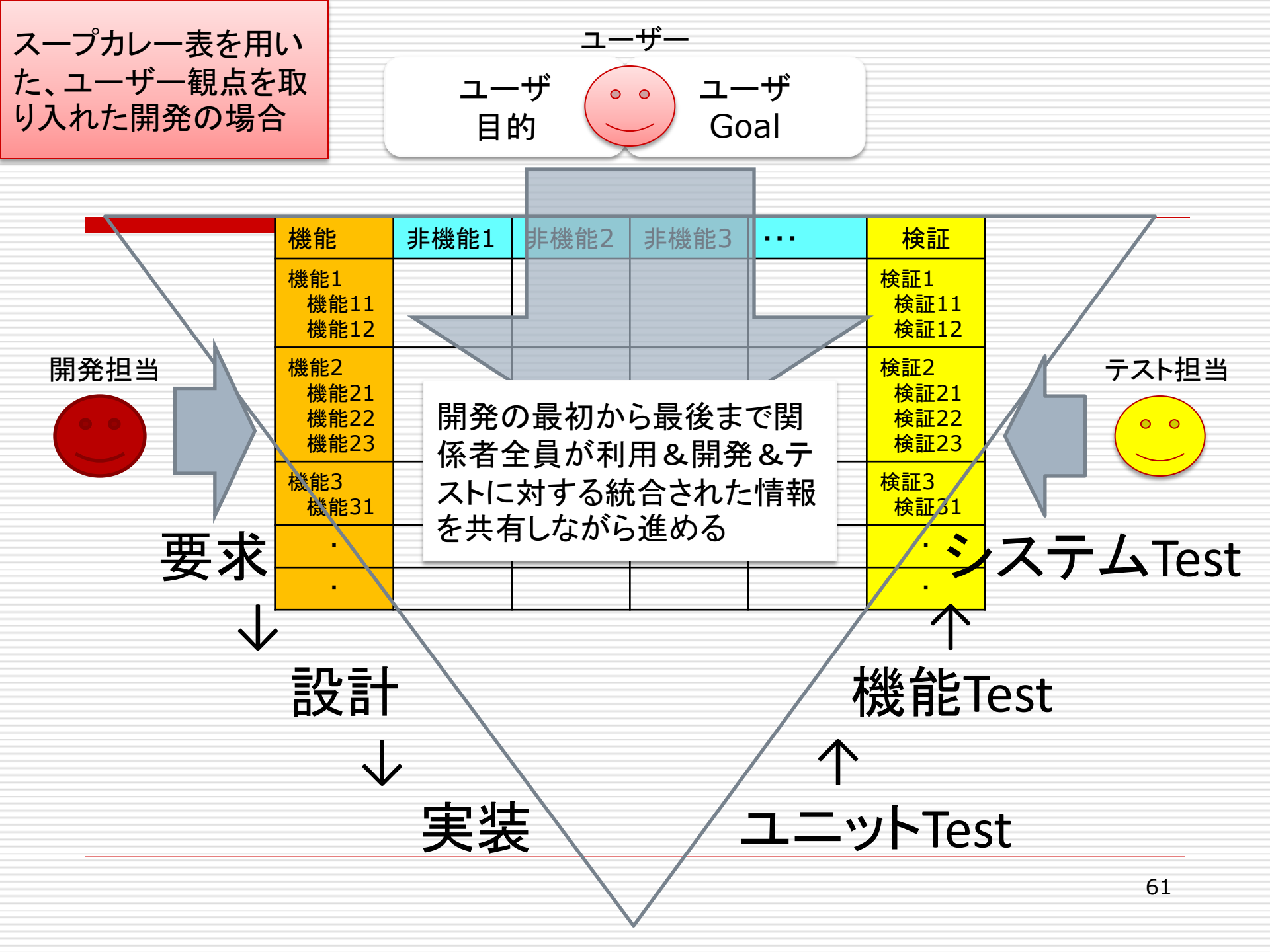
演習結果サマリ


ユーザー分析
想定ユーザー像
ユーザーシナリオ
システムに求められる事項の分析
品質特性に当てはめ
機能ごとに分析
スープカレー表で
システムテスト分析
ユーザー観点を考慮して作成したテストケース
システムテスト設計仕様書
システムテストケース仕様書
比較
要求定義
違いはこの段階にあるのではないか？
最初に作ったテストケース

典型的な開発の実態


ユーザー
……
機能
機能1
機能11
機能12
機能2
機能21
機能22
機能23
機能3
機能31
検証
検証1
検証11
検証12
検証2
検証21
検証22
検証23
検証3
検証31
開発担当
テスト担当
開発者は機能中心、テストは仕様に記載された機能確認中心、最後にユーザーからダメ出し→迷走へ
要求
↓
設計
↓
実装
システムTest
↑
機能Test
↑
ユニットTest

スープカレー表を用いた、ユーザー観点を取り入れた開発の場合



参考）スープカレー表：オプション追加

- 横軸に「開発者からの観点」を追加記入することも可能
- 共通の観点や、ある機能で思いついた観点を全機能でチェックすることが出来る
- 組合せにより、発想が広がる

【WHAT】機能要求		個別WHY	開発者観点		
大項目	中項目	機能の目的	境界値	エラーチェック	0／NULL
読書データ登録画面	本のタイトル表示	どの本を読んだか登録するため／確認するため	・本のタイトルの名称の長さの境界値		・データ上タイトルが未登録だった場合は？
	レベル表示	読んだ本の難易度を登録するため／確認するため	・レベルの境界値		・データ上レベルが未登録だった場合は？ ・データ上レベルが0だった場合は？
	日付入力	本を読んだ日を登録する／入力するため		無効な日付の入力（年・月・日・月末判定）	未入力で登録

まとめ

- ユーザーのゴールとシナリオから非機能要求を分析し、「開発」「テスト」に次ぐ第三の観点軸とする
- テスト仕様を作成する際に暗黙的に行ってきた「テスト分析」を明示的に行って、テスト方針を共有する
- 「テスト分析」と「テスト設計」の品質によって、テストケースの品質が決まる

結論

- テストエンジニアは、機能に対するテストだけではなく、**想定ユーザーとそのゴール**から、どのような**機能**と**非機能**を考える必要がある
- ユーザーが求める品質を、「開発者」「テストエンジニア」の両者が開発の早い段階から共有することで、**魅力あるソフトウェア**が生まれる

最後に

- 駆け足になって申しわけありませんでした
- 時間の都合上、まだまだご説明できない箇所が山ほどあります
  - 想定ユーザー像を作る「ペルソナ法」とは？
  - UIシナリオとは？
  - スープカレー方式の「オプション」は他にあるのか？
  - FV表（HAYST法®）に繋げるにはどうすれば？
  - 品質特性とは？
  - この取組みの続きはどうなる？

続きはコチラ

- 気になる方は、是非「TEF道」へ！
- 継続的に「ソフトウェアテスト」を追求したい方は、是非コチラまで！！！

tef-do-ml-owner@yahoogroups.jp

ありがとうございました