SPI Japan 2014 in 沼津

テストデータ自動生成による品質･コストの改善

～ テスト設計システム構築とアイデアを出す工夫 ～

住友電工情報システム株式会社
QCD改善推進部
品質改善推進グループ
服　部　悦　子

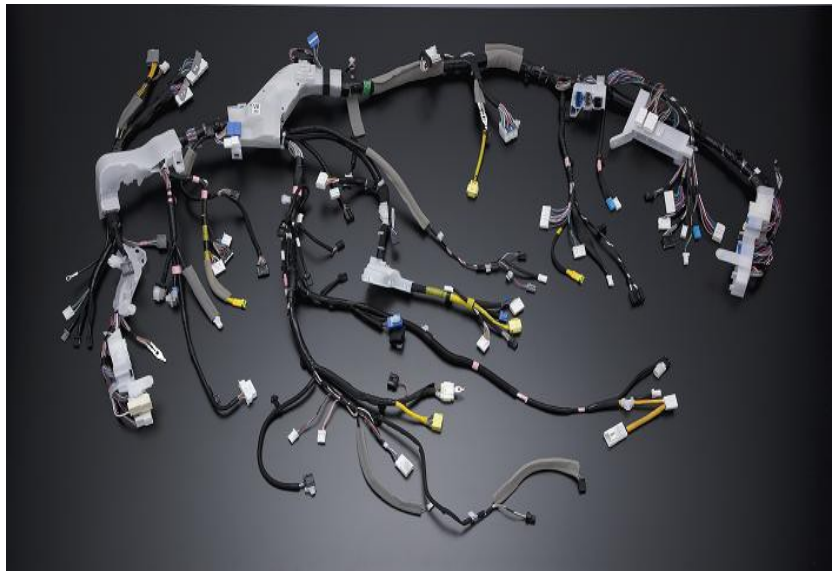
2014.10.16

住友電工情報システム株式会社　概要

■設　立：　1998年10月1日
■資本金：　4．8億円
　　住友電気工業株式会社：　60%
　　住友電装株式会社　　：　40%
■従業員：　450名
■代表取締役社長：　白井　清志
■事業内容：
　□パッケージソフトウェア（楽々シリーズ）の開発・販売
　□情報処理システムの開発受託
　□コンピュータ運用業務の受託
　□情報機器の販売
■URL：　http://www.sei-info.co.jp/


本社（SORA新大阪21）

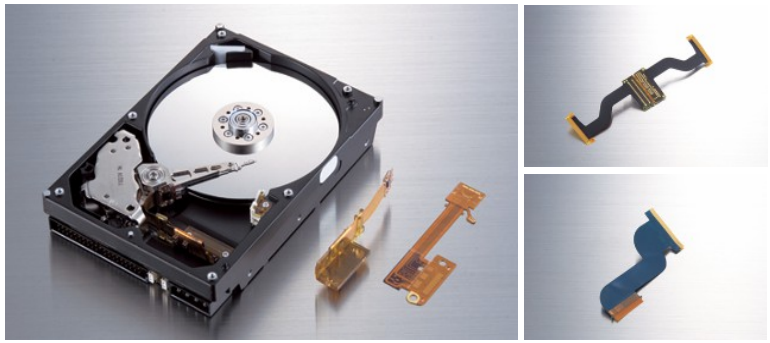
住友電気工業(親会社)の製品



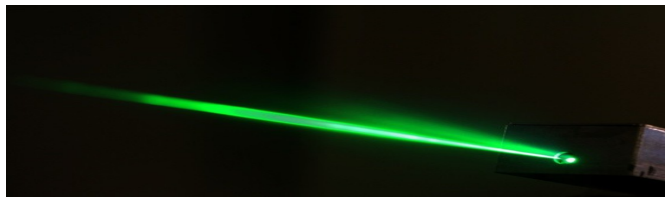
ワイヤーハーネス

超硬工具　イゲタロイ®

合成ダイヤモンド単結晶　スミクリスタル®

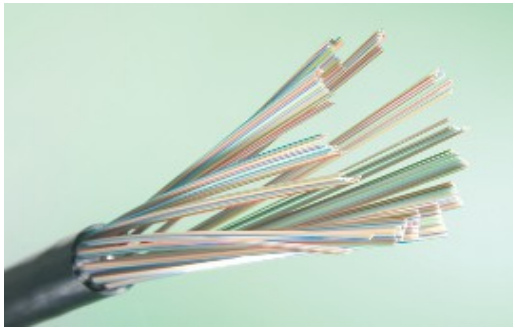


フレキシブルプリント回路

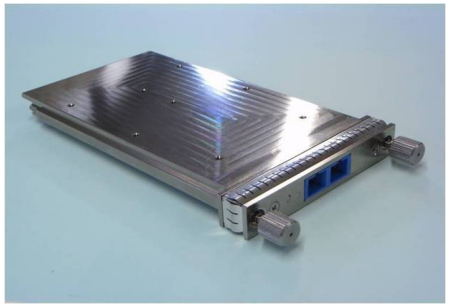


純緑色半導体レーザ

銅荒引線



多心光ファイバケーブル



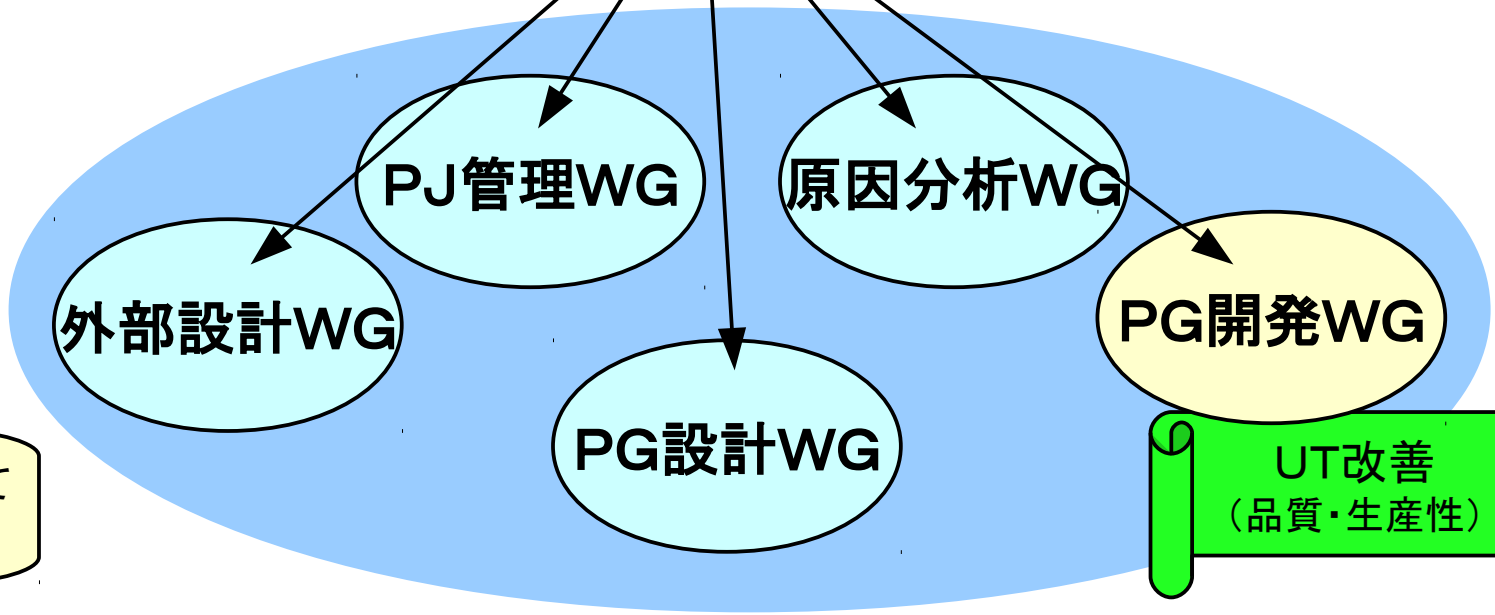
40Gbit/s伝送用光トランシーバ

改善活動の状況

目標達成の為に
ワーキンググループの新設
既存ワーキンググループへの役割割り当て

組織の
年度目標
（品質・コスト）

SWAT（部長、G長、SE）



私は改善推進担当として
複数のWGに参加

目次

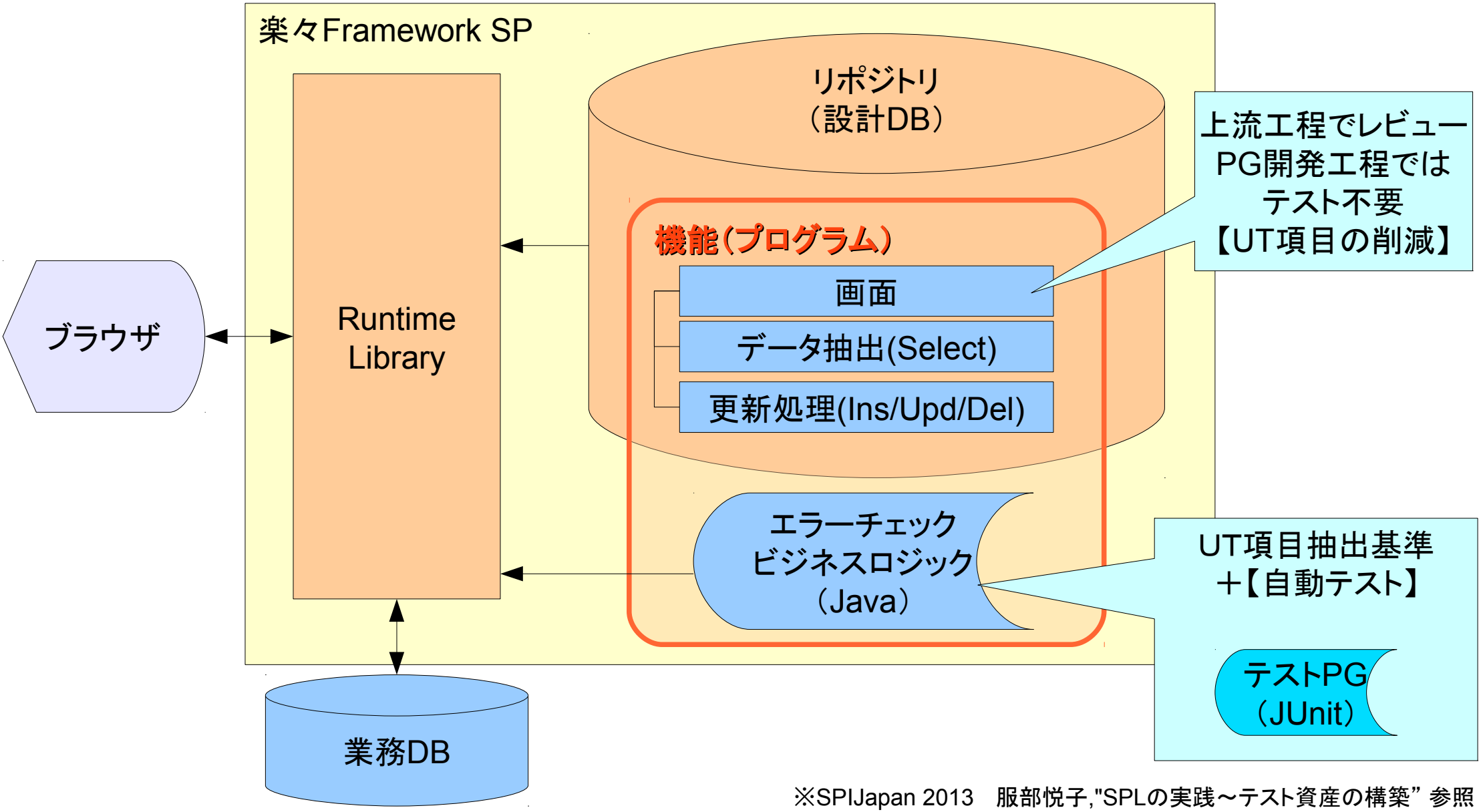
1. これまでの取り組み
　　自動テストの実現と課題

2. テストデータ自動生成の実現に向けて
　　ワーキンググループでアイデアを出す工夫

3. テストデータ自動生成の実現
　　テスト設計システムの構築

目次

これまでの取り組み　～自動テストの実現～



※SPIJapan 2013　服部悦子,"SPLの実践～テスト資産の構築”参照

自動テスト実施プロジェクトの結果

※組織基準を100とした相対値

	組織基準値	自動テスト実施PJ
PG開発　生産性	100	98.9
PG開発　品質　作込欠陥(*)	100	47

半減

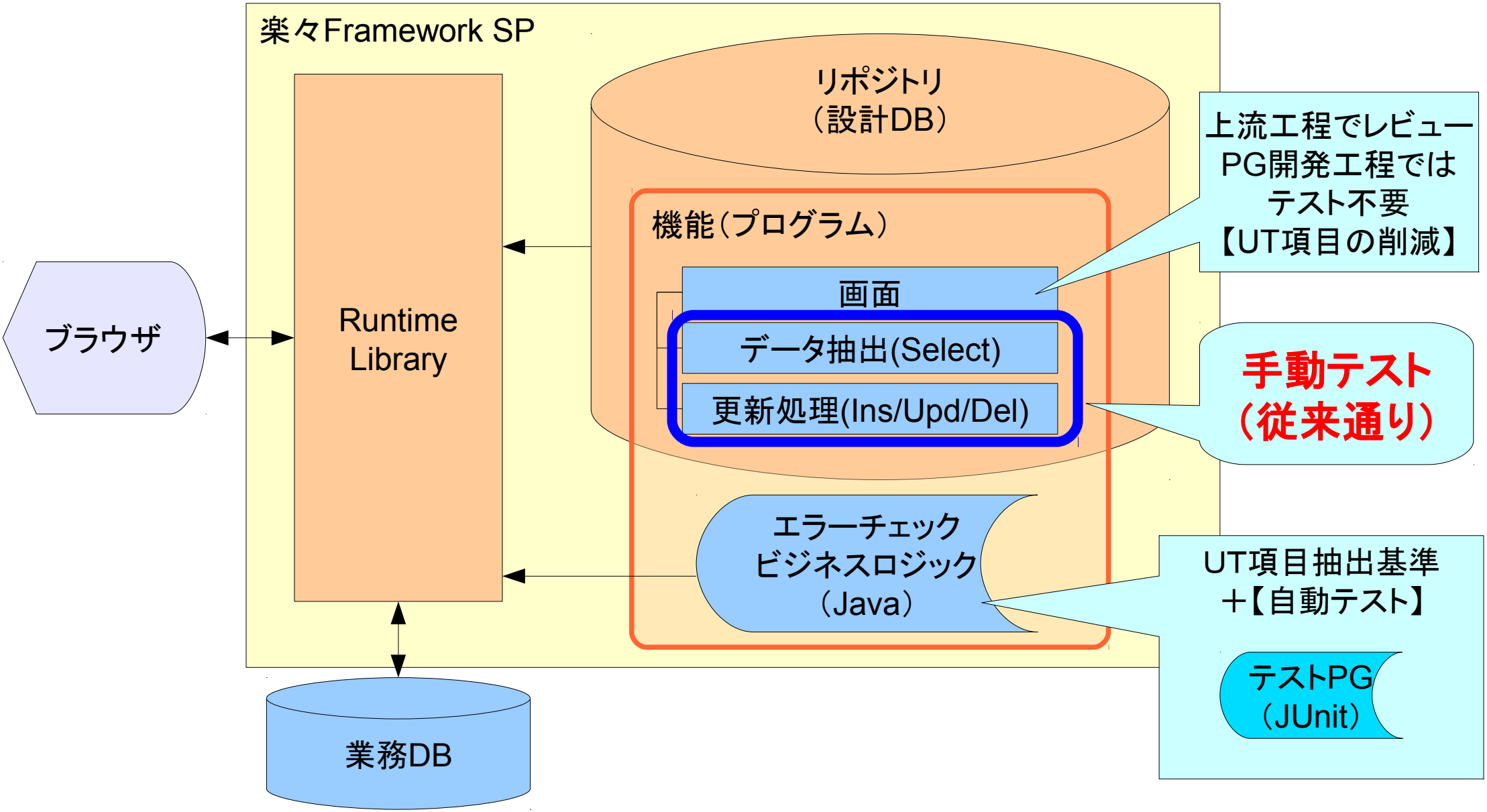
(*)作込欠陥＝CI＋UT検収 検出

(内 欠陥区分別内訳 TOP3)		
ロジックの実装ミス	23	2
実装漏れ	19	3
DB処理(データ抽出、更新処理)	10	9
:	:	:

自動テスト効果大

DB処理の欠陥がTOPになった
⇒次はここの対策

自動テスト 実現箇所 と できていない箇所



DB処理の自動テストに向けて・・・

欠陥を抑える対策

DB処理のUT項目抽出基準 作成

例. WHEREの条件網羅（Javaコーディングの if文 や case文と同じ）、境界値

テスト実行前に必要なデータを登録できる

Javaロジックの自動テストではテストデータ（画面入力値）はコーディング
Selectするものはテスト実行前にDBへ登録しておく必要がある

繰り返しテストをできるようにする

テスト（プログラムの実行）により更新され値が変わってしまう＝繰り返しテストできない

課題. テストデータ作成工数の削減

単体テストに必要なデータ(従来の工夫)

- 本番データをマスキングして開発環境にコピー
- プログラム開発順の工夫　先行機能→後続機能

本番データは常に
あるわけではない

部分的に請負に
出す場合できない

従来からの課題

なぜテストデータ作成は時間がかかるのか？

原 因

受注 → 製造指示 → 出来高 → 出荷

テストに必要なレコード(件数)の多さ

後続機能の場合、**先行機能から発生するデータを順に作成**する必要がある
自分が担当していてない機能の仕様を理解してデータを作成する必要がある

テーブル項目の多さ

仕様によってテスト値が決まる項目 と どうでもいい項目
どうでもいい項目 の場合でも **型・桁の定義（制約）に則った値を決める必要がある**
※上述のレコード数が多い場合、さらに時間がかかる

新たな対策

対策案「テストデータ自動生成システム」

DB処理の自動テスト実現　欠陥を抑える対策

DB処理のUT項目抽出基準 作成

テスト実行前に必要なデータを登録できる

繰り返しテストをできるようにする

システムとして
管理されていれば・・・

・テストに必要なデータが分かる
・何度でもテスト前の状態に戻せる

課題．テストデータ作成に　時間がかかる原因

テストに必要なレコード（件数）の多さ

テーブル項目の多さ

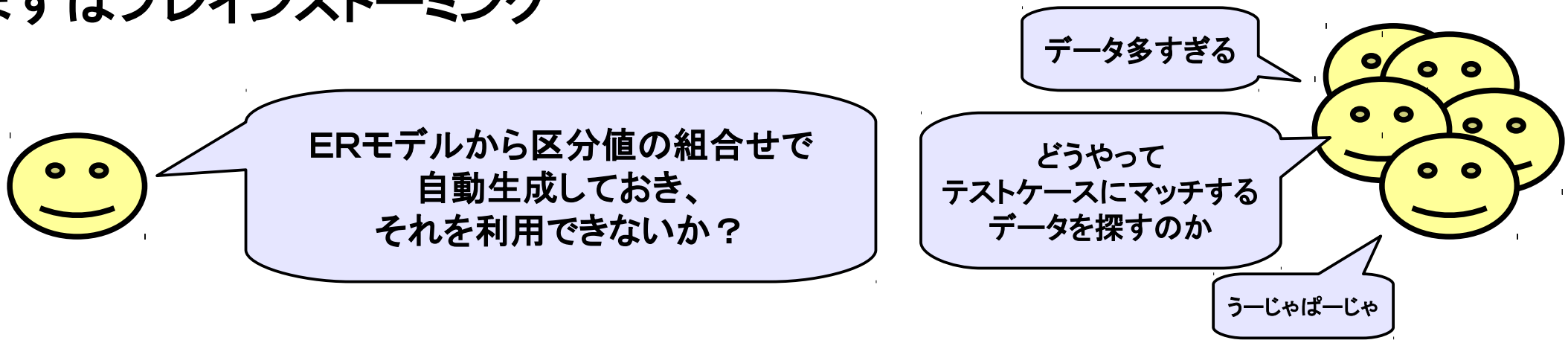
自動生成できれば・・・

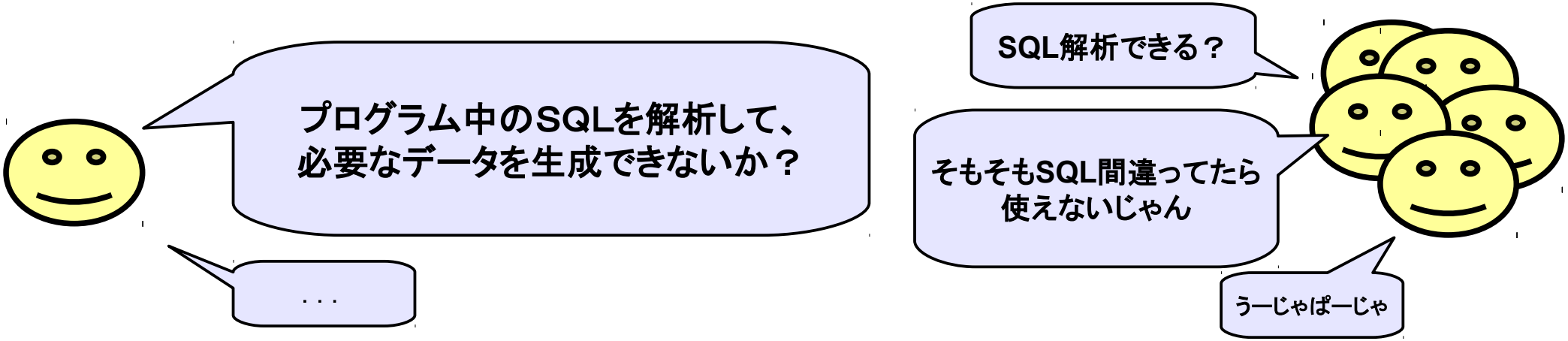
・最小限の手間で必要なテストデータを準備できる

目次

テストデータ自動生成をどうやって実現するか？

まずはブレインストーミング

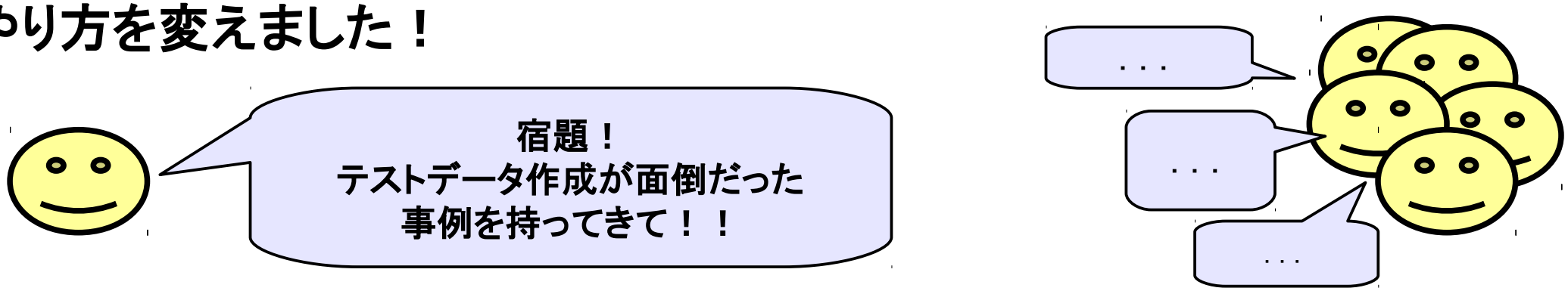




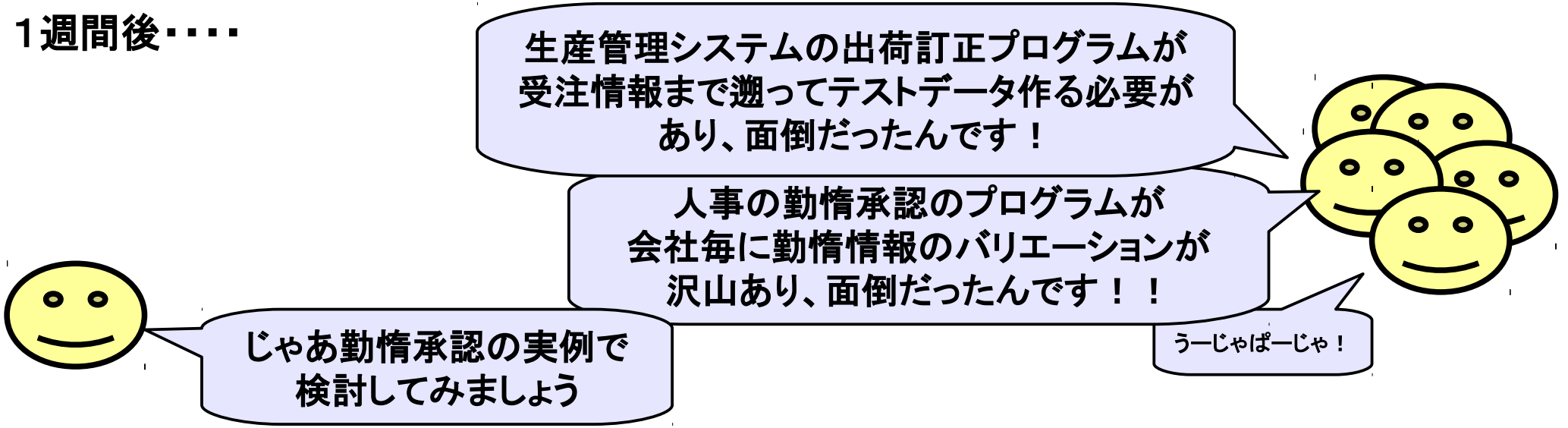
・・・使えるアイデアが生まれない

テストデータ自動生成をどうやって実現するか？

やり方を変えました！



1週間後・・・・



この後、議論が活発になり使えるアイデアが出だしました。
何故だろう・・・？

【工夫①】 実物で検討する

テストデータを自動生成するためのインプットを実際のプログラムで検討

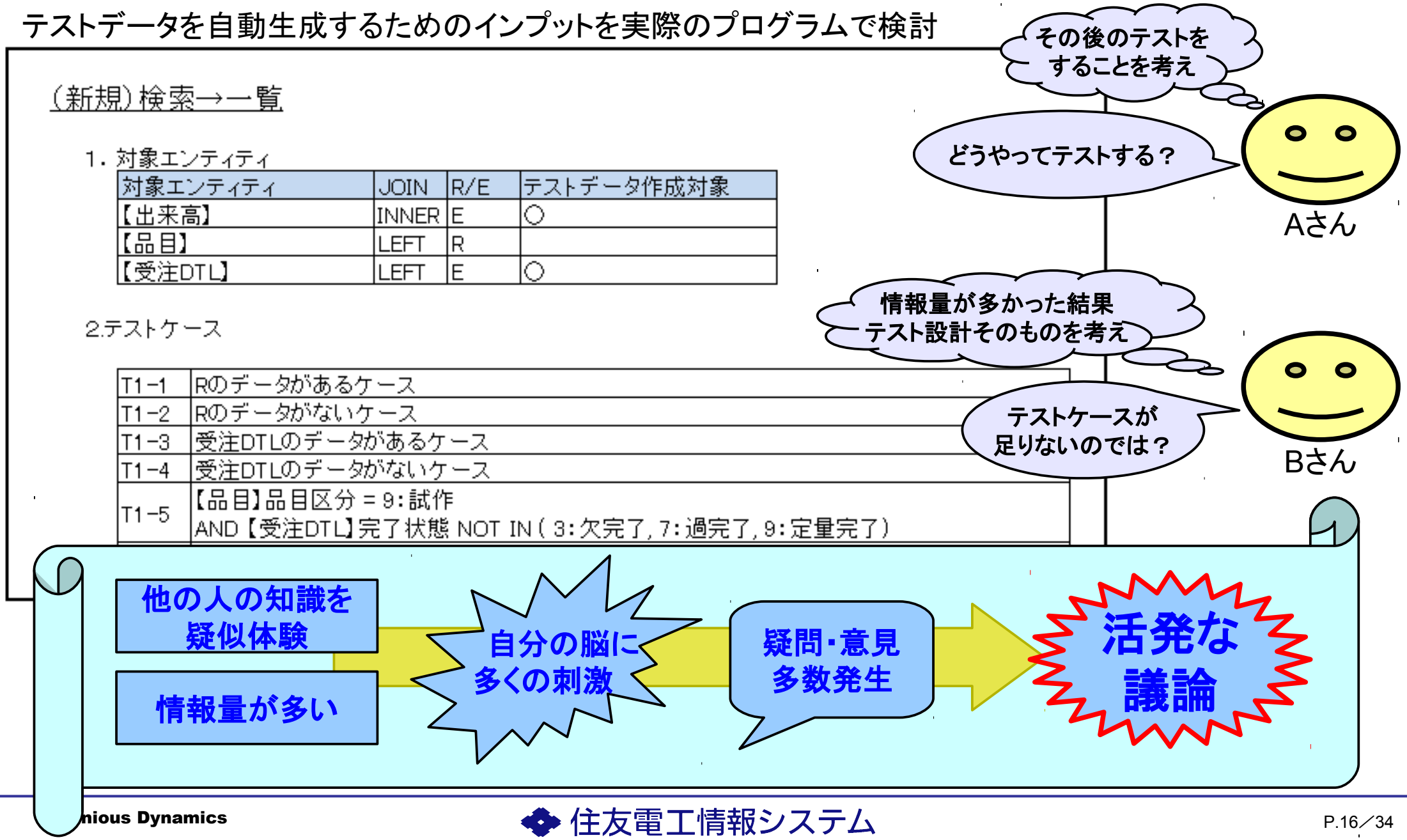
（新規）検索→一覧

1．対象エンティティ

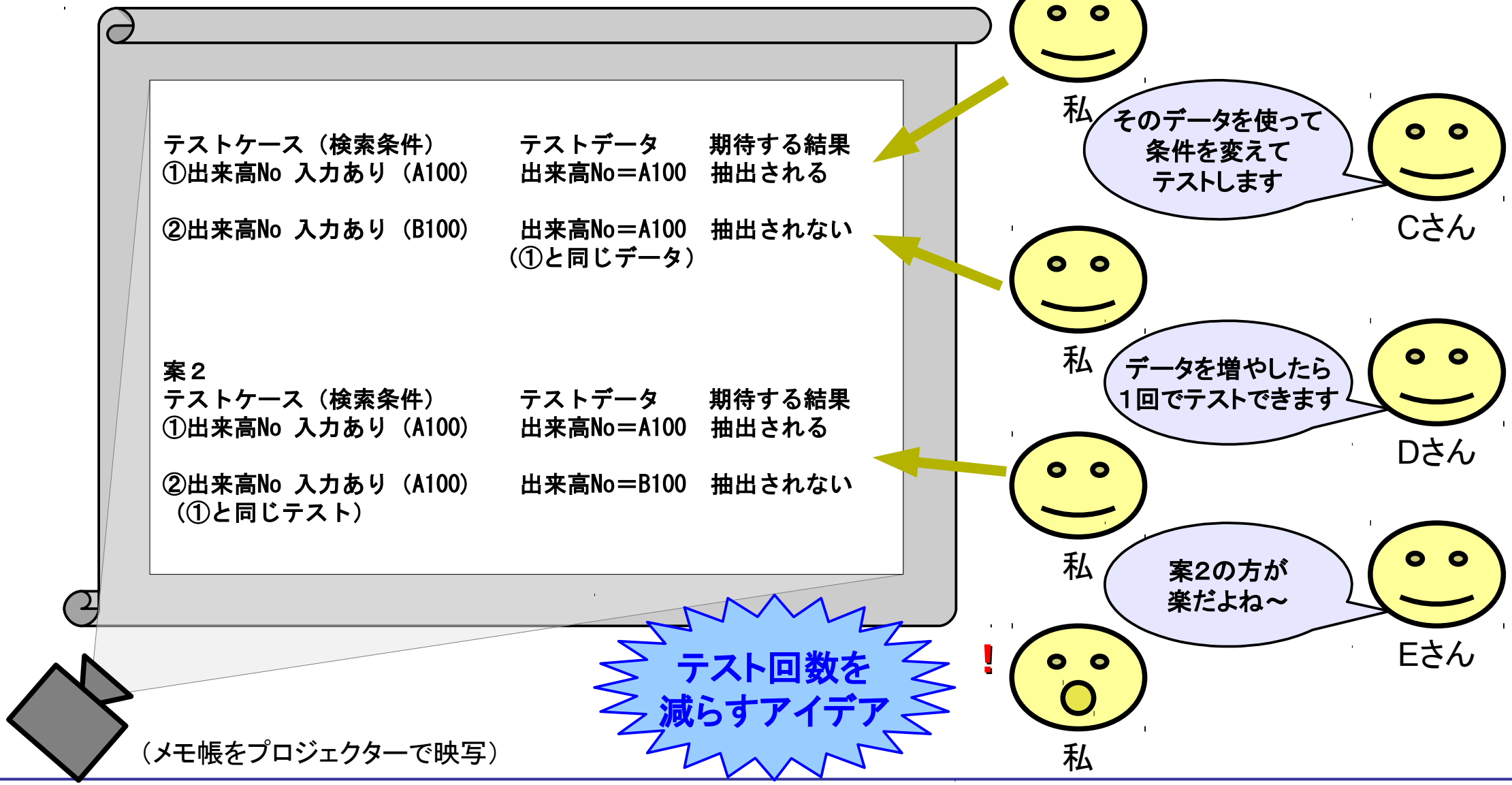
対象エンティティ	JOIN	R/E	テストデータ作成対象
【出来高】	INNER	E	○
【品目】	LEFT	R	
【受注DTL】	LEFT	E	○

2.テストケース

T1-1	Rのデータがあるケース
T1-2	Rのデータがないケース
T1-3	受注DTLのデータがあるケース
T1-4	受注DTLのデータがないケース
T1-5	【品目】品目区分 = 9:試作 AND【受注DTL】完了状態 NOT IN（3:欠完了, 7:過完了, 9:定量完了）

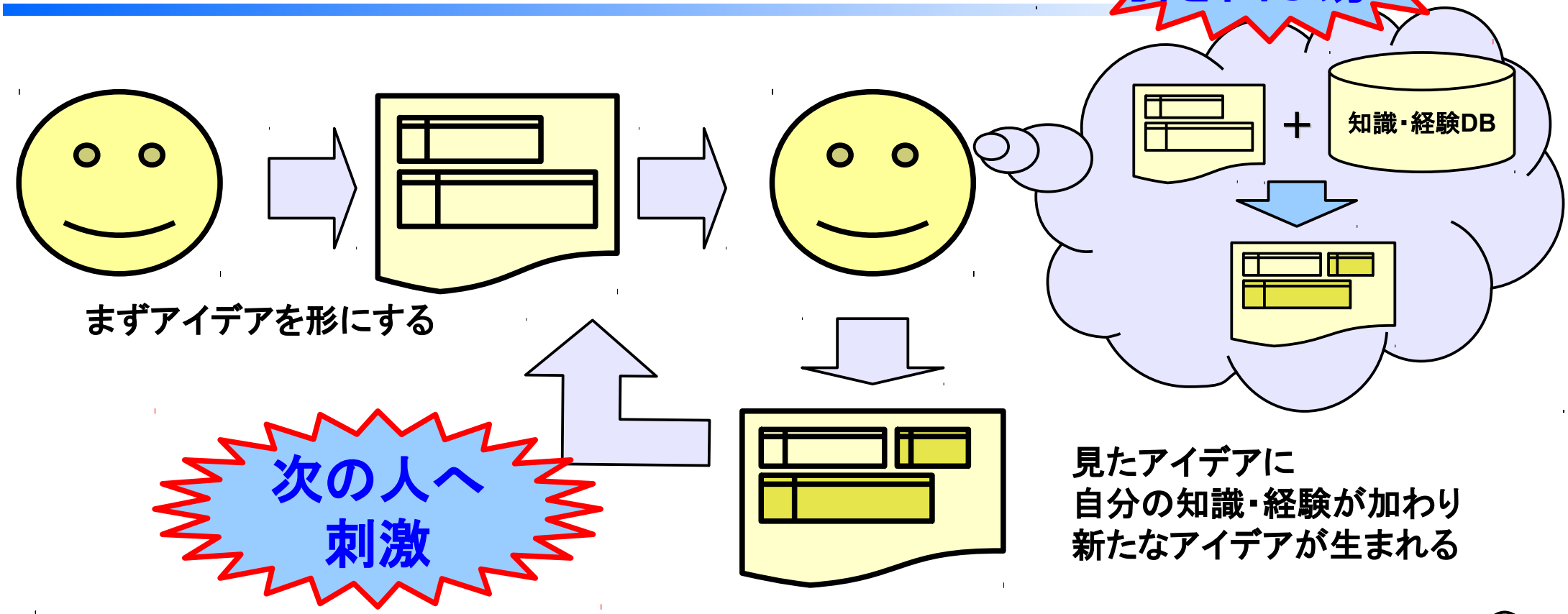


【工夫②】 議論を可視化する



「実物で検討」＋「議論の可視化」

知識・経験が
引き出し易い



可視化していないと・・・

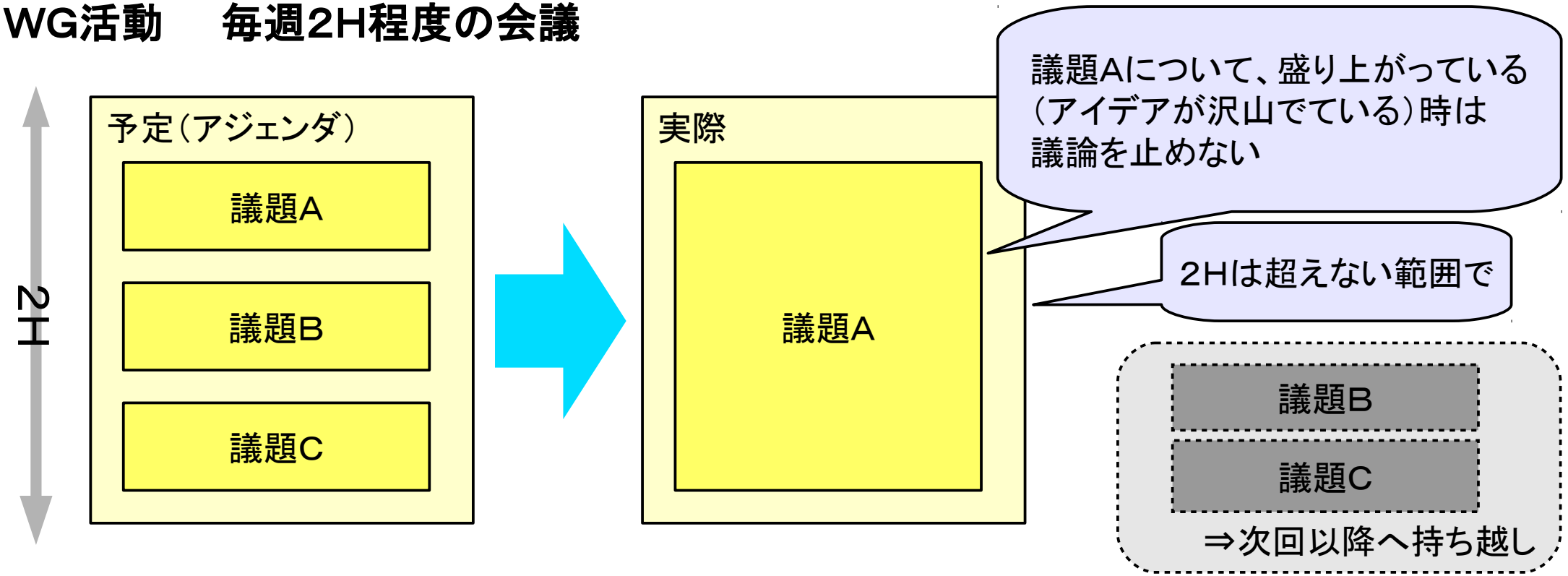
自分が考えている最中に議論が進んでしまうと**追いつけない**

最初に議論した内容が残っていないと **同じ議論** を **繰り返してしまう**

漏れ・ムダ

【工夫③】 効果的に時間を使う

WG活動　毎週2H程度の会議



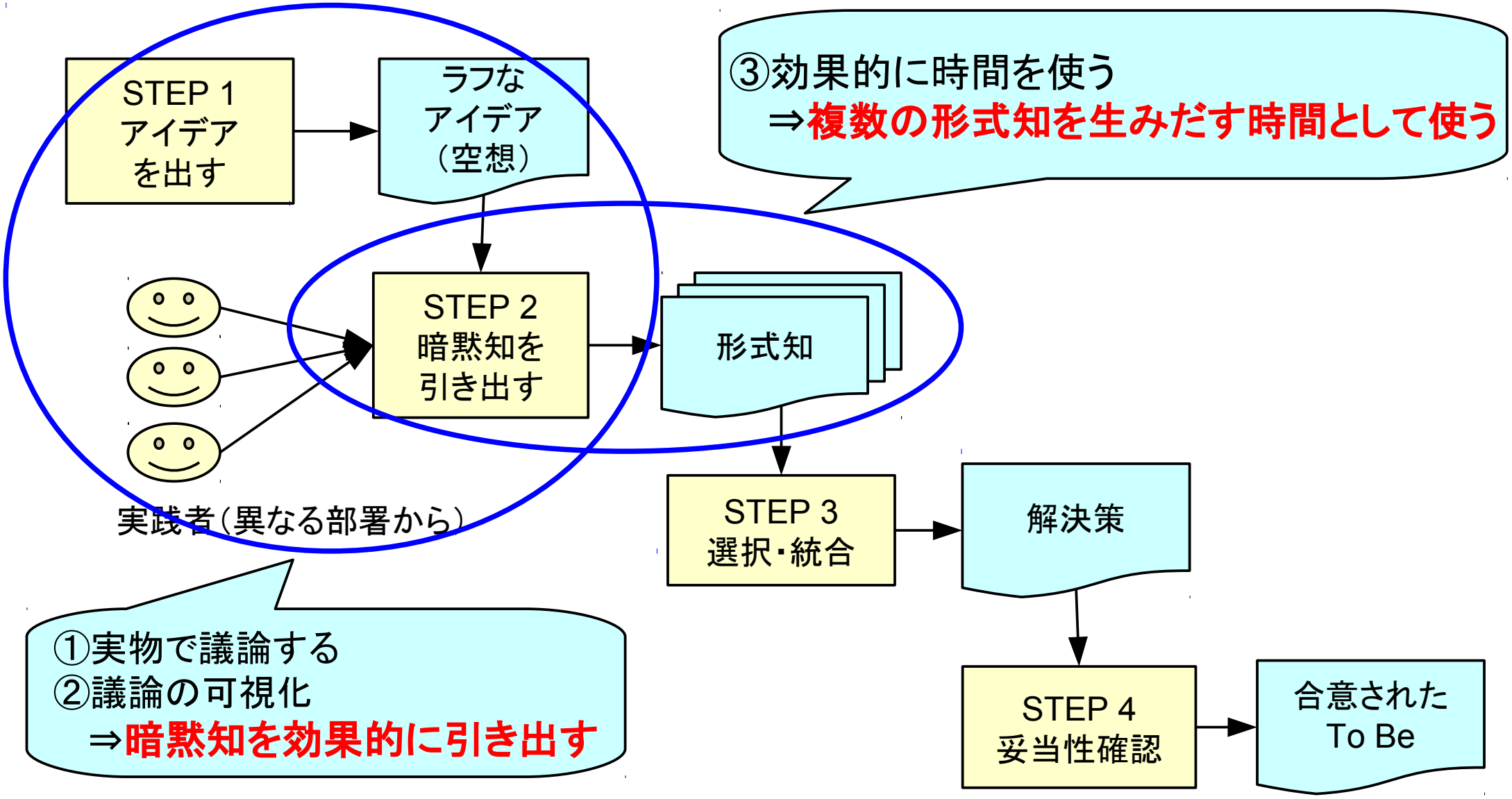
複数のアイデアが生まれる

↳ **よりよい案が採用される**

議論中は時間が超過しても止めない

To Be 創造プロセス

※SPIJapan 2014　中村伸裕,"SECIモデルによる改善活動基盤の評価”より

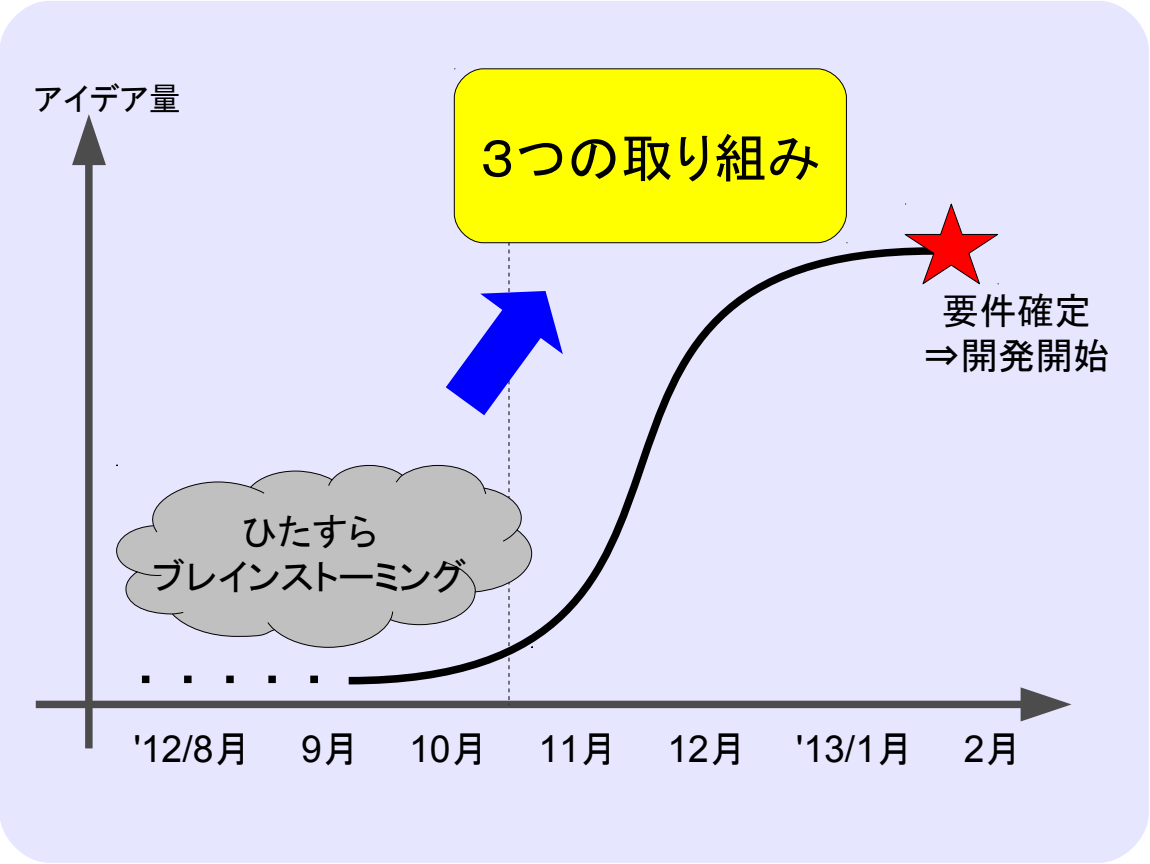


ワーキンググループでアイデアを出す工夫

① 実物で議論

② 議論の可視化

③ 効果的に時間を使う

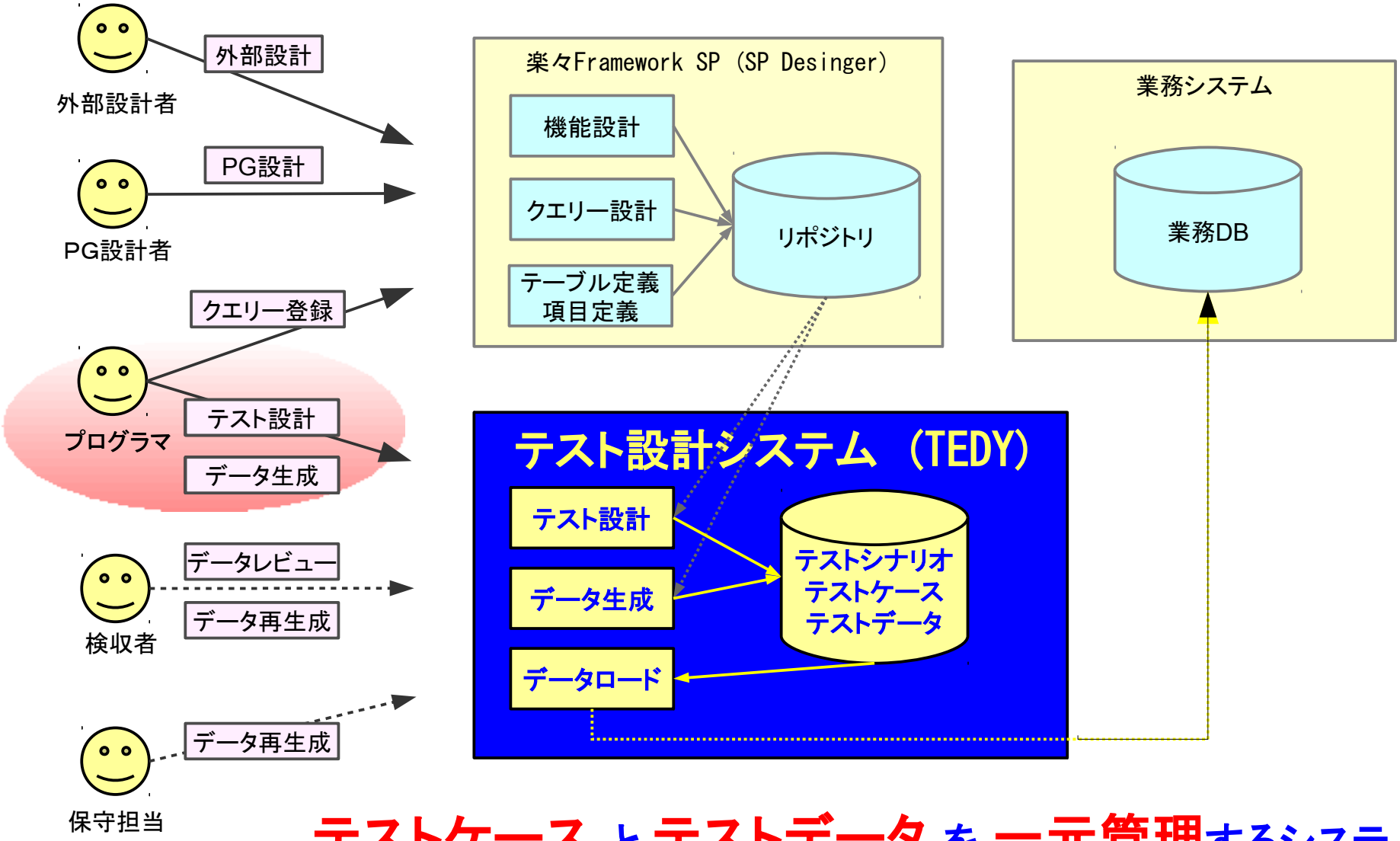


改善の
アイデア

この3つの取り組みがアイデアを出すには非常に効果的

目次

テスト設計システム



テストケース と テストデータ を 一元管理 するシステムとして構築

テストデータ自動生成の仕組み　何を入力すればいいのか？

テストケース毎の
・必要TBLと必要件数
・テスト対象の区分値（任意）
・各項目の値指定（任意）

Input

テスト設計システム

Output

テストデータ

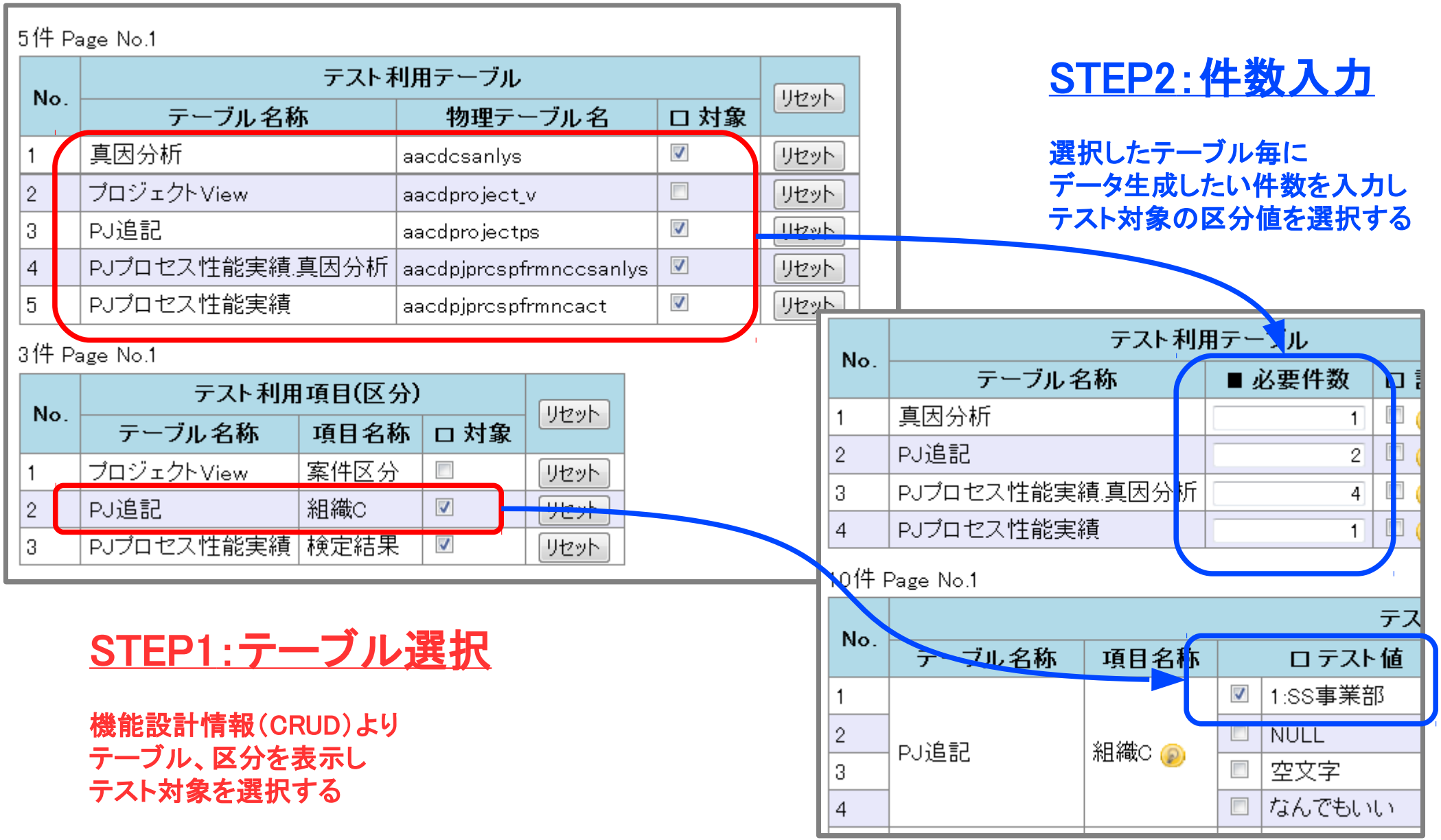
	ケース1	ケース2	ケース3
	:	:	:
	:	:	:
	:	:	:

資源

設計DB（リポジトリ）
・機能設計
・テーブル定義
・項目定義

生成したデータは
①Excel形式でダウンロードできる
②Excelで加工し アップロードできる
③DBへ繰り返し登録できる

テストデータ自動生成　～入力画面イメージ～



5件 Page No.1

No.	テスト利用テーブル			リセット
	テーブル名称	物理テーブル名	□ 対象	
1	真因分析	aacdcsanlys	☑	リセット
2	プロジェクトView	aacdproject_v	☐	リセット
3	PJ追記	aacdprojectps	☑	リセット
4	PJプロセス性能実績.真因分析	aacdpjprcspfrmnccsanlys	☑	リセット
5	PJプロセス性能実績	aacdpjprcspfrmncact	☑	リセット

3件 Page No.1

No.	テスト利用項目(区分)			リセット
	テーブル名称	項目名称	□ 対象	
1	プロジェクトView	案件区分	☐	リセット
2	PJ追記	組織C	☑	リセット
3	PJプロセス性能実績	検定結果	☑	リセット

STEP1：テーブル選択

機能設計情報（CRUD）より
テーブル、区分を表示し
テスト対象を選択する

STEP2：件数入力

選択したテーブル毎に
データ生成したい件数を入力し
テスト対象の区分値を選択する

No.	テスト利用テーブル		
	テーブル名称	■ 必要件数	□
1	真因分析	1	☐
2	PJ追記	2	☐
3	PJプロセス性能実績.真因分析	4	☐
4	PJプロセス性能実績	1	☐

10件 Page No.1

No.	テス		
	テーブル名称	項目名称	□ テスト値
1	PJ追記	組織C	☑ 1:SS事業部
2			☐ NULL
3			☐ 空文字
4			☐ なんでもいい

テストデータ自動生成　～生成データ～

テストケース毎のレコード

テーブル
と
項目

	B	C	D	E	F	G	H	I
3					アップロード時の注意点			
4	TS番号	1			色のついていないセル、および、黄色のセルのデータのみ修正可能です。			
5	シナリオ内容	真因分析名重複チェック			行列の追加および移動はできません。アップロード時にエラーとなります。			
6	TC番号				1	1	1	1
7	生成番号				1	2	3	4
8	テーブル名称	物理テーブル名	項目名称	項目名	生成値	生成値	生成値	生成値
9	真因分析	aacdcsanlys	真因分析ID	csanlys_id	-1131			
10	真因分析	aacdcsanlys	PJ追記ID	projectps_id	-1163			
11	真因分析	aacdcsanlys	PJプロセス性能実績報告I	piprcspfrmncrpt_id	-798822346			
12	真因分析	aacdcsanlys	真因分析ID_EK	csanlys_id_ek	-1131			
13	真因分析	aacdcsanlys	真因分析名	csanlys_nm	-1132:真因分析名XXXXXXX			
14	真因分析	aacdcsanlys	真因分析実施日	csanlys_prctc_dt	2014-06-01			
15	真因分析	aacdcsanlys	登録者	ins_userid	-1133:登録者XXXXXXXXX			
16	真因分析	aacdcsanlys	添付ファイル	attached_file	-1			
17	真因分析	aacdcsanlys	参考URL	re_url	-1134:参考URLXXXXXXXX			
18	真因分析	aacdcsanlys	概要	summary	-1135:概要XXXXXXXXXX			
19	PJプロセス性能実績	aacdpiprcspfrmncact	PJプロセス性能実績ID	piprcspfrmncact_id	-1000			
20	PJプロセス性能実績	aacdpiprcspfrmncact	PJプロセス性能実績報告I	piprcspfrmncrpt_id	-798437597			
21	PJプロセス性能実績	aacdpiprcspfrmncact	プロセス性能C	prcspfrmnc_cd	-1001:プロセス			
22	PJプロセス性能実績	aacdpiprcspfrmncact	検定結果	exam_kbn	3			
23	PJプロセス性能実績	aacdpiprcspfrmncact	添付ファイル	attached_file	-1			
24	PJプロセス性能実績	aacdpiprcspfrmncact	参考URL	re_url	-1002:参考URLXXXXXXXX			
25	PJプロセス性能実績	aacdpiprcspfrmncact	検定結果登録日	exam_ins_dt	2014-04-19			
26	PJプロセス性能実績 真因分	aacdpiprcspfrmnccsanlys	真因分析ID	csanlys_id	-1131	-1131	-1131	-1131
27	PJプロセス性能実績 真因分	aacdpiprcspfrmnccsanlys	PJプロセス性能実績ID	piprcspfrmncact_id	-1000	-1000	-1000	-1000
28	PJ追記	aacdprojectps	PJ追記ID	projectps_id	-1163	-1164		
29	PJ追記	aacdprojectps	組織C	div_cd	1	1		
30	PJ追記	aacdprojectps	年C	fsclyear_cd	2014	2014		
31	PJ追記	aacdprojectps	プロジェクトID	proj_id	-787664628	-1560614945		
32	PJ追記	aacdprojectps	PJ追記ID_EK	projectps_id_ek	-1163			

指定しない項目は
型・桁に合わせて
一定のルールで生成

指定した値で生成
指定した事がわかるように
強調表示

テスト設計システムの主な仕様

- 必要なテーブルを選択するだけで整合性のとれたレコードを生成
- テストに使わない項目は型・桁の制約に則った値を自動生成（テストに使う項目は値を指定できる）
- テストケース毎にテストデータを管理
- 繰り返しテスト環境へ反映できる

テスト設計システムの結果 ①再現テスト

- テスト方法

  本番稼働中システムの照会系機能

  開発時にUTで使用したテストデータを本システムで再現できるか確認
  - テスト数　　:6シナリオ、17ケース
  - テストデータ:6テーブル(計55項目)、のべ60レコード

- 結果

	開発時	再現テスト
データ作成方法	手動登録	**テスト設計システムで生成・登録**
データ網羅率	100%	100%
データ作成時間	4時間	3．6時間

1シナリオ目＝2時間
2シナリオ目以降＝15分/シナリオ

テスト設計システムの結果 ②プログラム種類別テスト

- テスト方法
  - WGメンバー　5名
  - 照会、登録、バッチ処理のプログラムを数本ずつテスト

- 結果

メンバーの声

- テストデータを**直しながら何度でもテストできる**のがうれしい
  （正常に動作するデータを用意できるまでに意外に
  時間がかかっていた）
- **何度でも**データベースへ**反映できる**ので**再テストがし易い**

- 部品表などの**再帰構造**のデータは**自動生成できない**

まとめ

成果

テスト設計システム構築

テストデータ作成に
時間がかかる原因

- テストに必要な
  レコード（件数）の多さ
- テーブル項目の多さ

欠陥を抑える対策
（DB処理の自動テスト実現）

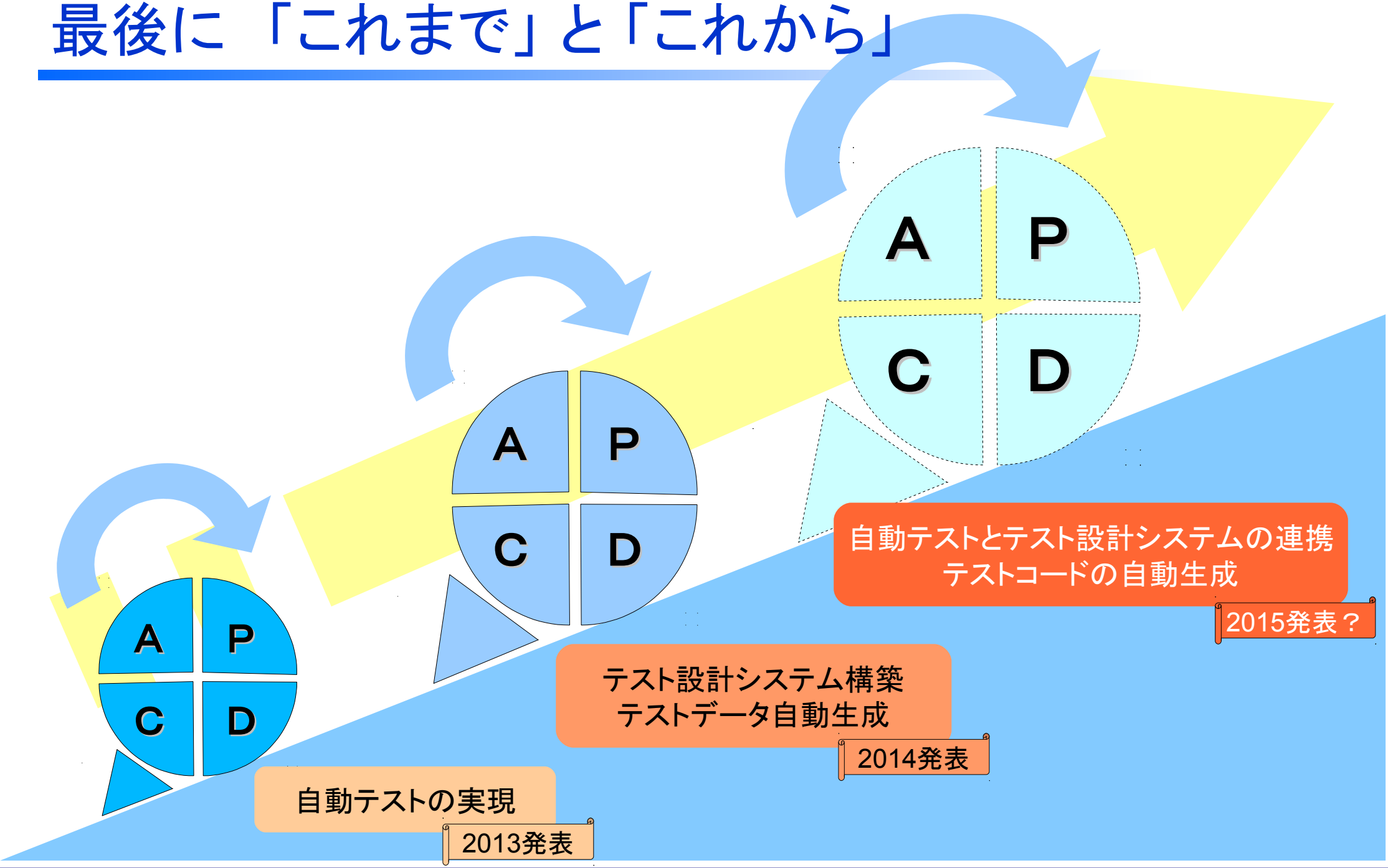
- テスト実行前に
  必要なデータを登録できる
- 繰り返しテストを
  できるようにする

- 1ケース約15分でデータ生成から
  DB登録、動作確認までできた ◎
- 再帰構造など生成できない
  ものがある △
- テスト実行前にDBに登録できる
  繰り返しテスト前の状態に戻せる ◎
- 自動テストPG（JUnit）から
  実行できない △

今後の取り組み

- 開発プロジェクトでの試行・評価、展開
- 初回シナリオ作成の効率化
  - 教育、ノウハウの収集・展開、・・・
- テストデータ生成機能の強化
  - 再帰構造の対応
  - 既存値の流用　　・・・他
- 自動テストとの連携
  - 業務データベースへ反映をテストコードから実行

最後に 「これまで」と「これから」



終わり

ご清聴ありがとうございました。