ソフトウエア品質シンポジウム2018

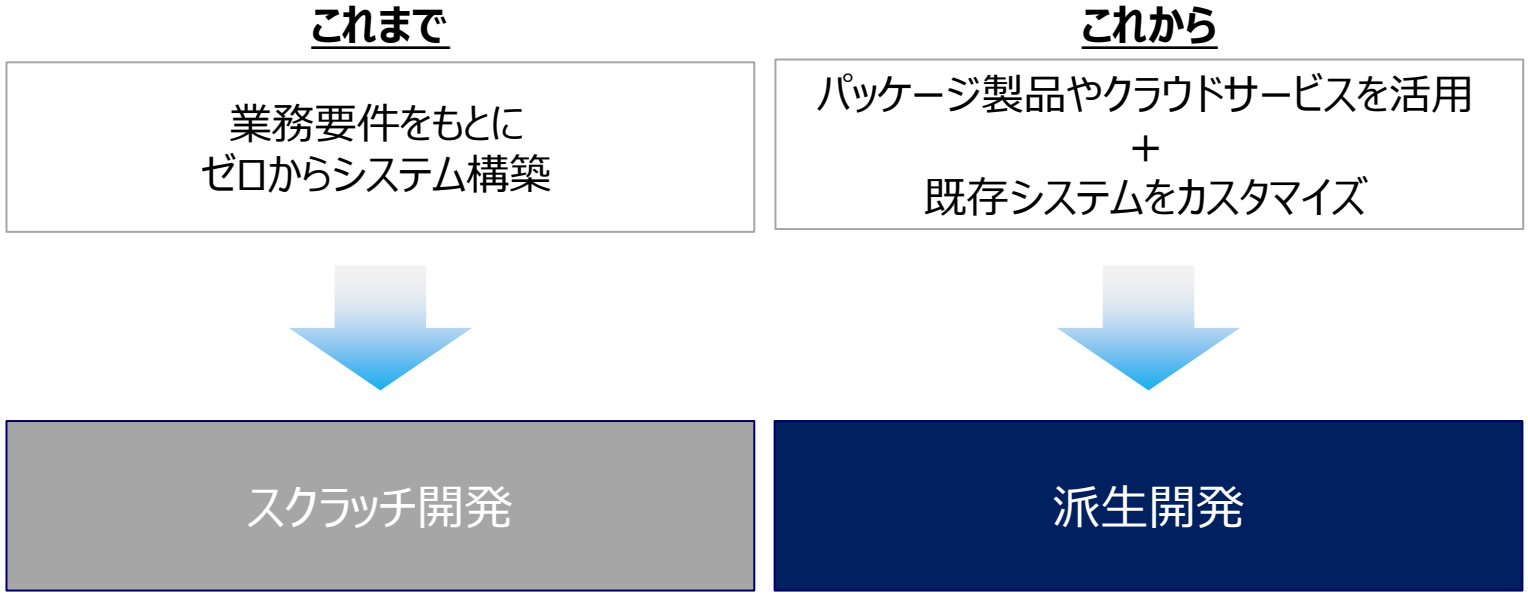
派生開発における
テストケースの第三者検証について

2018年9月13日
フューチャーアーキテクト株式会社
ファイナンシャルビジネスグループ
長坂昭彦
a.nagasaka.5b@future.co.jp

目次

1. 背景と課題

■ エンタープライズ領域のシステム構築は『作る』時代から『組合せて使う』時代へ



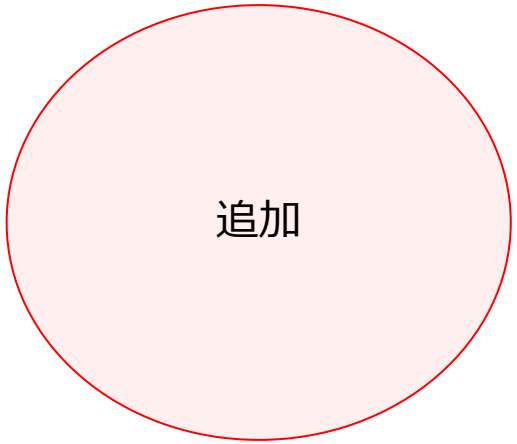
パッケージ製品やクラウドサービスを活用したり、既存システムをカスタマイズする
「派生開発」が今後のシステム構築の主流となると想定

1．1　派生開発のメリット・デメリット

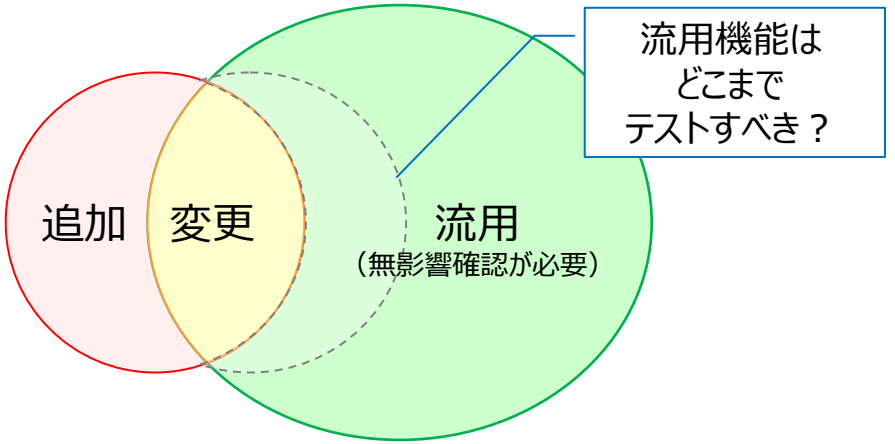
■ 派生開発は開発コスト・期間が軽減できる一方、テスト範囲の見極めが難しい

	スクラッチ開発	派生開発
メリット	抜本的な業務・システム刷新が可能 （既存のしがらみなし）	スクラッチ開発に較べて 開発コスト、期間を軽減
デメリット	派生開発に較べて 開発コスト、期間が増加	流用機能の無影響確認が必要 テスト範囲の見極めが難しい （テスト過小：品質低下、テスト過大：コスト圧迫）

機能内訳
（追加/変更/流用）



開発範囲＝テスト範囲



開発範囲≠テスト範囲

無影響確認の実施範囲とテストボリュームの妥当性を検証する事が
派生開発の品質確保には重要

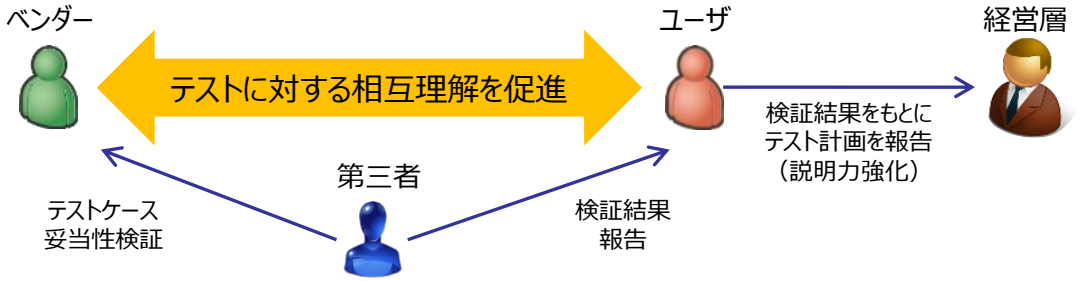
1．2　第三者検証の有効性

■ 開発ベンダーとユーザ企業の間には経験・意識にギャップがある事が多い

	開発ベンダー	ユーザ企業
システム構築経験	特定システムの経験豊富 既存システム仕様を熟知	システム担当者は定期的に入替え 既存システム仕様は概要レベルで理解
テストケースに対する意識	勘と経験に基づいてケース作成 （選定ケースが特定業務に偏る場合も）	テストケースは客観的に評価したい （テストの妥当性を経営層へ説明する必要あり）

ギャップ解消アプローチ

- 開発見積りにおいて普及が進む第三者検証をテストケースに応用
- 第三者が橋渡し役となり、開発ベンダーとユーザ企業の相互理解を促進



開発ベンダーが作成したテストケースの妥当性をユーザ企業が判断する事が困難な場合
第三者検証を活用するのも一案

1. 背景と課題
2. 課題解決の方向性
3. 実施結果
4. 効果
5. 今後の展開

2. 課題解決の方向性

■ ベンダーが作成したテストケースと第三者が作成したテストケースを比較し検証

検証観点	● ケース数は妥当か ● 選定されたテストケースは妥当か

■ テストケース比較における課題と取組み

	課題	取組み
ケース粒度不一致	ベンダー作成のテストケースと 第三者作成のテストケースは 粒度が不一致	双方のテストケースを 最小単位に分解して 粒度を統一
ケース数妥当性	第三者が作成するテストケース数の 客観性が不明瞭 (ケース数が乖離した場合主張が折り合わない場合も)	テストケース数を複数の方法で見積り 客観性を確保 (2点見積り)
テストケース作成時間	第三者のテストケース作成に 時間がかかる	上流工程の設計書から テストケースを自動生成 (結合：CRUD表、ST：業務フロー)

今回は結合テストにおける取組みをご紹介

2．1　テストケースの粒度統一

■ 結合テストの目的からテストケースの最小粒度を定め、ベンダー作成のテストケースを分解

結合テストの目的
- 機能間の関連を検証

着眼点
- 各機能はデータを介して結びついている（結合）

粒度定義
- データアクセスをともなう機能の組合せをテストケースの最小粒度と定義し
  ベンダー作成のテストケースを分解

分解前

テストケース
　テストブロックA
　　画面1→画面2→画面3→画面4
　テストブロックB
　　画面2→画面3→画面5

テスト仕様書

テストケース分解

分解後

テストケース
画面1 → 画面2
画面2 → 画面3
画面3 → 画面4
画面2 → 画面3
画面3 → 画面5

テストケース（ベンダー）

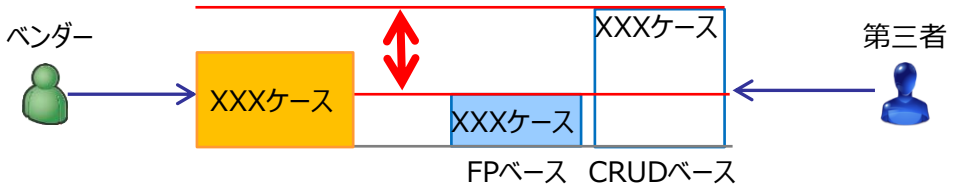
テストケース数　**2**　　**4**

テストケースの粒度は一般的な指標がないためテストケース作成者に依存
テスト粒度の基準を定め、同一基準でテストケースを検証

2．2　テストケース数の2点見積り

■ 開発見積の検証手法を応用し、テストケース数を複数の方法で見積り

見積方法	● システム規模（FP）とテスト密度からテストケース数を算出 ● CRUD表からテストケース数を算出

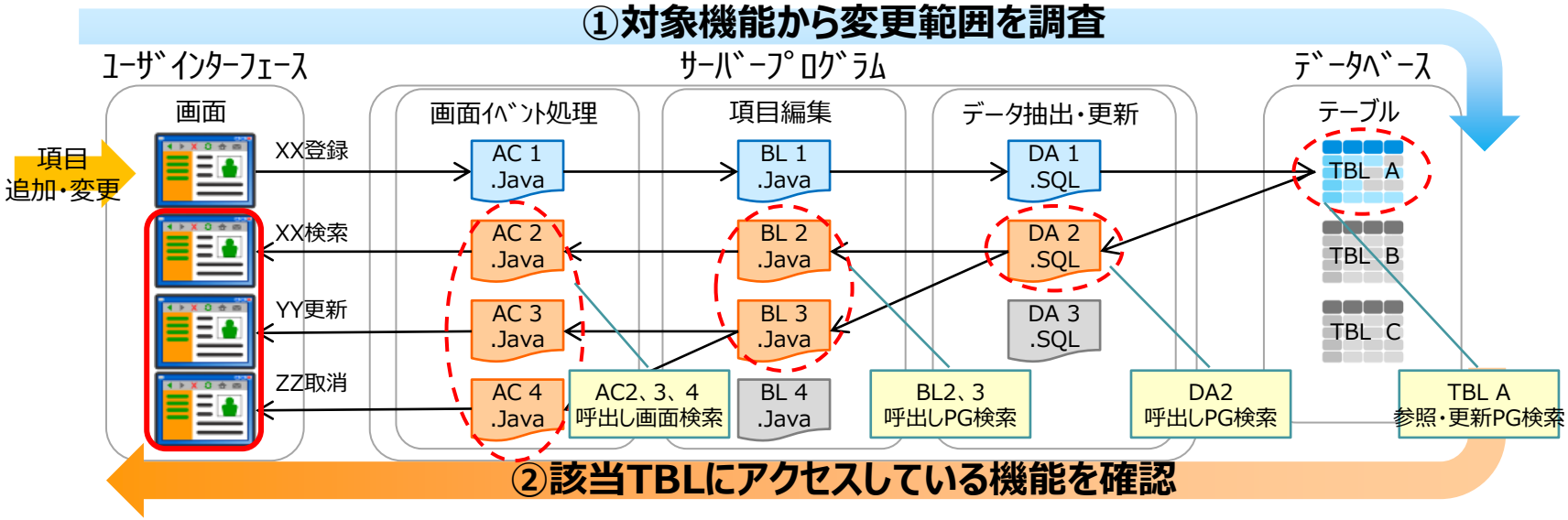


	FPベース	CRUDベース
インプット	要件定義工程で作成された 各種一覧資料	CRUD表
ケース数 見積方法	システム規模（FP）÷　テスト密度	CRUD表から算出
見積 可能時期	要件定義 終了時点	外部設計～内部設計 終了時点

開発見積の検証で普及している2点見積りの考え方をテストケース見積りに応用
テストケース数を幅を持って検証する事で客観性を確保

2．3　テストケースの自動生成

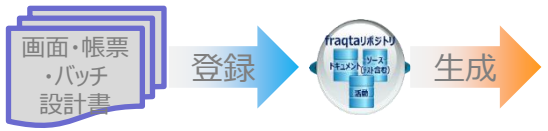
■ 設計書とプログラムが乖離している場合、テストケースの自動生成は困難



テストケース作成

- 複雑・非効率
- 属人的（PG構造の理解が必要）
- ケース漏れリスクあり

■ 設計書とプログラムが一致している場合、CRUD表からテストケースを自動生成する事が可能



CRUD表		テーブル		
		A	B	C
画面	XX登録	C		
	XX検索	R		
	YY更新	R	C	
	ZZ取消	R	U	C

生成

テストケース

<TBL A>
XX登録 → XX検索
XX登録 → YY更新
XX登録 → ZZ取消

<TBL B>
YY更新 → ZZ取消

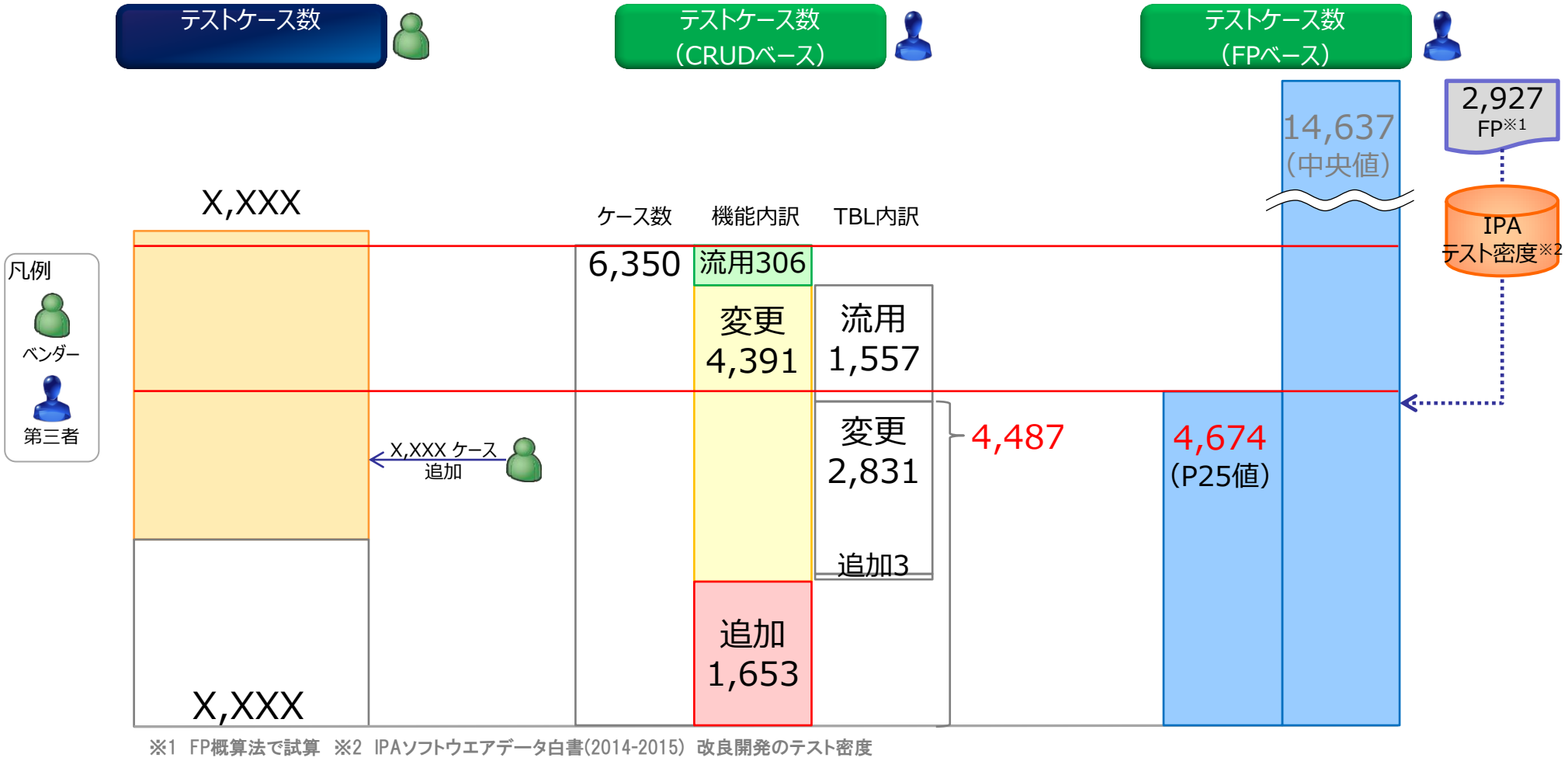
- 単純・効率的
- 標準的（PG構造の理解は不要）
- ケース漏れリスクなし

設計書とプログラムが一致している場合、CRUD表からテストケースの自動生成が可能

1. 背景と課題
2. 課題解決の方向性
3. 実施結果
4. 効果
5. 今後の展開

3. 実施結果

- 該当システムに必要な結合テストケース数は4,674～6,350相当と試算



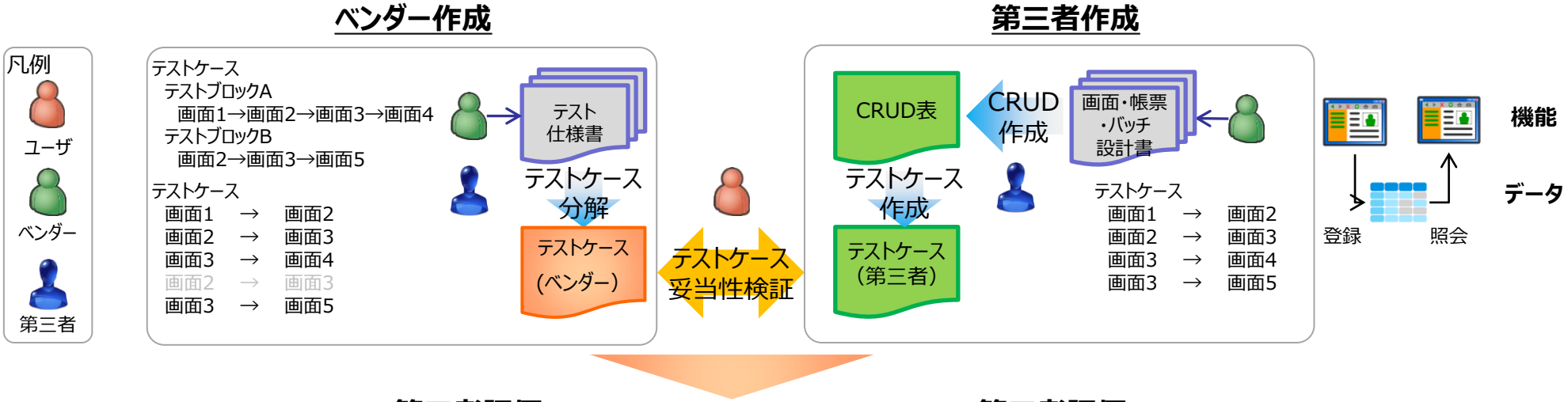
※1 FP概算法で試算 ※2 IPAソフトウエアデータ白書(2014-2015) 改良開発のテスト密度

最低限必要なテストケースは追加機能＋変更機能がアクセスするTBLが追加・変更となるケース
（FPベースのP25値と同水準）

1. 背景と課題
2. 課題解決の方向性
3. 実施結果
4. 効果
5. 今後の展開

4. 効果

■ テスト開始後に都度ケースを追加するプロセスからテスト開始前にケースを確定するプロセスに変更
→テスト開始前に不要ケースを除外しベンダー/ユーザ双方で合意



	第三者評価 実施前	第三者評価 実施後
ケース作成方法	ケース作成者の勘と経験に依存	機能 × データの組合せ （不足ケースを追加）
作成ケース	一部の結合ケース	全部の結合ケース
ケース確定方法	テスト開始後に場当り的にケース追加（足し算）	テスト開始前に不要ケースを除外（引き算）
網羅性	不明瞭	明瞭

テストケースの第三者検証を実施する事により、
第三者検証に耐えうるテスト網羅性の確保とテスト過小に伴う品質低下リスクの抑制が期待できる

1. 背景と課題
2. 課題解決の方向性
3. 実施結果
4. 効果
5. 今後の展開

5.　今後の展開

■ 今後は更新パターンに加えて、データパターンを考慮したテストケースの自動生成を目指す

	現状	今後の展開
ケース作成方法	機能　×　データ	機能　×　データ　×　データパターン
インプット	CRUD表	CRUD表　＋　ディシジョン表
対象ケース	トランザクション更新が中心	トランザクション　＋　マスタ

より精緻なテストケースとするにはデータパターンを組合せたテストケースの精緻化が必要
今後はデータパターンを加味したテストケースを自動生成し妥当性検証を高度化していきたい

ご清聴ありがとうございました

本書において使用されている商標は、フューチャー株式会社または他社の登録商標または登録出願中の商標です。