JaSST’17 Tokyo

大規模業務システムにおける
再利用可能なテスト自動化の取り組み

フューチャーアーキテクト（株）
テクノロジーイノベーショングループ
杉本 秀樹

Copyright ©2017 by Future Architect, Inc. Japan

自己紹介

【名前】 杉本　秀樹　（すぎもと　ひでき）

【所属】 フューチャーアーキテクト（株）

テクノロジーイノベーショングループ所属

シニアスペシャリスト

【仕事】 システム開発を効率化するもろもろの整備

Copyright ©2017 by Future Architect, Inc. Japan

会社紹介

- 社名　：　フューチャーアーキテクト株式会社
  (英文表記：Future Architect, Inc.)
- 業種　：　ITコンサルティング業
- 設　立　：　1989年11月28日（設立27周年）
- 上　場　：　2002年6月　東証1部上場

- 代表者：　代表取締役会長　金丸恭文※
  代表取締役社長CEO　東 裕二

※内閣官房 産業競争力会議 議員、
高度情報通信ネットワーク社会推進戦略本部 本部員、
内閣府 規制改革会議 委員･同会議　農業WG座長、
社団法人　経済同友会　副代表幹事･政策懇談会　委員長、
公益財団法人総合研究開発機構（NIRA）代表理事、新経済連盟　理事を務める。

Challenge Together
経営とITをデザインする、フューチャーアーキテクト。

Copyright ©2017 by Future Architect, Inc. Japan

ITコンサルティング会社がシステム開発？

- フューチャーアーキテクトは「ITのコンサルティング」ではなく、「ITを活かした経営の実現」を行います
- ゴールはお客様の未来価値増大です。「ITをどうする」ではなく、「ITでどうする」です
- グランドデザイン（基本構想、システム化計画）から開発、運用に至るまでの各フェーズをお客様と一体となって推進します
- 中立の立場から豊富な業務ノウハウと高い技術力とを駆使してハードウェア・ソフトウェアの最適化にあたり、付加価値の高いプロジェクト推進を支援します

Copyright ©2017 by Future Architect, Inc. Japan

アジェンダ

- 弊社が行ってきた単体テストの問題
- 問題に対する対策
- 対策に対する評価
- 今後の展望

Copyright ©2017 by Future Architect, Inc. Japan

始めに

Copyright ©2017 by Future Architect, Inc. Japan

始めに：対象とする領域

- 弊社では長年にわたり顧客企業向けの大規模業務システムの構築を行ってきました

- 大規模業務システムの構築には以下のような特徴があります
  - 作成する機能が多い（数百～数千）
  - 機能の種類が多い（画面／帳票／バッチ処理／データ連携など）
  - ウォーターフォールモデルで構築されることが多い
  - 開発にかかわるメンバーが多く、スキルレベルのばらつきも大きい

- 今回はこのような大規模業務システムの構築を対象として、複数のプロジェクトで再利用可能な単体テスト自動化の取り組みについて発表します

Copyright ©2017 by Future Architect, Inc. Japan

始めに：これまでの単体テストの問題

- これまで弊社でも各プロジェクトで単体テストの効率化に取り組んできましたが、さまざまな問題がありました

1．テスト方法が統一されていない

2．テスト結果がテスト仕様を満たしているか判断しづらい

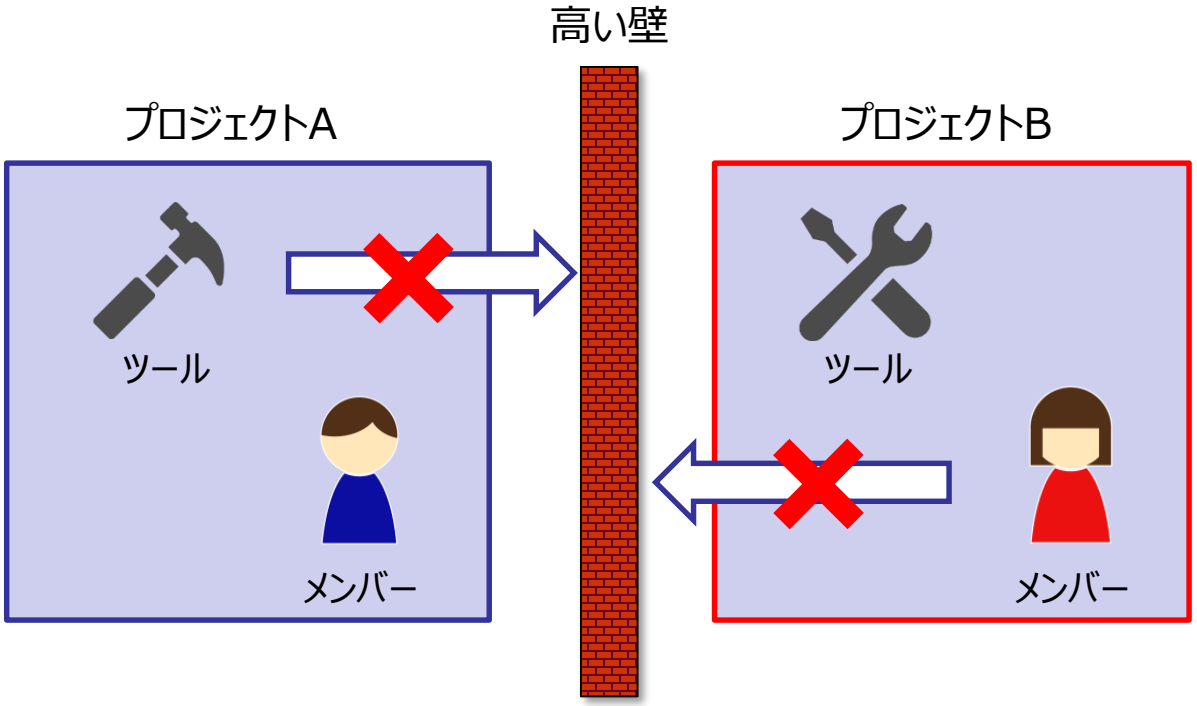
3．保守・メンテナンスコストの高さ

Copyright ©2017 by Future Architect, Inc. Japan

始めに：これまでの単体テストの問題

1．テスト方法が統一されていない

- これまではプロジェクト毎にテスト方法を検討し、ツールを準備していました
- 個々のツールは実装方法やテスト適用範囲がバラバラで、プロジェクトを跨いだツールの利用やノウハウ共有が出来ていませんでした



Copyright ©2017 by Future Architect, Inc. Japan

始めに：これまでの単体テストの問題

2． テスト結果がテスト仕様を満たしているか判断しづらい

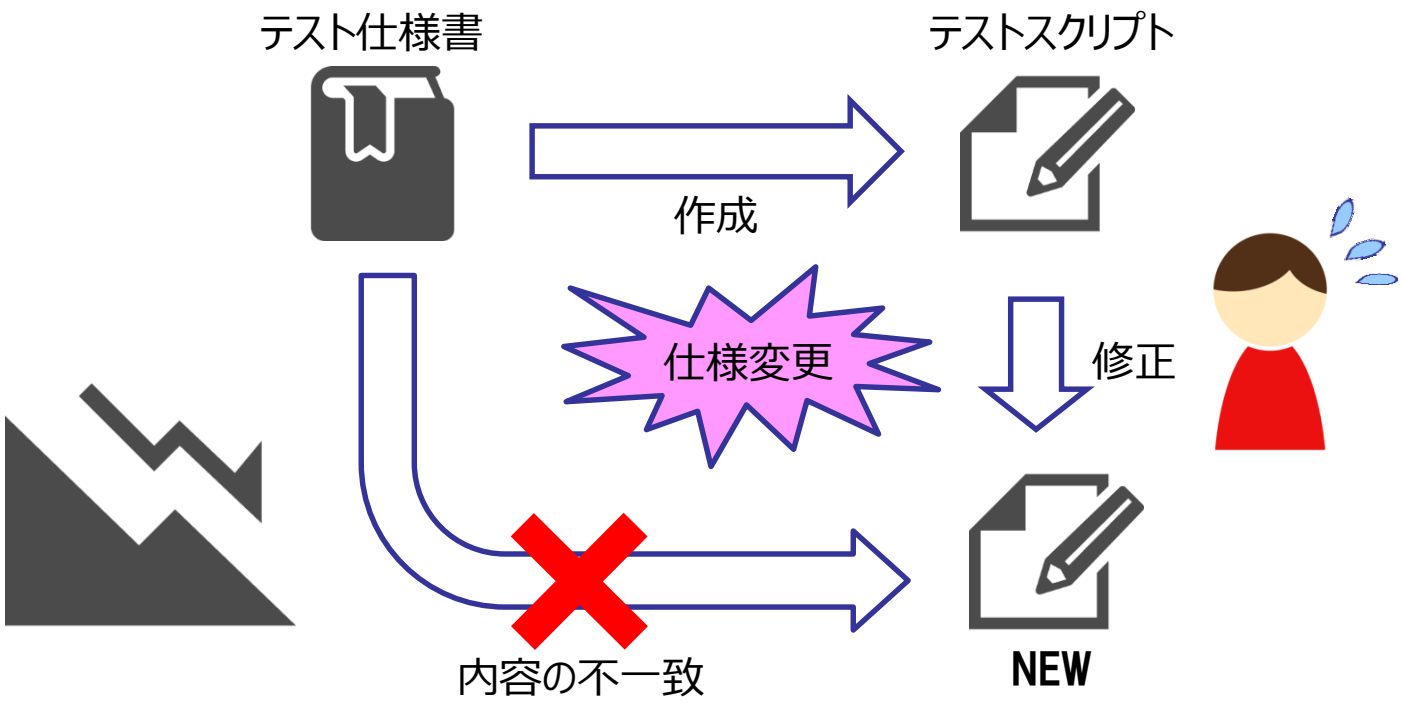
- テスト仕様書をもとにしてテストスクリプトを作成し、テストを実施する
- テストレビュー時、テストスクリプトが仕様書に記載されているテスト仕様を満たしているかどうかの確認に時間がかかる
- レビュー時間の増大 / レビューアの負荷増大

Copyright ©2017 by Future Architect, Inc. Japan

始めに：これまでの単体テストの問題

3． 保守・メンテナンスコストの高さ

- 機能が多いことでテストスクリプトも大量に作成されることになり、テストスクリプトの保守・メンテナンスに多くのコストがかかった
- テスト仕様書とテストスクリプトの内容に差異が発生しやすい
- 結果として保守のタイミングでテスト仕様書が役に立たなくなる



Copyright ©2017 by Future Architect, Inc. Japan

問題に対する対策

Copyright ©2017 by Future Architect, Inc. Japan

問題に対する対策

- 先にあげた単体テストの問題を解決するために、２つの対策を実施しました

対策１：テスト粒度と対象範囲の再設計

対策２：汎用E2Eテストツール開発

Copyright ©2017 by Future Architect, Inc. Japan

対策１：テスト粒度と対象範囲の再設計

- そもそもUnitテストって効率的ですか？
  - １つのクラスを動かすためにたくさんのモックを作るのって非効率では？

- Unitテストが出てきたときとは状況が変わってきているのでは？
  - フレームワークの進化
    - ✓ アプリケーションフレームワークの利用により実装コード量は減少
    - ✓ 代わりに単一のクラスで機能が完結することが少なくなってきている
  - 開発を取り巻く環境の変化
    - ✓ PC性能の向上やDockerなどの仮想化技術により、1台のPC上でDBも含めた実行環境を構築できるようになってきた

Unitより大きな粒度でテストを行う

Copyright ©2017 by Future Architect, Inc. Japan

対策１：テスト粒度と対象範囲の再設計

- プログラム言語以外で実装される機能のテストはどうする？
  - 例えば
    - ✓ 帳票のテストは？
    - ✓ ファイルに記述されたSQLの動作が正しいことのテストは？
    - ✓ ETLツールに設定した内容が正しいことのテストは？

自動テストの対象とする範囲を明確に！

Copyright ©2017 by Future Architect, Inc. Japan

対策１：テスト粒度と対象範囲の再設計

- 検討結果

Unitより大きな粒度でテストを行う

↓

テスト粒度を「機能単位」とする

自動テストの対象とする範囲を明確に！

↓

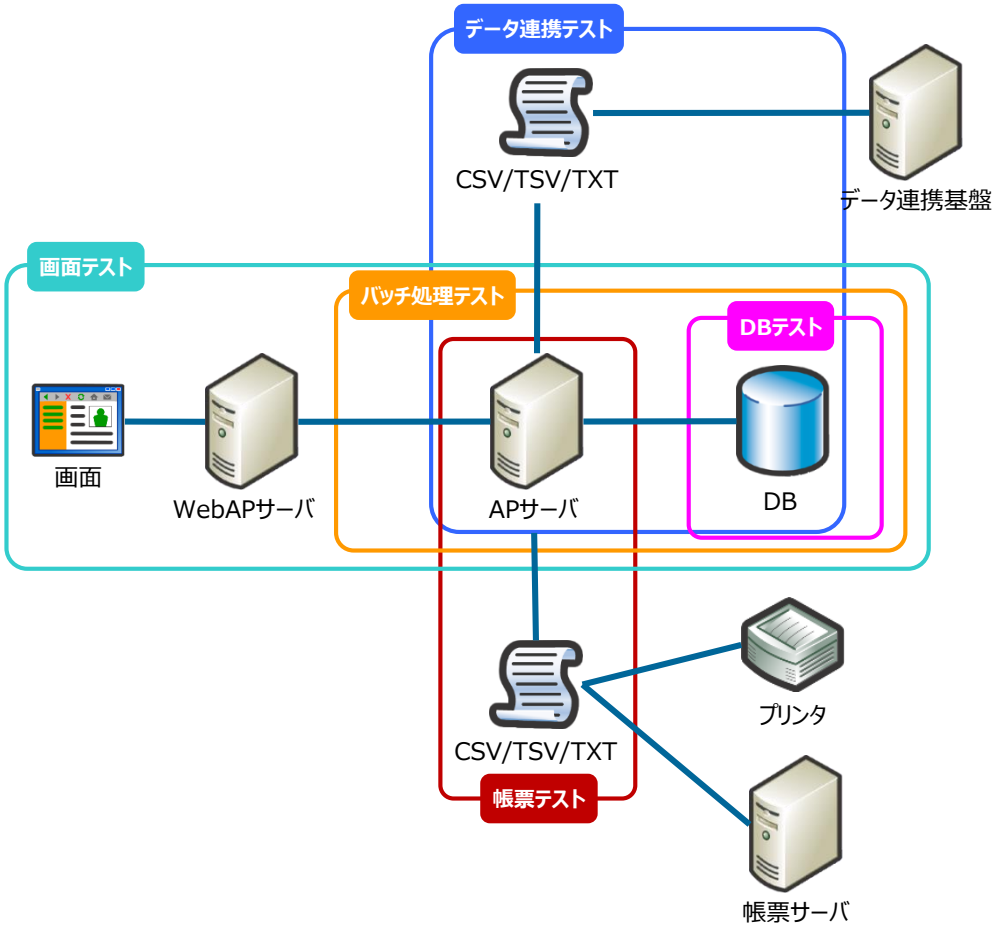
DB/ファイルへのOUTPUTがあるものを対象とする

Copyright ©2017 by Future Architect, Inc. Japan

対策１：テスト粒度と対象範囲の再設計

テスト粒度と対象範囲の再設計結果

機能単位	テスト内容
画面テスト	画面の挙動と表示内容、DBデータ編集がある場合はその内容を検査する
帳票テスト	生成する帳票のINPUTとなるテキストデータ（CSV/TSV/独自形式）の内容を検査する 帳票レイアウトや出力項目のマッピングについては別途目視での確認を行う
データ連携テスト	データ連携処理の実行結果としてDBデータやファイルの編集がある場合はその内容を検査する 合わせてデータ連携処理の状態（連携ステータスなど）を検査する
バッチ処理テスト	バッチ処理の実行結果としてDBデータやファイルの編集がある場合はその内容を検査する 合わせてバッチ処理の戻り値（ステータスコードなど）を検査する
DBテスト	SQLやストアドプロシージャの実行結果としてDBデータの編集内容を検査する

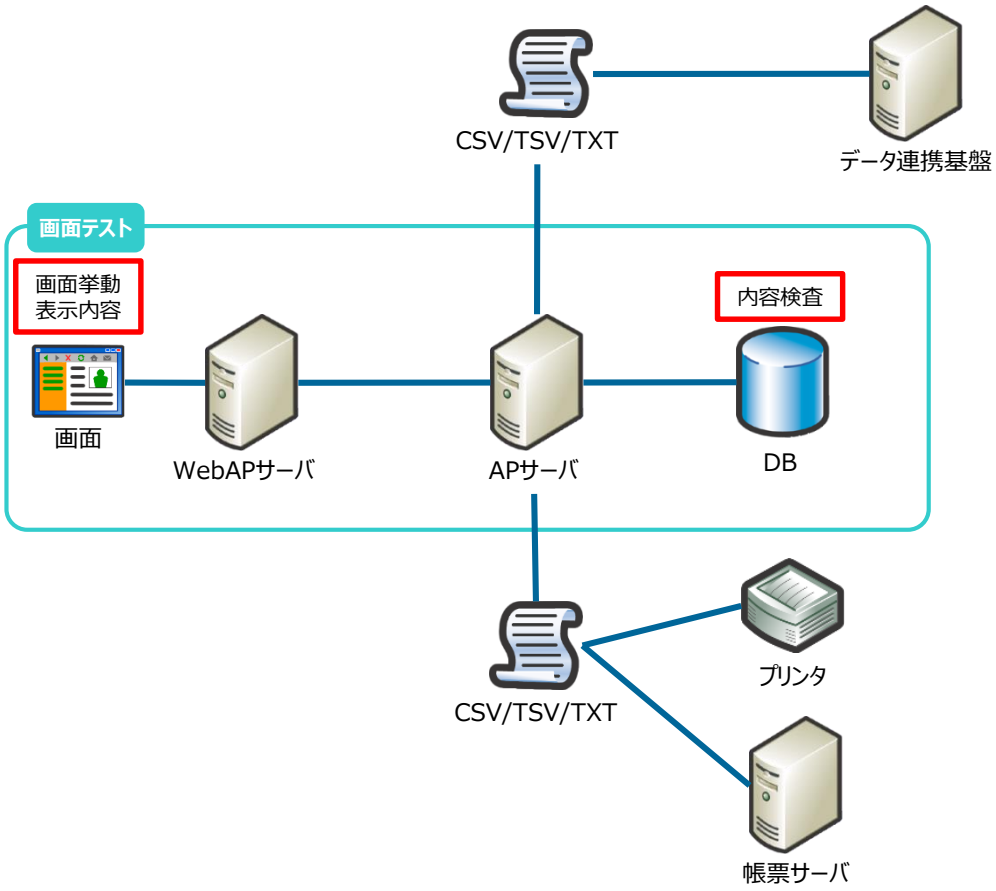


Copyright ©2017 by Future Architect, Inc. Japan

対策１：テスト粒度と対象範囲の再設計

テスト粒度と対象範囲の再設計結果

機能単位	テスト内容
画面テスト	画面の挙動と表示内容、DBデータ編集がある場合はその内容を検査する
帳票テスト	生成する帳票のINPUTとなるテキストデータ（CSV/TSV/独自形式）の内容を検査する 帳票レイアウトや出力項目のマッピングについては別途目視での確認を行う
データ連携テスト	データ連携処理の実行結果としてDBデータやファイルの編集がある場合はその内容を検査する 合わせてデータ連携処理の状態（連携ステータスなど）を検査する
バッチ処理テスト	バッチ処理の実行結果としてDBデータやファイルの編集がある場合はその内容を検査する 合わせてバッチ処理の戻り値（ステータスコードなど）を検査する
DBテスト	SQLやストアドプロシージャの実行結果としてDBデータの編集内容を検査する

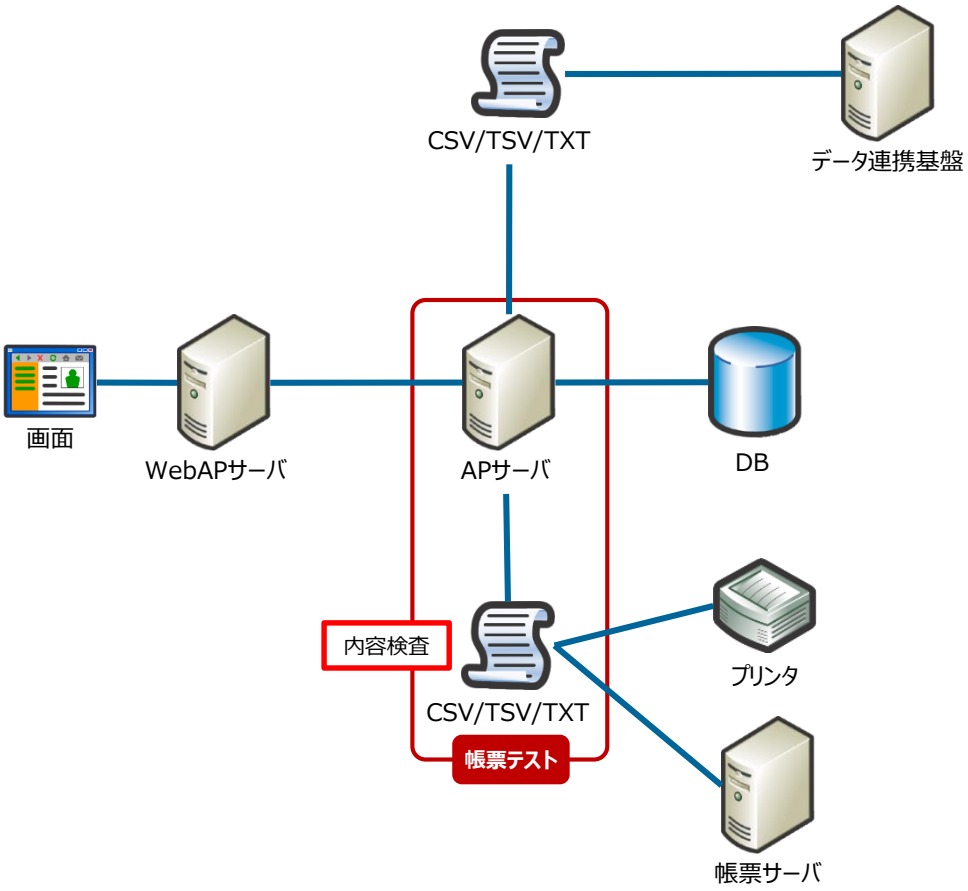


Copyright ©2017 by Future Architect, Inc. Japan

対策１：テスト粒度と対象範囲の再設計

テスト粒度と対象範囲の再設計結果

機能単位	テスト内容
画面テスト	画面の挙動と表示内容、DBデータ編集がある場合はその内容を検査する
帳票テスト	生成する帳票のINPUTとなるテキストデータ（CSV/TSV/独自形式）の内容を検査する 帳票レイアウトや出力項目のマッピングについては別途目視での確認を行う
データ連携テスト	データ連携処理の実行結果としてDBデータやファイルの編集がある場合はその内容を検査する 合わせてデータ連携処理の状態（連携ステータスなど）を検査する
バッチ処理テスト	バッチ処理の実行結果としてDBデータやファイルの編集がある場合はその内容を検査する 合わせてバッチ処理の戻り値（ステータスコードなど）を検査する
DBテスト	SQLやストアドプロシージャの実行結果としてDBデータの編集内容を検査する

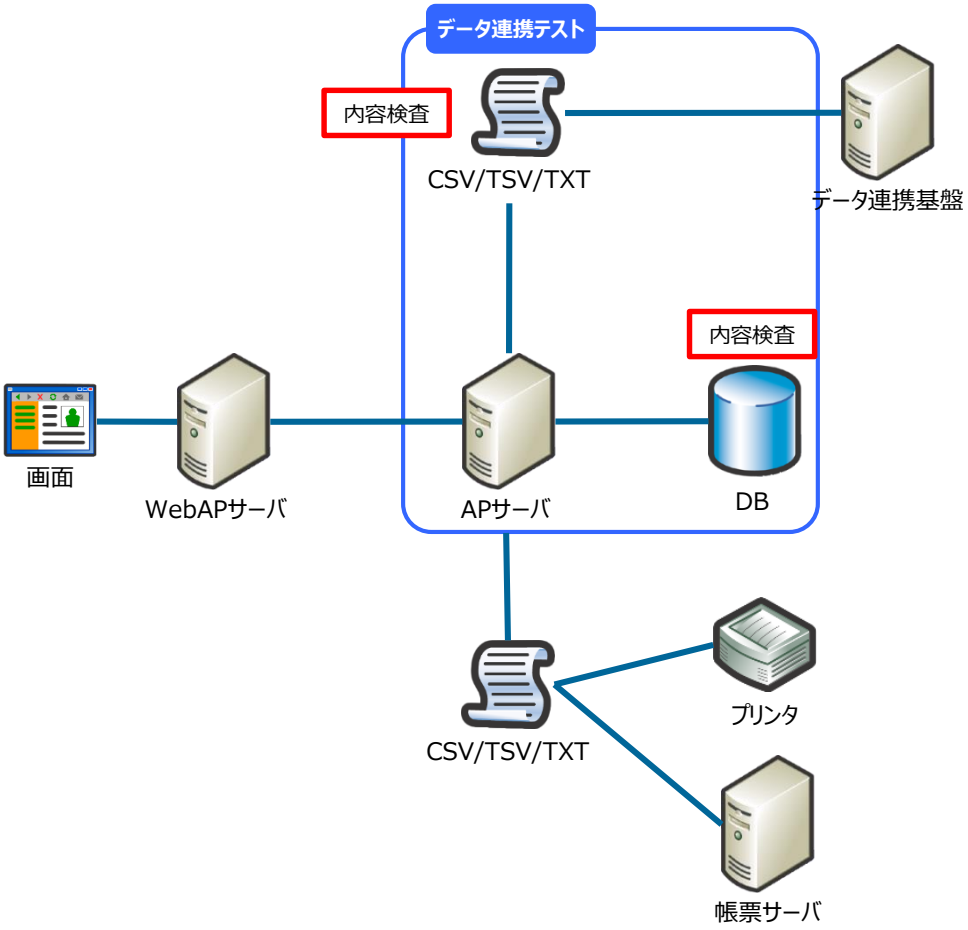


Copyright ©2017 by Future Architect, Inc. Japan

対策１：テスト粒度と対象範囲の再設計

テスト粒度と対象範囲の再設計結果

機能単位	テスト内容
画面テスト	画面の挙動と表示内容、DBデータ編集がある場合はその内容を検査する
帳票テスト	生成する帳票のINPUTとなるテキストデータ（CSV/TSV/独自形式）の内容を検査する 帳票レイアウトや出力項目のマッピングについては別途目視での確認を行う
データ連携テスト	データ連携処理の実行結果としてDBデータやファイルの編集がある場合はその内容を検査する 合わせてデータ連携処理の状態（連携ステータスなど）を検査する
バッチ処理テスト	バッチ処理の実行結果としてDBデータやファイルの編集がある場合はその内容を検査する 合わせてバッチ処理の戻り値（ステータスコードなど）を検査する
DBテスト	SQLやストアドプロシージャの実行結果としてDBデータの編集内容を検査する

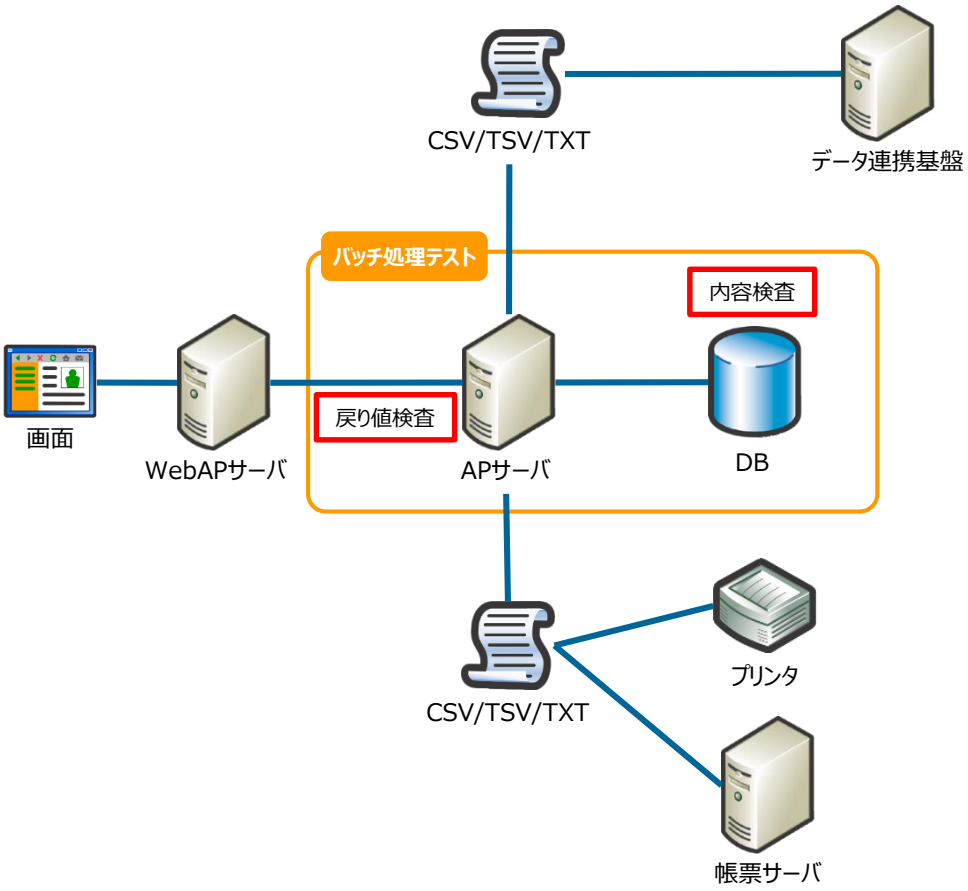


Copyright ©2017 by Future Architect, Inc. Japan

対策１：テスト粒度と対象範囲の再設計

テスト粒度と対象範囲の再設計結果

機能単位	テスト内容
画面テスト	画面の挙動と表示内容、DBデータ編集がある場合はその内容を検査する
帳票テスト	生成する帳票のINPUTとなるテキストデータ（CSV/TSV/独自形式）の内容を検査する 帳票レイアウトや出力項目のマッピングについては別途目視での確認を行う
データ連携テスト	データ連携処理の実行結果としてDBデータやファイルの編集がある場合はその内容を検査する 合わせてデータ連携処理の状態（連携ステータスなど）を検査する
バッチ処理テスト	バッチ処理の実行結果としてDBデータやファイルの編集がある場合はその内容を検査する 合わせてバッチ処理の戻り値（ステータスコードなど）を検査する
DBテスト	SQLやストアドプロシージャの実行結果としてDBデータの編集内容を検査する

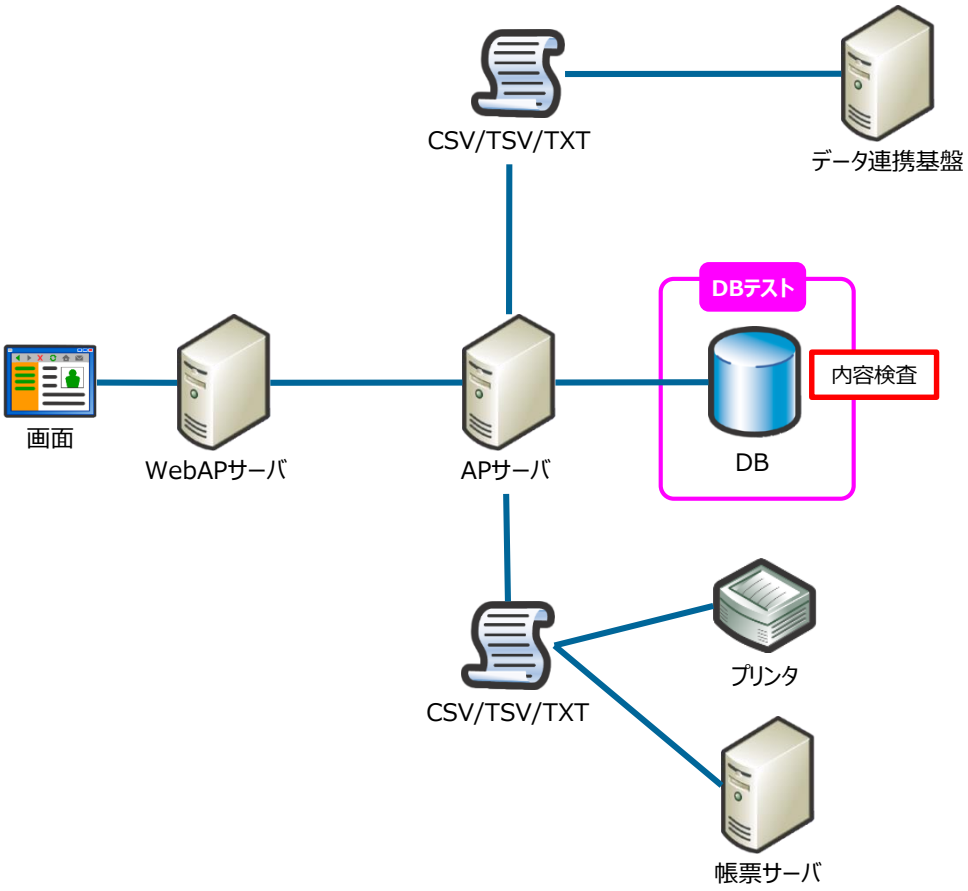


Copyright ©2017 by Future Architect, Inc. Japan

対策１：テスト粒度と対象範囲の再設計

テスト粒度と対象範囲の再設計結果

機能単位	テスト内容
画面テスト	画面の挙動と表示内容、DBデータ編集がある場合はその内容を検査する
帳票テスト	生成する帳票のINPUTとなるテキストデータ（CSV/TSV/独自形式）の内容を検査する 帳票レイアウトや出力項目のマッピングについては別途目視での確認を行う
データ連携テスト	データ連携処理の実行結果としてDBデータやファイルの編集がある場合はその内容を検査する 合わせてデータ連携処理の状態（連携ステータスなど）を検査する
バッチ処理テスト	バッチ処理の実行結果としてDBデータやファイルの編集がある場合はその内容を検査する 合わせてバッチ処理の戻り値（ステータスコードなど）を検査する
DBテスト	SQLやストアドプロシージャの実行結果としてDBデータの編集内容を検査する



Copyright ©2017 by Future Architect, Inc. Japan

対策２：E2Eテストツール開発

- 対策１で再設計したテスト粒度と対象範囲に従って自動テストを行うE2Eテストツールを開発しました

- テストツール開発に際して、これまでの経験や反省を踏まえ、以下の点を考慮しました
  1. 開発言語やテスト対象に依存しない
  2. テスト仕様書＝テストスクリプト
  3. 対象範囲の広さ
  4. テスト結果を自動で記録

Copyright ©2017 by Future Architect, Inc. Japan

1．開発言語やテスト対象に依存しない

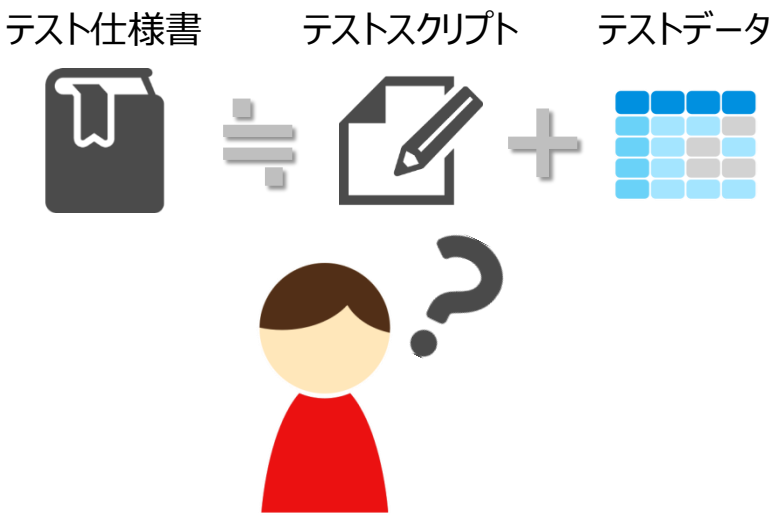
- やりたいことはテストであってテストスクリプトを書くことじゃない！
- 開発言語やテスト対象によって個別にテストツールを覚えないといけないのっておかしくないですか？

Copyright ©2017 by Future Architect, Inc. Japan

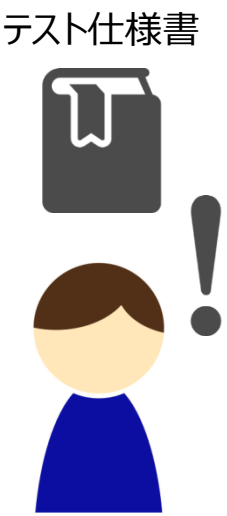
2．テスト仕様書＝テストスクリプト

- テスト仕様書をテストツールのINPUTにして、常にテスト仕様とテストが一致するようにする
- テスト実施に必要なデータ群もテスト仕様書とセットで管理
- テスト仕様書のみをレビューすれば良い状態にする

[これまで]



[これから]



Copyright ©2017 by Future Architect, Inc. Japan

3．対象範囲の広さ

- 対策１で決めた対象範囲をカバーできるものにする
  - 画面 / 帳票 / バッチ処理 / データ連携 / DB

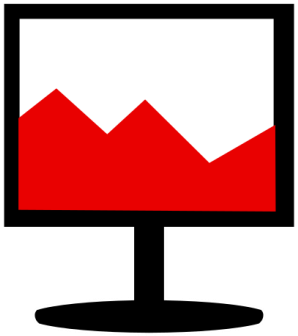
- 加えて、様々なプロジェクトで利用可能にする為、テストツールを容易に拡張できるようにする
  - プラグイン機構の導入

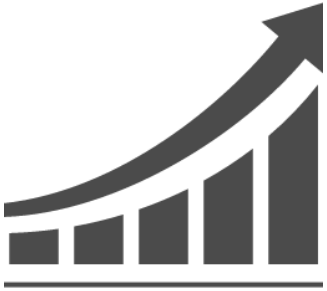
Copyright ©2017 by Future Architect, Inc. Japan

４．テスト結果の自動記録

- テスト実施結果の整理に時間と人手をかけない
  - 画面のキャプチャやDBデータの取得、そしてそれらをエビデンスとしてExcelにまとめるという不毛な作業をなくす

- テスト実施結果をいつでも確認できるように
  - 開発者からの進捗報告はあてにならない場合がある
    - ✓ 永遠に終わらない「８０％完了」報告
  - 全体の進捗把握や日々のテスト実施状況をタイムリーに確認し、問題の発生をより早く検知できるようにする





Copyright ©2017 by Future Architect, Inc. Japan

E2Eテストツール

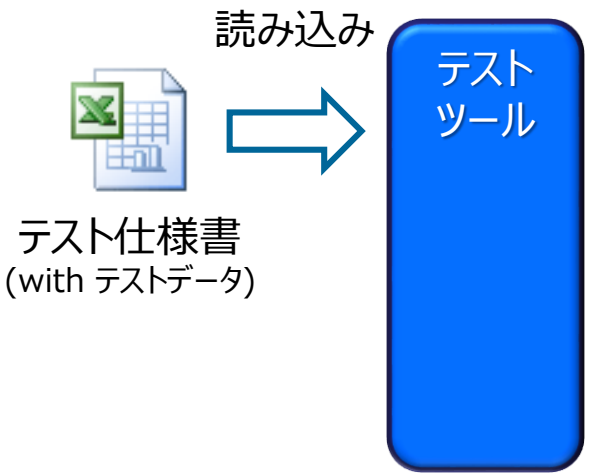
- 作成した自動テストツールは以下のように動作します

Copyright ©2017 by Future Architect, Inc. Japan

E2Eテストツール

- 作成した自動テストツールは以下のように動作します

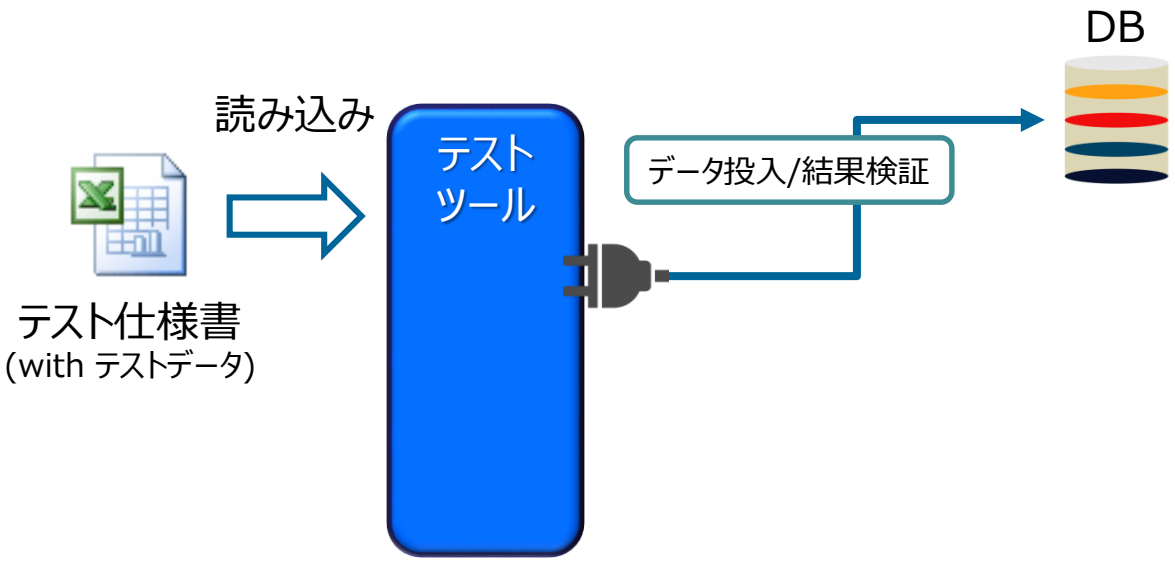
① テストツールでテスト仕様書を読み込み



Copyright ©2017 by Future Architect, Inc. Japan

E2Eテストツール

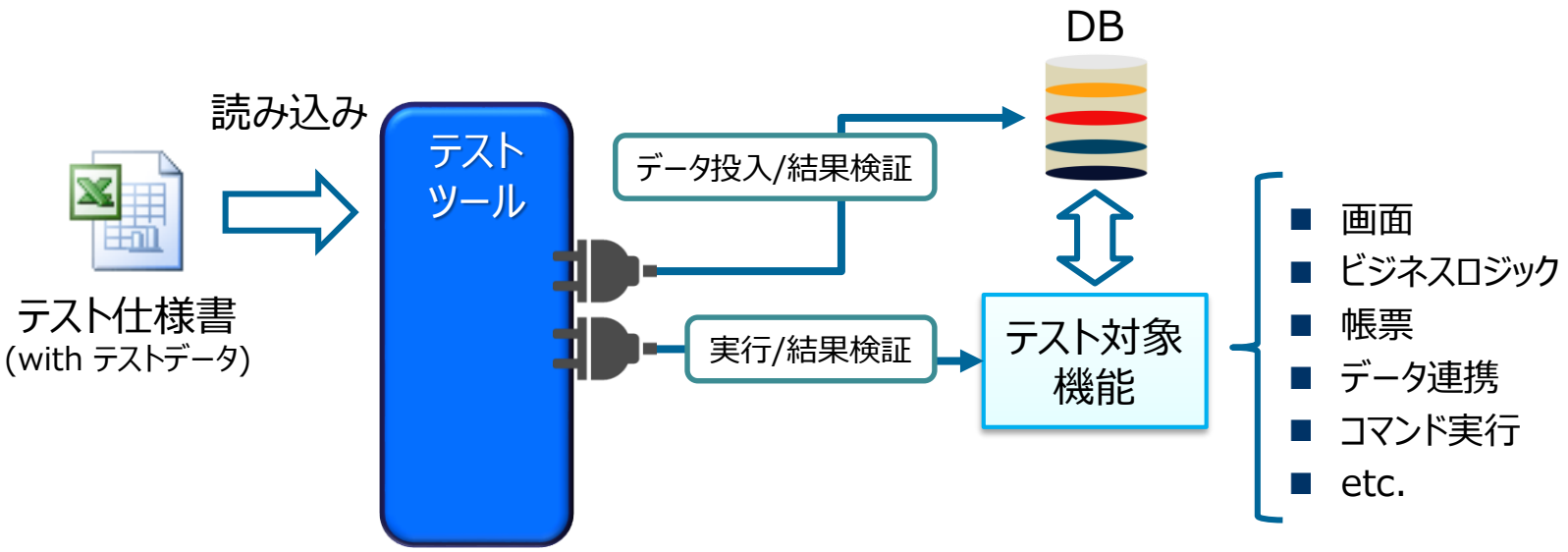
- 作成した自動テストツールは以下のように動作します
  ① テストツールでテスト仕様書を読み込み
  ② プラグイン経由でテスト仕様書に記載された操作を順に実施



Copyright ©2017 by Future Architect, Inc. Japan

E2Eテストツール

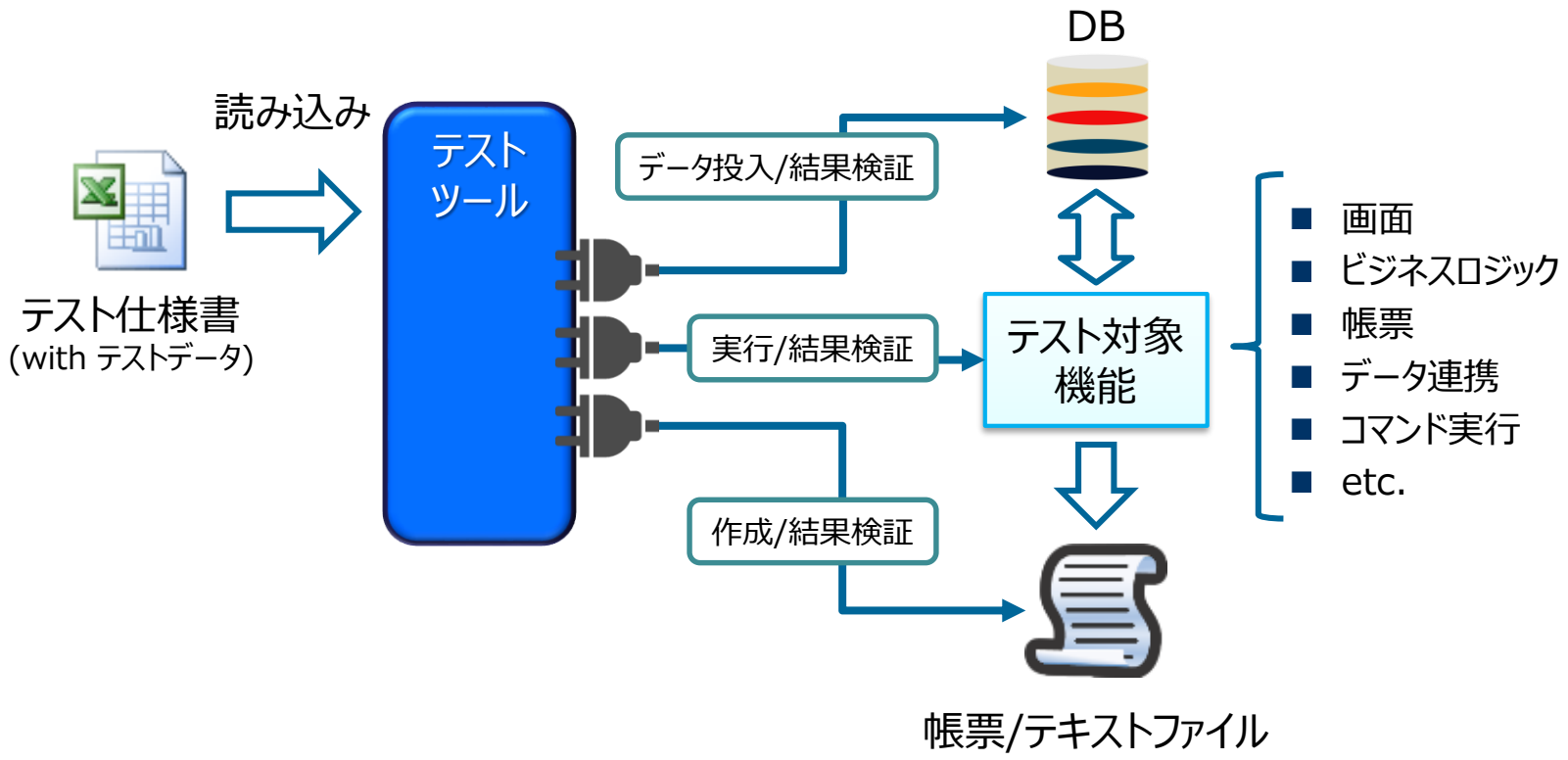
- 作成した自動テストツールは以下のように動作します
  ① テストツールでテスト仕様書を読み込み
  ② プラグイン経由でテスト仕様書に記載された操作を順に実施



Copyright ©2017 by Future Architect, Inc. Japan

E2Eテストツール

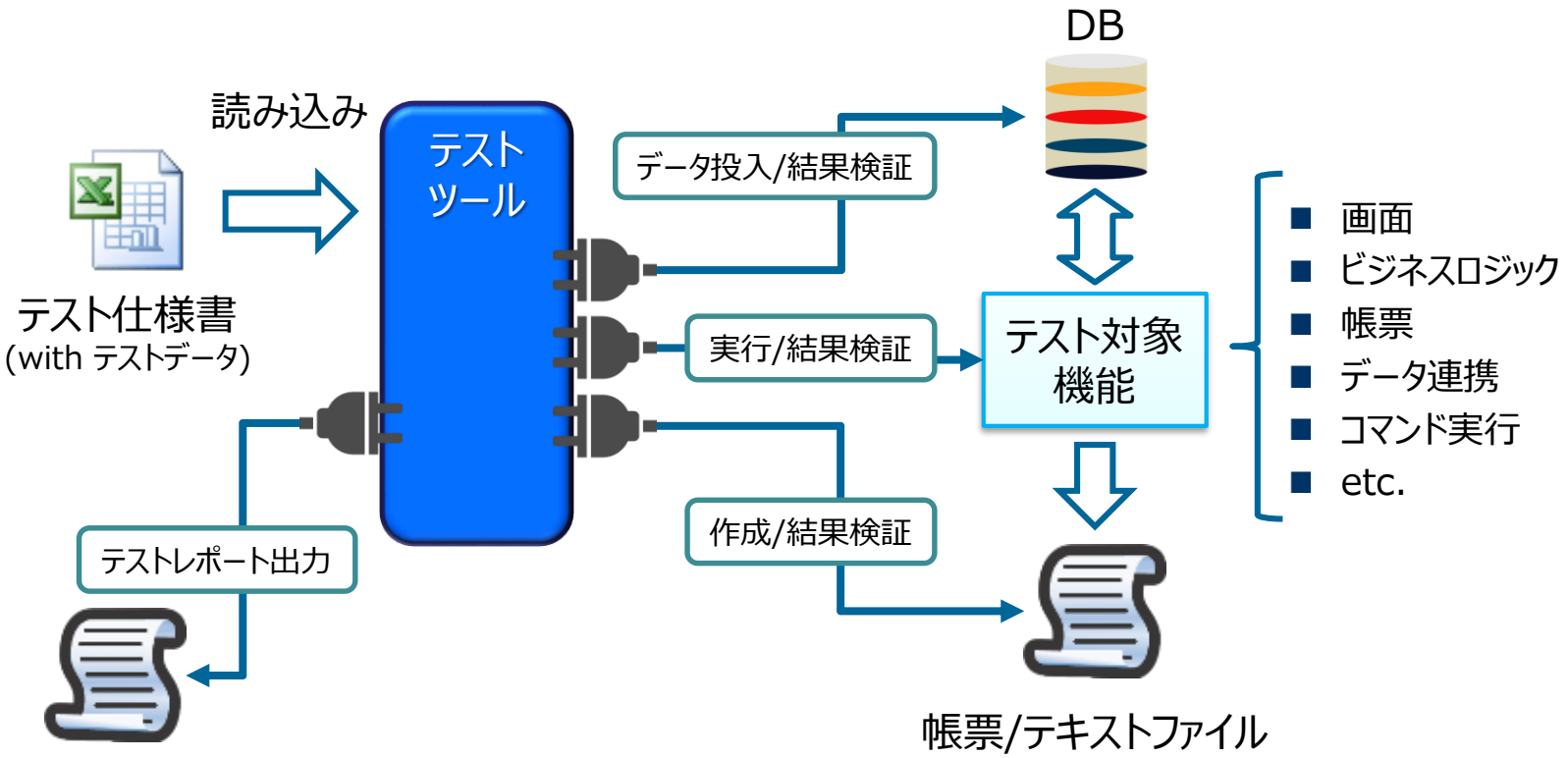
- 作成した自動テストツールは以下のように動作します
  ① テストツールでテスト仕様書を読み込み
  ② プラグイン経由でテスト仕様書に記載された操作を順に実施



Copyright ©2017 by Future Architect, Inc. Japan

E2Eテストツール

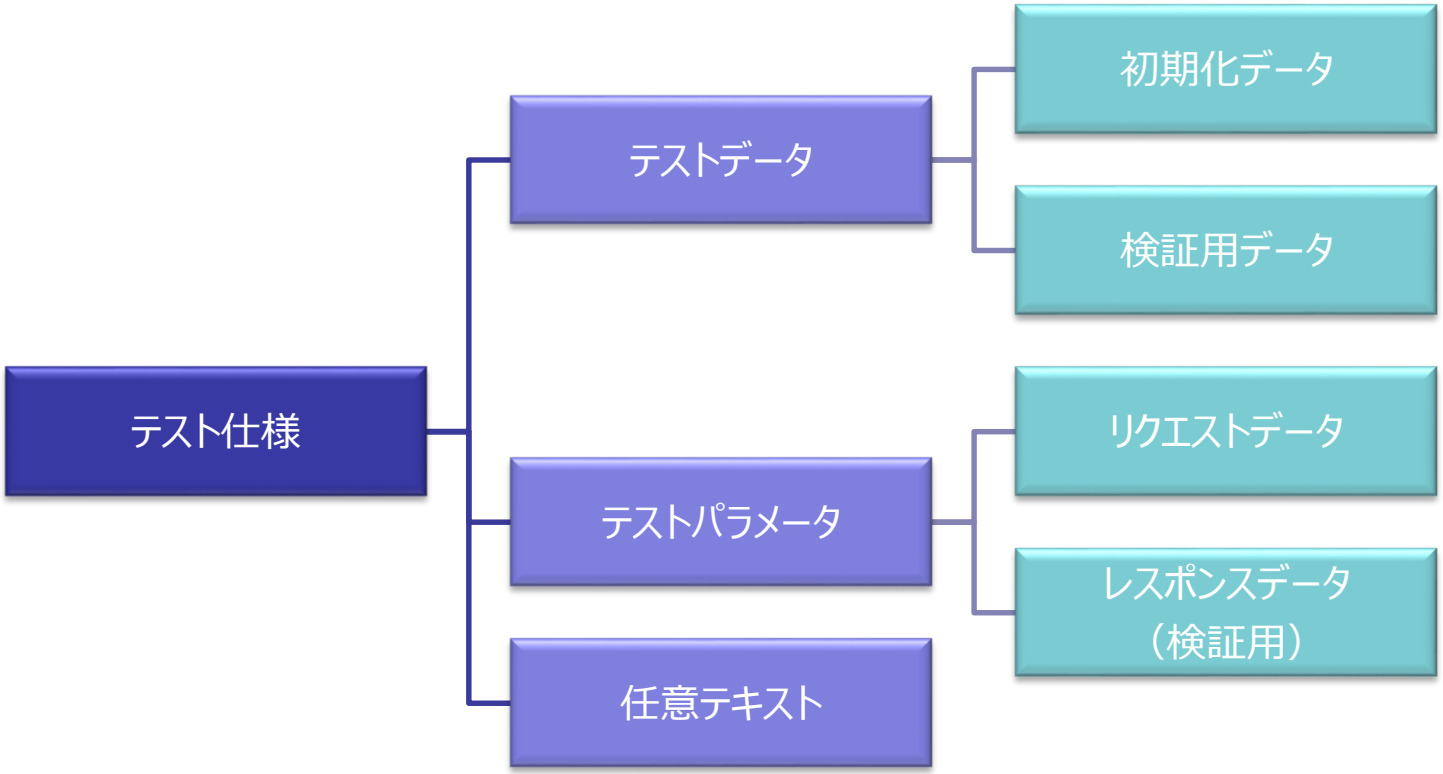
- 作成した自動テストツールは以下のように動作します
  - ① テストツールでテスト仕様書を読み込み
  - ② プラグイン経由でテスト仕様書に記載された操作を順に実施
  - ③ テスト結果をテストレポートとして出力



Copyright ©2017 by Future Architect, Inc. Japan

テスト仕様書

- テストツールのINPUTとなるテスト仕様書は機能単位に１つ作成します
- テスト仕様書は以下の要素で構成されます



Copyright ©2017 by Future Architect, Inc. Japan

テスト仕様書（テスト仕様）

- 一般的なテスト仕様書として利用できるよう、テスト項目や想定結果の列があります
- 各テスト項目に対して、そのテストを実施するための操作手順を**テストコマンド**として記述します

【テスト仕様】
テスト仕様に関する情報を記載する。
テスト項目No、テスト項目、想定結果を記述する

	B	C				K	L	M	N	P	Q	R
1	単体テスト仕様書											
2	システム名					2014/07/10						
3	サブシステム名					Future						
4												
5	テストケース一覧							テスト実施内容詳細				
6	テスト対象仕様							テストスクリプト				
7	項目No	ケースNo	カテゴリ	確認仕様説明	想定結果		※	操作/検証		対象項目	INDEX	パラメータ
8												
9	DB確認	1	正常系	XXXの場合、YYYYになることを確認	XXXの場合、ZZZテーブルの各項目がYYYYになること							
10	DB確認-0001				DBスキーマ内のテーブルに「t1」で始まるシートの初期データを登録する。			操作コマンド	初期データの登録	oracle,db2共通		t1
11	DB確認-0002				DBスキーマ内のテーブルが「v1」で始まるシートに定義された想定データ値と一致することを検証し、エビデンスとしてスナップショットを取得する。			要求コマンド	DBテーブルのスナップショットと内容検査	oracle,db2共通		v1 TEST_CD = '1111'
12	DB確認-0003				DBスキーマ内のテーブルに「i1」で始まるシートの初期データを登録する。		ora	操作コマンド	初期データの登録	oracleのみ		i1
13	DB確認-0004				DBスキーマ内のテーブルが「v2」で始まるシートに定義された想定データ値と一致することを検証し、エビデンスとしてスナップショットを取得する。		ora	要求コマンド	DBテーブルのスナップショットと内容検査	oracleのみ		v2
14	DB確認-0005				DBスキーマ内のテーブルに「i1」で始まるシートの初期データを登録する。		db2	操作コマンド	初期データの登録	db2のみ		i1
15	DB確認-0006				DBスキーマ内のテーブルが「v2」で始まるシートに定義された想定データ値と一致することを検証し、エビデンスとしてスナップショットを取得する。		db2	要求コマンド	DBテーブルのスナップショットと内容検査	db2のみ		v2

記述不要

記述不要

【テスト実施内容詳細】
テスト仕様に対するテストコマンドを定義する。
このテストコマンドがテストツールで実行される。

Copyright ©2017 by Future Architect, Inc. Japan

テスト仕様書（テストデータ）

- テスト実行に必要なDBテーブルデータを定義します
- テスト仕様の**テストコマンド**で定義したデータを指定して操作をおこないます
- テストデータの役割は２つ
  - テスト実行前にDBテーブルに投入するデータの定義
  - テスト実施後のDBテーブルの期待値の定義

	A	B	C	D	E	F	G
1	データ説明	データタイプ	TEST_CD	STR_VAL	NUM_VAL	REGIS_DATE	REGIS_TIMESTAMP
2			1111	1234567890	12.345	2014-10-10	2014-10-10 00:00:00
3			1112	1234567	12.346	2014-10-10	2014-10-10 00:00:00
4			1113	1234567890	12.347	2014-10-10	2014-10-10 00:00:00

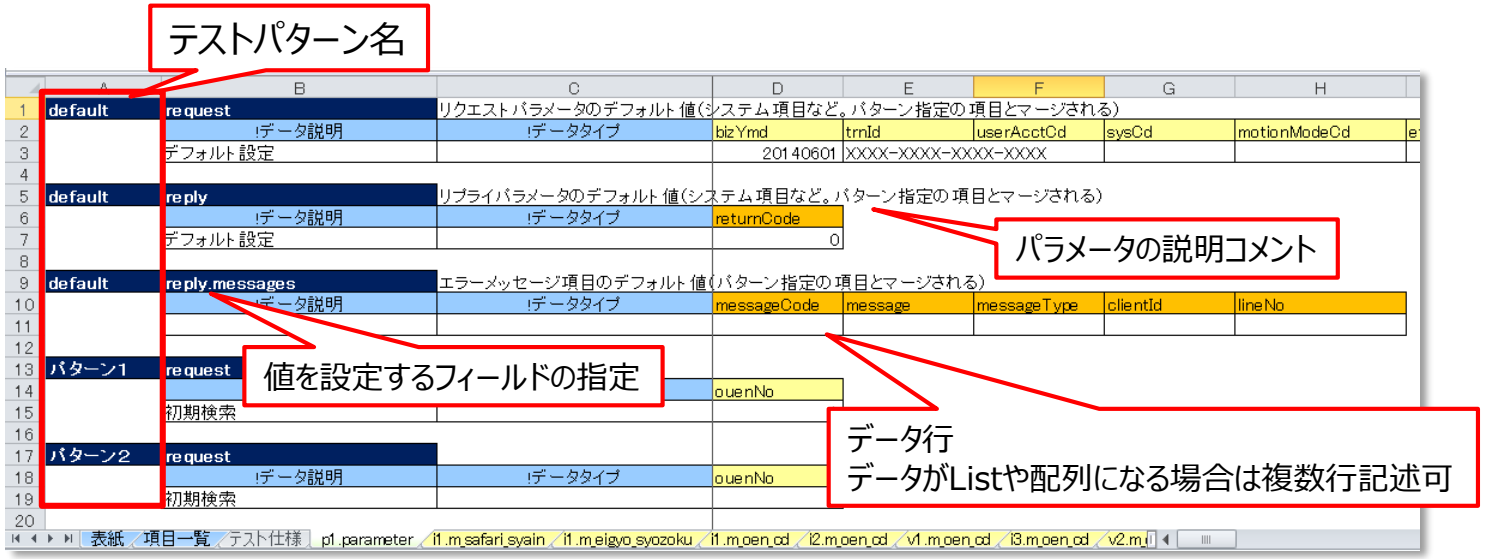
データ行の説明やデータの種類
（正常な行、異常な行など）を説明する
データ投入/データ検証時は無視される

Copyright ©2017 by Future Architect, Inc. Japan

テスト仕様書（テストパラメータ）

- ビジネスロジックやSOAP/REST通信の呼び出しに必要な、リクエストデータの定義と、検証用のレスポンスデータを定義します

- テスト仕様でテストパターン名を指定すると、そのパターン名のデータからリクエストパラメータ/レスポンスパラメータを生成しテストを実行します



Copyright ©2017 by Future Architect, Inc. Japan

テスト仕様書（任意テキスト）

- SQL実行やファイルの比較などで、指定したシートの内容をテキストファイルとして出力し利用する場合に定義します

	A	B	C	D	E	F	G	H
1	delete from m_sris_cd_hnkn_ks;							
2	delete from m_krgm_flg_hnkn_ks;							
3	delete from m_sy_inf_hnkn_ks;							
4	delete from m_bzi_inf_hnkn_ks;							
5	delete from m_nib_cd_hnkn_ks;							
6	delete from m_syhn_cd_knrn_kmk_hnkn_ks;							
7	delete from m_suti_kg_hnkn_ks;							
8	delete from t_mit_inf_1140;							
9	delete from t_mit_sy_inf_1140;							
10	delete from t_mit_misi_1140;							
11	delete from t_mit_bzi_inf_1140;							
12	delete from t_bzi_csv_1140;							
13	delete from w_hrds_grp;							
14	delete from w_hrds_grp_dtl;							
15	delete from w_hrds;							
16	delete from w_hrds_sy_cndtn;							

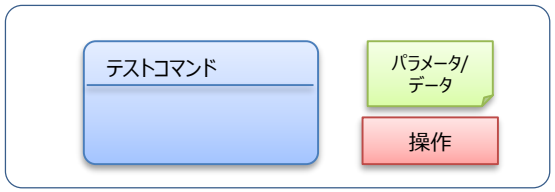
表紙 テスト仕様 DELETE_ALL.sql p1.parameter

Copyright ©2017 by Future Architect, Inc. Japan

テスト実行の流れ

- 下記のテスト仕様書がある場合、テストは次の流れで実行されます

テスト実施内容詳細					
テストスクリプト					
※	操作/検証		対象項目	INDEX	パラメータ
	カテゴリ	コマンド			
	操作コマンド	初期データの登録			i1
	操作コマンド	ビジネスロジックの実行			p1 パターン1 com Online A0102B1101 Execute
	要求コマンド	DBテーブルのスナップショットと内容検査			v1

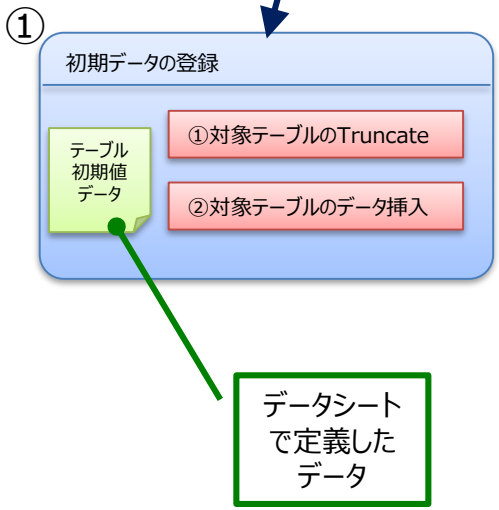


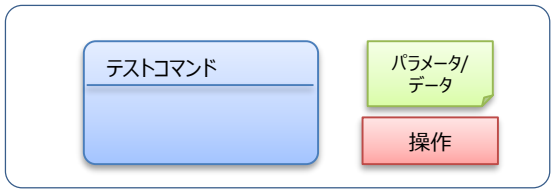
Copyright ©2017 by Future Architect, Inc. Japan

テスト実行の流れ

- 下記のテスト仕様書がある場合、テストは次の流れで実行されます

テスト実施内容詳細					
テストスクリプト					
※	操作/検証 カテゴリ	操作/検証 コマンド	対象項目	INDEX	パラメータ
①	操作コマンド	初期データの登録			i1
	操作コマンド	ビジネスロジックの実行			p1 パターン1 com Online A0102B1101 Execute
	要求コマンド	DBテーブルのスナップショットと内容検査			v1



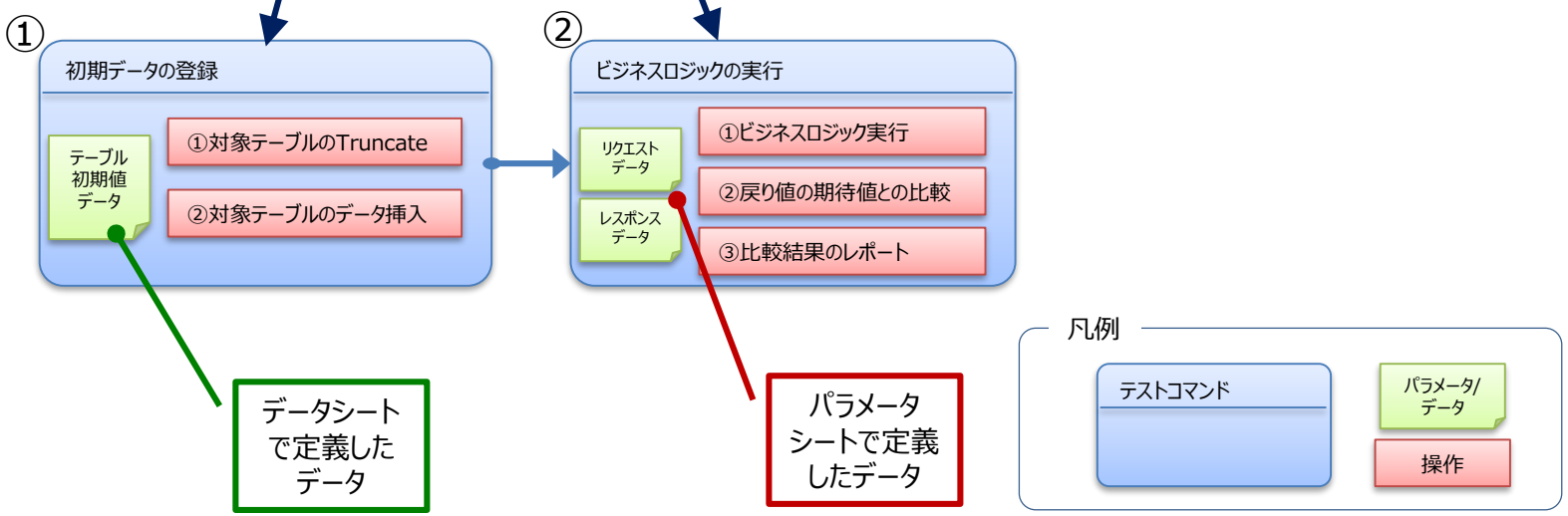


Copyright ©2017 by Future Architect, Inc. Japan

テスト実行の流れ

- 下記のテスト仕様書がある場合、テストは次の流れで実行されます

	テスト実施内容詳細				
	テストスクリプト				
	※	操作/検証 カテゴリ	操作/検証 コマンド	対象項目	INDEX	パラメータ
①		操作コマンド	初期データの登録			i1
②		操作コマンド	ビジネスロジックの実行			p1 パターン1 com Online A0102B1101 Execute
		要求コマンド	DBテーブルのスナップショットと内容検査			v1



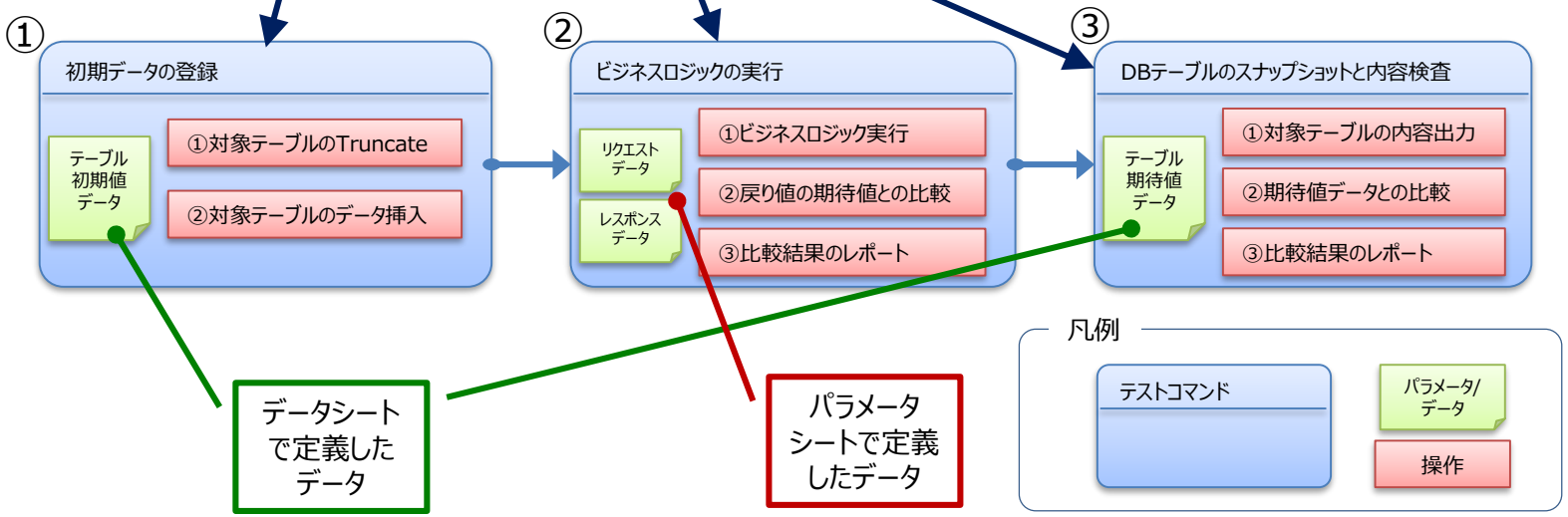
Copyright ©2017 by Future Architect, Inc. Japan

テスト実行の流れ

- 下記のテスト仕様書がある場合、テストは次の流れで実行されます

テスト実施内容詳細
テストスクリプト

	※	操作/検証 カテゴリ	操作/検証 コマンド	対象項目	INDEX	パラメータ
①		操作コマンド	初期データの登録			i1
②		操作コマンド	ビジネスロジックの実行			p1 パターン1 com Online A0102B1101 Execute
③		要求コマンド	DBテーブルのスナップショットと内容検査			v1



Copyright ©2017 by Future Architect, Inc. Japan

対策に対する評価

Copyright ©2017 by Future Architect, Inc. Japan

単体テストの問題は解決できたのか？

1．テスト方法が統一されていない

- 複数プロジェクトで今回作成したE2Eテストツールを使用することにより、テストの実施方法や対象範囲が統一された

2．テスト仕様を満たしているか判断しづらい

- テスト仕様は全てテスト仕様書にまとまっており、常に最新の状態であるため、テスト仕様書のレビューを行うことでテスト仕様を満たしていることを確認できるようになった

3．保守・メンテナンスコストの高さ

- テスト仕様書がテストスクリプトを兼ねることで、記述方法が統一され保守性が向上した
- 機能単位のテストにしたことでUnit単位のテストより記述量が大幅に削減され、それに伴いメンテナンスコストも減少した

Copyright ©2017 by Future Architect, Inc. Japan

機能単位テストで品質は低下しないのか？

- 定量的な評価はできていないが、機能単位テストに変更したことによる大幅なテスト品質の低下は見られませんでした
- Unitテストの場合はコードカバレッジの担保を重要な指標にしますが、コードカバレッジが高いからと言って仕様を満たしていることにはなりません
- 機能テストの場合は機能に対するテスト仕様のパターン網羅を重要な指標とすることにより、効果の高いテストが実施できることが分かりました

  - **機能単位テストの優位な点**
    - ✓ 複数のモジュールを繋げて動作させることになり、モジュール間インタフェースの確認も合わせて行うことができる
    - ✓ グレーボックステストとしてINPUT（画面操作やリクエストパラメータなど）とOUTPUT（DBテーブルの変更やファイル出力）の検証を行えばよい為、テスト初心者にもわかりやすい
    - ✓ 実際のアプリケーションを動作させるため、OSやミドルウェア固有の問題も見つけやすい

  - **機能単位テストの課題**
    - ✓ 巨大な機能を作った場合、テストケースが複雑で膨大になる
    - ✓ Unit単位のテストに比べ、テスト実施に時間がかかりやすい
      - ➢ アプリケーションの起動や停止、DBの初期化などを行うため

Copyright ©2017 by Future Architect, Inc. Japan

弊社におけるテストツール導入プロジェクト実績

- 弊社では多種多様な業界の基幹システム構築でこのE2Eテストツールが利用されています

業界	導入年度	テスト対象
製造業	2007年	・ビジネスロジック　・帳票　・データ連携
小売業	2009年	・ビジネスロジック　・帳票
流通業	2012年	・画面　・ビジネスロジック　・データ連携　・帳票
EC	2013年	・ビジネスロジック　・データ連携
地銀	2014年	・ETL定義
住宅設備機器	2015年	・ビジネスロジック　・帳票　・データ連携
保険業	2015年	・画面
証券	2015年	・REST-API
エネルギー	2016年	・画面　・ビジネスロジック　・帳票

Copyright ©2017 by Future Architect, Inc. Japan

プロジェクト別テストケース数の比較

- テストツールを統一することにより、同じ尺度でテスト品質を図ることができるようになりました

  - プロジェクト別機能種類別の規模とテストケース数の比較

プロジェクト		画面	ビジネスロジック	データ連携	帳票出力	REST-API	合計	１機能あたりテストケース数
流通業	機能数	311	368	267	162	0	1,108	16.8
	テストケース数	8,051	7,621	2,242	750	0	18,664	
エネルギー	機能数	144	54	35	119	0	352	15.5
	テストケース数	4,587	540	0	348	0	5,475	
EC	機能数	6	147	170	0	0	323	4.0
	テストケース数	0	514	786	0	0	1,300	
証券	機能数	15	25	0	9	73	122	11.5
	テストケース数	0	0	0	0	1,406	1,406	

Copyright ©2017 by Future Architect, Inc. Japan

プロジェクト別テストケース数の比較

- テストツールを統一することにより、同じ尺度でテスト品質を図ることができるようになりました

  - プロジェクト別機能種類別の規模とテストケース数の比較

プロジェクト		画面	ビジネスロジック	データ連携	帳票出力	REST-API	合計	1機能あたりテストケース数
流通業	機能数	311	368	267	162	0	1,108	16.8
	テストケース数	8,051	7,621	2,242	750	0	18,664	
エネルギー	機能数	144	54	35	119	0	352	15.5
	テストケース数	4,587	540	0	[illegible]	[illegible]	75	
EC	機能数	6	147	170	0	0	323	4.0
	テストケース数	0	514	786	0	0	1,300	
証券	機能数	15	25	0	9	73	122	11.5
	テストケース数	0	0	0	0	1,406	1,406	

極端にテストケース数が少ない！

Copyright ©2017 by Future Architect, Inc. Japan

今後の展望

Copyright ©2017 by Future Architect, Inc. Japan

今後の展望

今回紹介したE2Eテストツールを利用するプロジェクトが増えていくと、同じ尺度でテスト進捗や機能毎のテスト密度といったメトリックスを収集可能になります

それにより、より正確な品質指標やテスト工数見積もりの算出が可能になることが期待されます

社内に閉じず、多くのプロジェクトでこのテストツールを利用してもらえるように、現在このツールのオープンソース化を計画しています

ご意見・ご要望をお待ちしています

Copyright ©2017 by Future Architect, Inc. Japan

FUTURe Architect