TOSHIBA
Leading Innovation

システムテスト自動化の普及・展開に向けた取り組み

2017年10月13日

株式会社東芝 ソフトウェア技術センター
小笠原秀人（hideto.ogasawara@toshiba.co.jp）

今回の発表で伝えたい事

- **テスト自動化ツールを使わないのはもったいない！**
  - Ranorex、TestComplete、Selenium、Sikuri、UWSC、etc.
    - 昔から、ツール（X-Runner、Win-Runner、etc.）はあったが、失敗の繰り返し。その反省を踏まえて進め方を考えよう。

- **ツールを使いこなす、テスト実行環境を構築するには、テスト技術者が必須！**
  - リソース（テスト技術者）が確保できなければ自動化は難しい

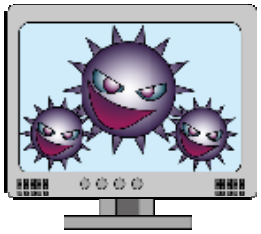
- **テストから入って、開発に深く入り込む！ … オフショア**
  - 簡単なツールや機能の開発から始めてシステムの理解を深め、将来的には、コアの部分の開発をお願いするという作戦は、うまくいかない事が多い…

目次

1.テストにおける背景と解決すべき課題

テスト工数が増大し、減らせない

- **時代の変化への対応**
  - スマートデバイスや各種ブラウザでのテスト
  - 操作性のテスト
  - セキュリティのテスト
  - …



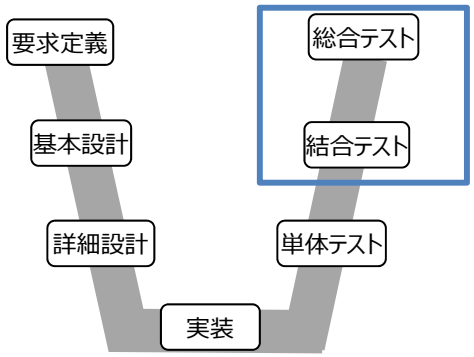
- **テストを減らせない**
  - そもそも減らない
    - 製品の品質保証をするうえでは、網羅的にテストを実施し、問題がないことを説明できなければいけない。
  - 増やさないと対応できない
    - シリーズ製品開発の場合、変更の影響が及ぶと思われる箇所を厚くし、その他の部分も変更の影響がないことを確認するため、大幅にテストが減ることはない。
    - さらに、プラットフォームやブラウザの進化に追随するためにテストが増える。

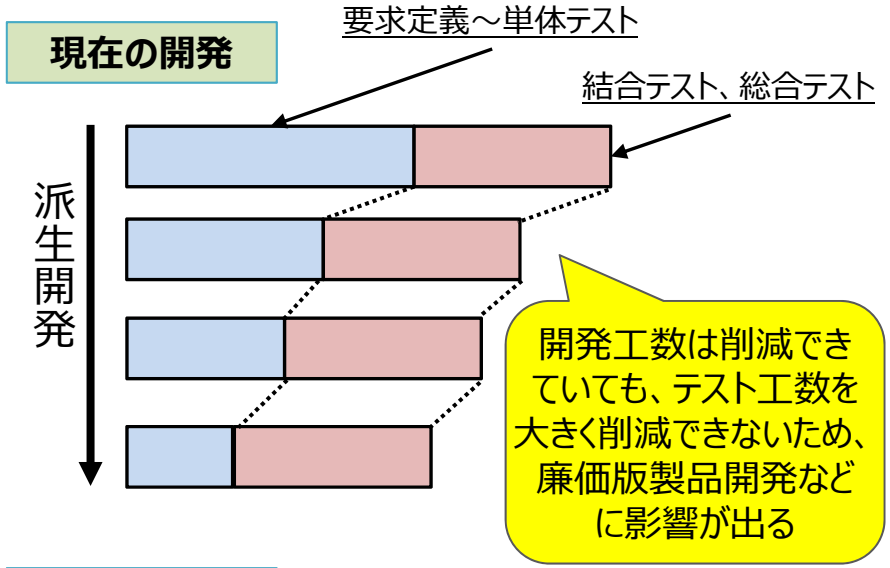
解決すべき課題

製品開発において、「テスト工数」の増大への対応が必須

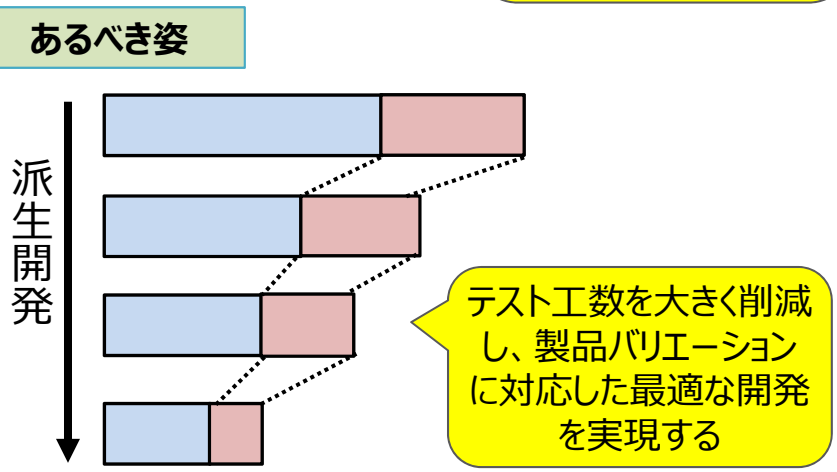
■背景

- ソフトウェア開発のテスト期間とテスト工数の比率は**約30%**に達する（※1）





- IT技術の進化と多様化、ビジネス・技術革新のスピードの加速により多量のテストを短期間に行う必要あり



■課題

①繰り返し行うテストによる工数の増大

②テストの抜けもれによる後戻り工数の増大

※1:ソフトウェア開発データ白書2012-2013,IPASEC

システムテスト自動化導入の阻害要因と解決方法

システムテスト自動化技術を導入するも７割が問題あり（※2）

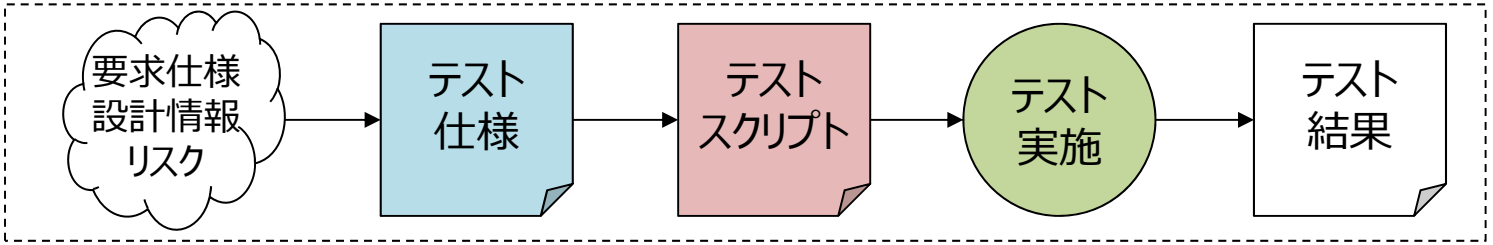
・導入コストが高いため手作業の方が早いと思われる
・テストスクリプトの保守コストが高い

・自動化ツールを使いこなすまでに2ヶ月以上かかる
・スクリプトの変更割合が多く、管理が難しい

Step1の阻害要因：導入・維持コストがかかり効果が出ない

既存のテスト仕様をベースにテストスクリプトを作成・適用し、その後、複数の製品シリーズに展開して、テスト工数を大幅に削減する

Step1：自動化で工数を削減する



システムテスト自動化プロセス

Step2：抜け漏れないテストを作る

Step2の阻害要因：テスト仕様に抜け漏れがあり効率的・効果的に作れない

・自動化しても不具合の発見に繋がらない

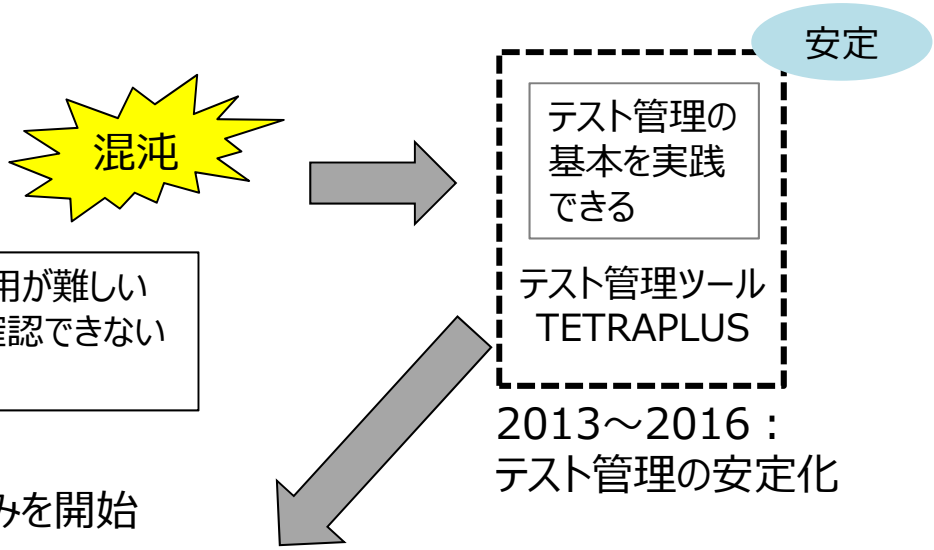
抜け漏れのないテストスクリプトを効率良く開発・維持し、QCD を満たす開発を実現する

Dorothy Graham 2014
※2:Test Automation Problems –Survey results

前提条件と施策

- **導入成功の前提条件**

システムテスト自動化を安定運用するには整然としたテスト管理が前提



～2012年代：テスト管理の課題

Excel/Wordで作成 → バラツキが大きく、再利用が難しい
メールやフォルダで共有 → 過去や最新の状況を確認できない
人手で集計 → 管理工数が増大

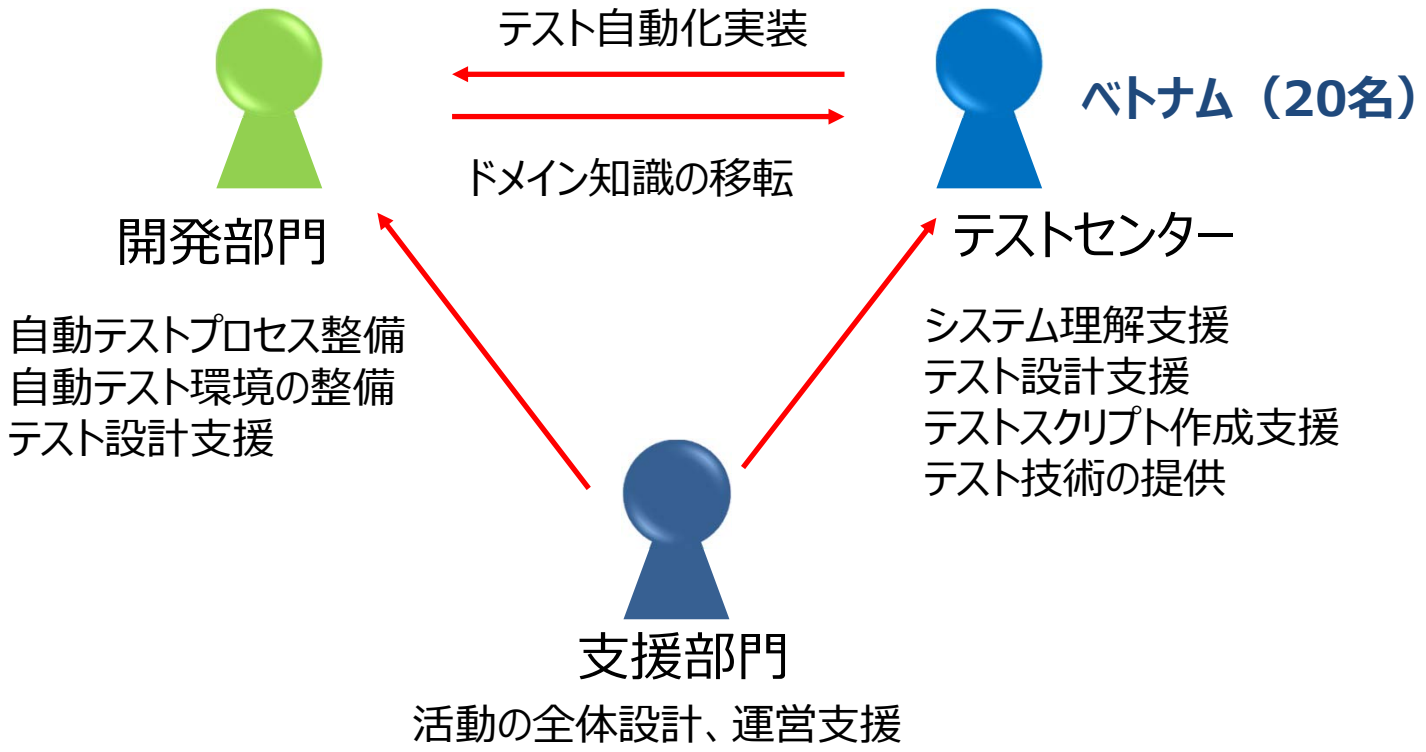
- **施策** 2016～：システムテスト自動化の取り組みを開始

阻害要因	施策のポイント	施策内容
Step1の阻害要因：導入・維持コストがかかり効果が出ない	①専門家の知識により効率的に自動化を導入	専門家による導入支援、テスト自動化ガイド
	②導入・維持コスト削減のための体制を整備	テストセンター活用
	③維持コスト削減のための環境を整備	仕様、スクリプト、実行結果の一元管理環境を構築（TETRAPLUSを軸に）
	④再利用でき保守しやすいテストスクリプトの開発	テストスクリプト開発技術
Step2の阻害要因：テスト仕様に抜け漏れがあり効率的・効果的に作れない	⑤テスト設計の品質向上のための診断と改善	テストプロセス診断・改善技術（テストプロセス成熟度モデルTPI Nextの利用）
	⑥テスト仕様からテストスクリプトを効率よく作る	テストスクリプト自動生成技術

テストセンターの活用

導入・維持コスト削減のためのテスト自動化専門家育成

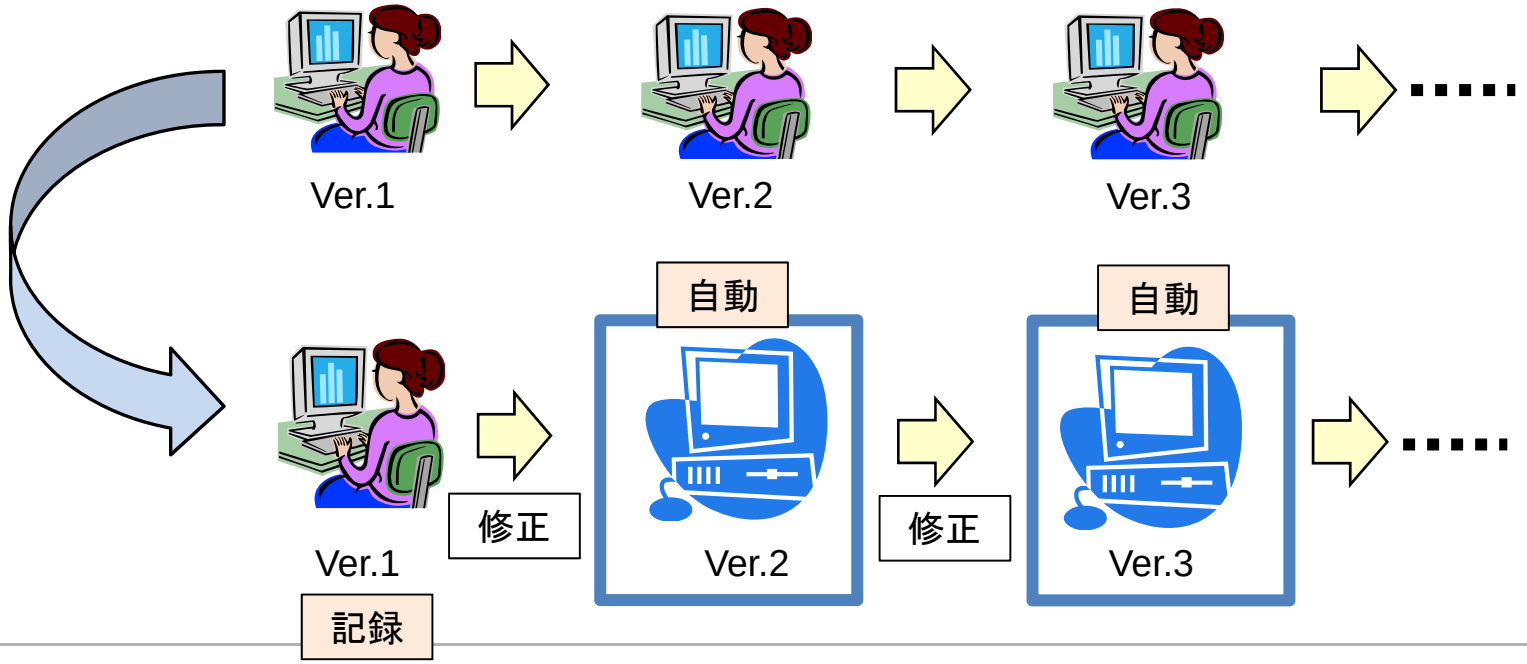
- システムテスト自動化技術を集約し、立ち上がりを早くする
- テストセンターのリソースを活用し、テストスクリプト開発のコスト削減
- テストセンターの専門家によるテストの充実で品質向上を狙う



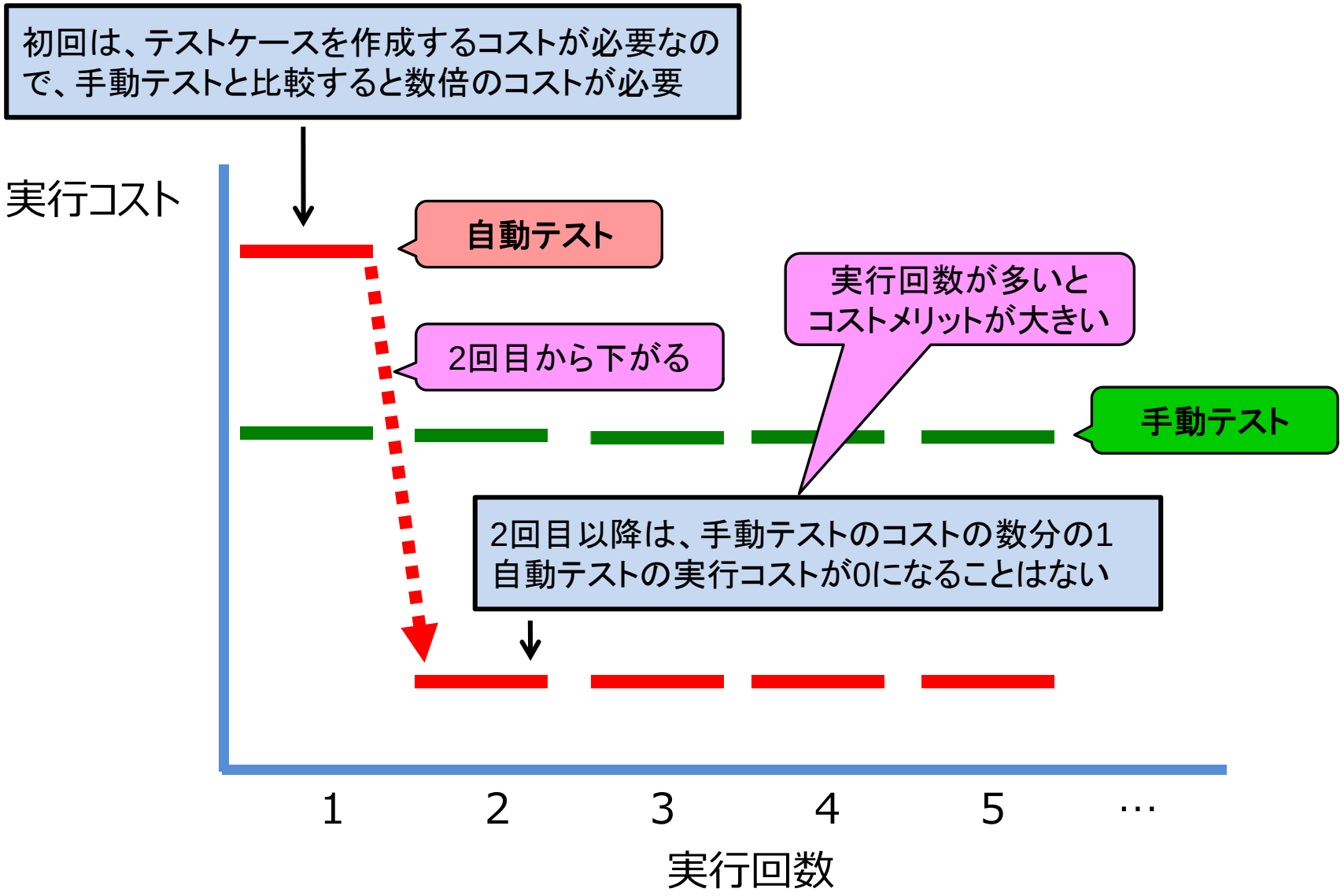
2. テストの自動化とツール

- **テストの自動化とは？**
  - GUIやコマンド処理を通して操作した場合と同等になるように、ツールで操作を再現することで行うテスト

- **利点**
  - 一時実施した事（テスト）を繰り返し実施できる



テストの実行コスト

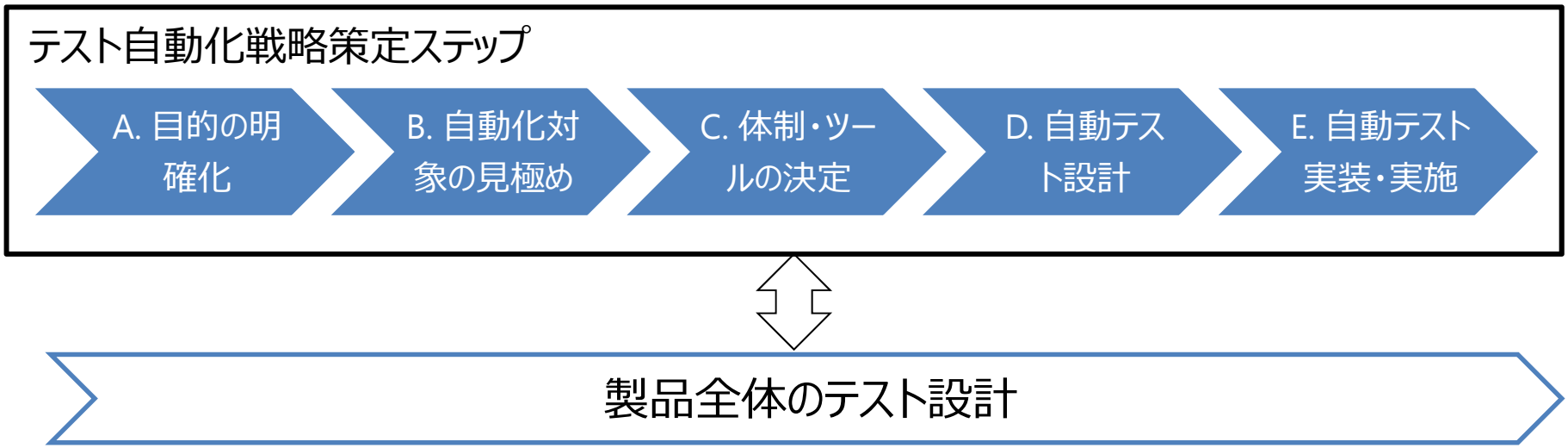


テスト自動化ツールの選定

- **市販ツール、フリーソフトなど、数多くのツールが存在する**
  - テストの自動化を進めるために、事前に検討し、準備しておく必要がある
- **評価項目の例：**
  - 動作環境（OS、ブラウザ、スマートフォン/タブレットなどの対応、ハードウェアのスペック、必要なライブラリ、など）
  - 価格
    - 本体価格と保守費は？
    - フローティングライセンスでよいか、それともノードロックか？
    - ランタイムは必要か？
      - テストスクリプトを編集する頻度と作成したテストスクリプトを実行する頻度は？
  - 安定性（安定して動作するか？）
  - 日本語化（日本語化対応しているか？）
  - スクリプト言語（ツール自体が出力するコードは？ ユーザコードの言語は？）
  - 検証結果の出力（検証結果はどのような言語、あるいは形式で出力されるのか？）
  - UIの操作（ユーザの操作がどこまで記録できるのか？ 何を記録しているのか分かりやすいか？）
  - キーボード入力（特に、記録中にかな漢字変換が安定して使えるか？）
  - 検証方法（実行結果を検証する機能はあるか？ 記録後に検証コードを追加できるか？）
  - スクリーンキャプチャ（スクリーンキャプチャはできるか？ 記録後に追加か削除ができるか？）

3. テスト自動化の進め方

- **テスト自動化を推進するためのプロセス**



実践事例

- **対象**
  - ソフトウェア：システムを監視したり、パラメータを設定したりするソフトウェア
  - テスト環境：システムを模擬できるテスト環境

- **A. 目的の明確化**
  - 品質保証部門によるテスト（1～2週間／1～2人）を効率化する
    - 特に、“心が折れそうになる（膨大で単純な作業の繰り返し）作業” を自動化する
- **B. 自動化対象の見極め**
  - 単純な繰り返しの作業を効率化する
    - 例：サブシステム毎に数千件の確認項目
      - 手作業でデータをセットし目視で結果を確認
  - 膨大なバリエーションの中から合理的に組み合わせのパターンを生成し、網羅性を向上させる
    - 例：スケジュールや確認のタイミングなどを設定しながらテストを実施

- **C. 体制･ツールの決定**
  - 体制：開発部門、テストセンター、支援部門による体制を構築
  - ツール：Ranorex
- **D. 自動テスト設計**
  - オフサイト活動（約10日）
    - ステップ1：ドメインの理解
      - システムを説明した資料（pptで約70ページ）で学習
    - ステップ2：テスト対象の理解
      - 取扱説明書（wordで約170ページ）、テスト仕様書（TETRAPLUSで約300件）の内容を確認
  - オンサイト活動（約10日）
    - ステップ３：システムの動作確認
    - ステップ4　：テストスクリプトの開発方法の決定
      - 独立性の維持（前処理、後処理の方法）
      - モジュール化する部分の見極め、など

- **E. 自動テスト実装・実施**
  - オンサイト期間中に、約40件のテストスクリプトを開発
    - テストスクリプトの生産性について：
      - 算出はできるが比較できるものではないので、提示する際には注意が必要
      - 最初は小さく出発し、どのようにすれば長く使えるかという視点で計画すべき
  - クライアント（Ranorexが入ったPC）と信号の送受信を担うPC間の通信方法とテストスクリプト実装方法の確認（PsExec.exe を利用）
  - 自動テストの利用方法と管理方法、今後の体制を検討

テスト仕様書例（今回の対象とは違うテスト仕様）

タイトル	チェックポイント	テスト方法	成功条件
テスト項目の格納フォルダを変更する	既に登録してあるテスト項目の格納フォルダを変更する	任意のフォルダに登録されているテスト項目を選択し、メニューより、「格納フォルダを変更する」を選択する。表示されたフォルダ選択画面にて、別のフォルダを指定する	ブラウザ再読み込み後、指定したフォルダにテスト項目が格納されている。 元のフォルダにはテスト項目は格納されていない テスト項目を移動した後の画面で、「ごみ箱」という表示がでないこと

4. テスト自動化の成果

- **対象製品：**
  - 社会インフラ系の大規模システム
- **適用期間**
  - 2016年10月～2017年３月までの半年間
- **特徴：**
  - 製品系列シリーズの最初の開発であり、この開発以降数年間に渡り、シリーズ製品を複数リリースする
  - シリーズ製品は、ハードウェア構成には大きな変更がなく、個別の設置場所毎に小規模な仕様変更が入る
- **今回の適用における阻害要因と施策**

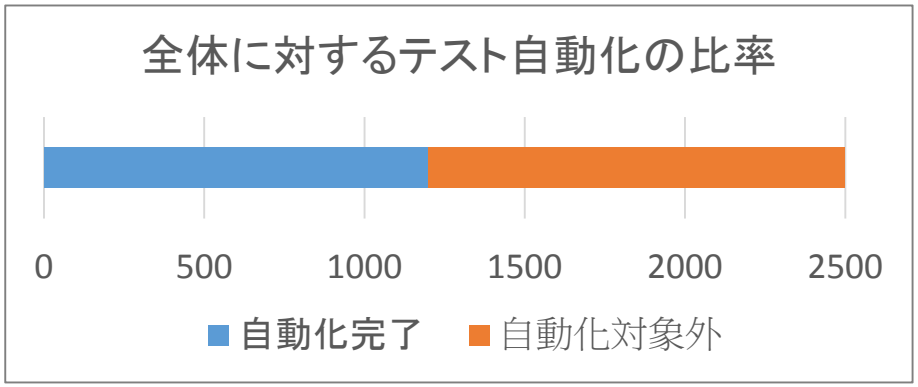
  - 阻害要因
    - 自動化の見極めが難しい
  - 施策①
    - 支援部門が導入支援を行い、テスト自動化の戦略、計画を策定

  - 阻害要因
    - テスト自動化の導入コストが高い
    - 自動化のための体制／プロセスがない
  - 施策②
    - テストセンターを活用したシステムテスト自動化体制・プロセスの整備

施策①の成果

自動化対象を見極め再利用可能なテストスクリプトの開発完了

- 自動化の向き不向き、再利用を考慮したテスト自動化対象の見極め実施（約50%が自動化が有効と判断）
  - 自動化対象の中から他システムの影響を受けにくいテストを選定した1,200件のテストスクリプトの開発完了



- 自動化対象外のテスト
  - 外部機器との接続などが必要となるテスト
  - マイグレーションのテスト
  - 画面の表示結果の確認方法が難しいテスト（40%white、波形、etc.）
  - 例外フロー
  - …

施策②の成果

自動化導入によるコスト効果、品質効果を確認

- **コスト効果**
  - 自動化の導入により次機種以降、テストコストが30%以上削減見込み（図1）
  - テストセンターの利用により初期導入コストを40%削減（図2）

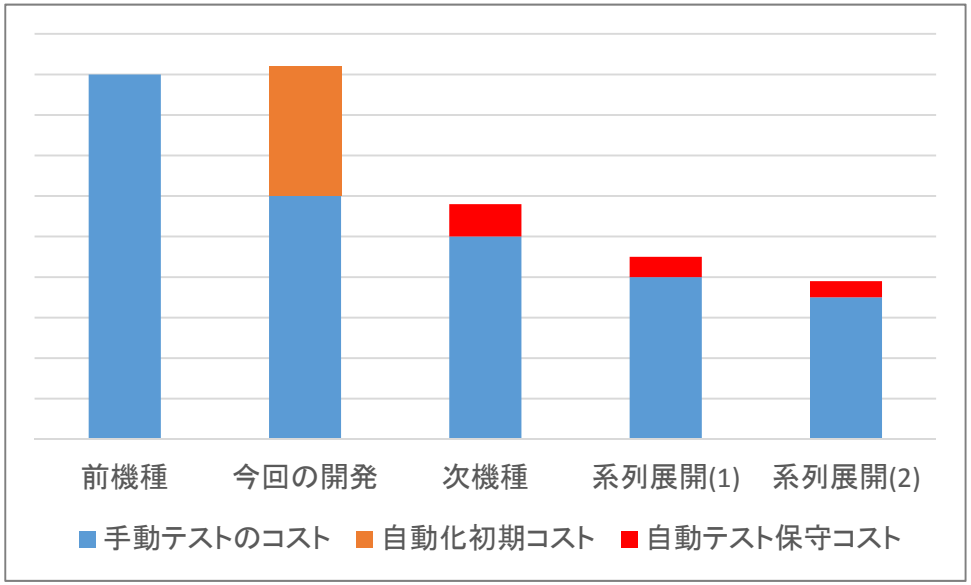


図1 システムテストのコスト推移予測

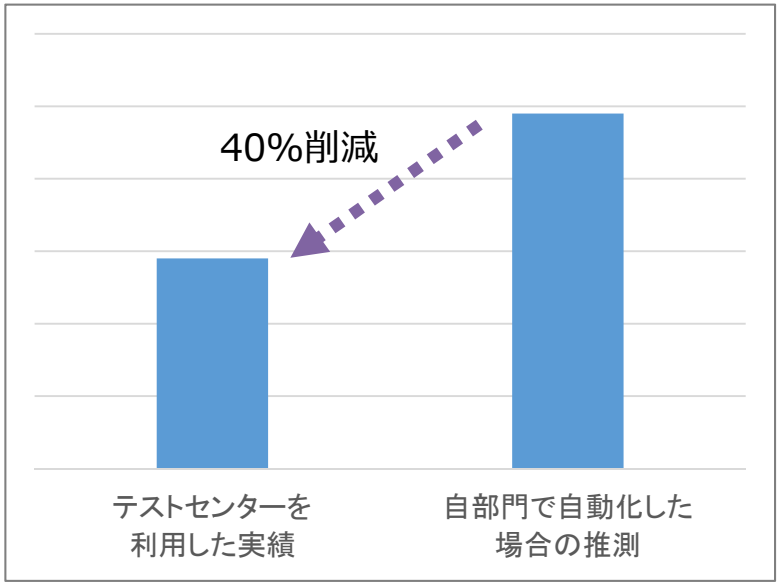


図2 初期導入コストの評価

- **品質効果**
  - 自動化した部分のテストを手動で実施：1週間／人 → 3時間で実行可能
  - 毎晩自動テストを実行。変更に対するデグレードがないことを翌日には確認。

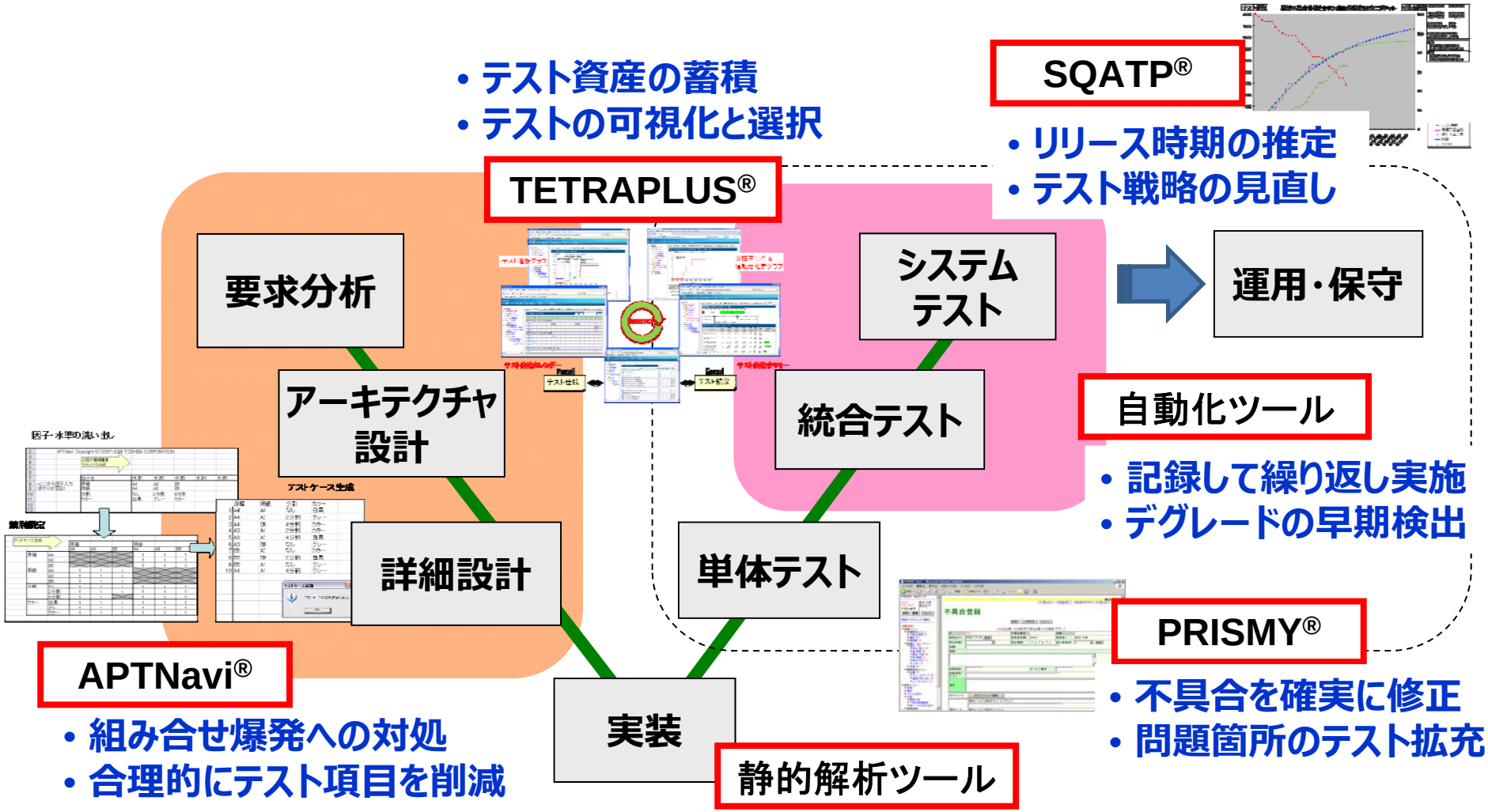
5. テスト自動化を推進するためのポイント

- **テスト技術者は必須**
  - 自動化ツールを使いこなす、テスト環境を構築する、開発したテストスクリプトを保守する、といった役割があり、テスト技術者がいないと自動化は進められない
- **自動化をする前に開発は大丈夫か？**
  - 自動化しようとしても、肝心のソフトウェアが開発途中であったり、品質が安定していないと、どうにもならない。
- **テスト環境を事前によく確認しておくこと**
  - ネットワークに接続できず作業が滞ったりツールのライセンスが取得できない、実機がないと動かせない、実機があっても開発やテストで占有されてしまう、OSやブラウザの制約があり想定したツールが使えない、といったことがないか事前によく確認する必要がある。
- **開発プロセスに組込み、周知すること**
  - 自動化したテストをどのように活用するのかを決め、それを開発メンバに周知する必要がある。例えば、周知が十分でないと、ヒューマンファクターにより自動化が中断してしまうこともある（自動実行中に、機器の切断やテスト対象システムの操作が起こり、自動化が中断する）。

- **海外リソース活用拡大とテスト技術者の育成**
  - 海外リソース活用拡大の方法（今までの経験による）
    - 深く理解し、広く展開する作戦（▲ or ×）
      - 製品開発において、周辺機能やツール系の開発が中心であり、コアの部分を出せない状態が続く事が多い。結局、コアの部分の開発に移行できない。
    - 広く理解し、次に深く入り込む作戦（○）
      - システムレベルの環境や動作が分かるので、システムの全体像を把握しやすい。改善提案も出しやすい。ドロドロした開発状況も理解できる。
  - テスト技術者の育成とローテーション
    - テスト自動化を推進するためには、自動化ツールの活用、テストスクリプト開発、テスト設計、テスト環境構築など、さまざまな技術が必要。
      - 技術者としてのロードマップを示しやすい
    - テストしやすい設計についても思いを巡らせる事ができるので、設計・開発力の向上にもつながる。
      - 設計・開発とテストとのローテーションを回しやすい

付録1. テスト技術の全体像

ソフトウェア開発の最後の砦を守る！品質と効率を両立するツール群の提供



付録2. テスト自動化を推進するための環境

- **基本的な考え方：既存のテスト仕様を活用する**
- **利用するツール：テスト管理ツールTETRAPLUS＋自動化ツール**

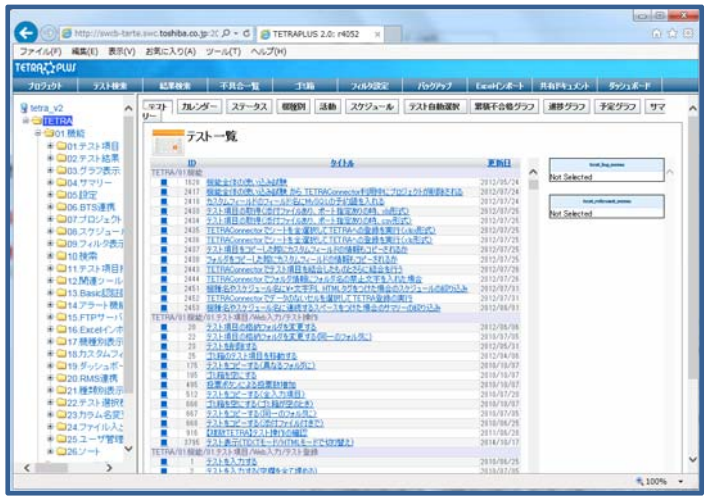
  ※ TETRAPLUS の説明 ⇒ 22～26ページ

  ※ TETRAPLUS と自動化ツールの連携イメージ ⇒ 27ページ
- **自動実行の仕組み：継続的インテグレーションツールJenkinsを活用**

テストTETRAPLUSの概要

- **機能**
  - テスト資産（テスト項目とその結果）を**保管する**
  - 最新のテスト実施状況を**表示する**
  - 蓄積されたテスト資産をさまざまなビューから**可視化する**
  - テスト資産を**再利用する**ための仕組みを提供する

- **ねらい**
  - テスト情報の共有
    ・曖昧に成りがちなテスト（システム）仕様を
    チーム全体で共有
  - リアルタイムなテスト進捗の確認
    ・最新のテスト状況を分かりやすく、手早く確認
    ・ボトルネックの追跡が容易
  - テスト管理作業の軽減
    テスト結果の収集→グラフ化作業を自動化

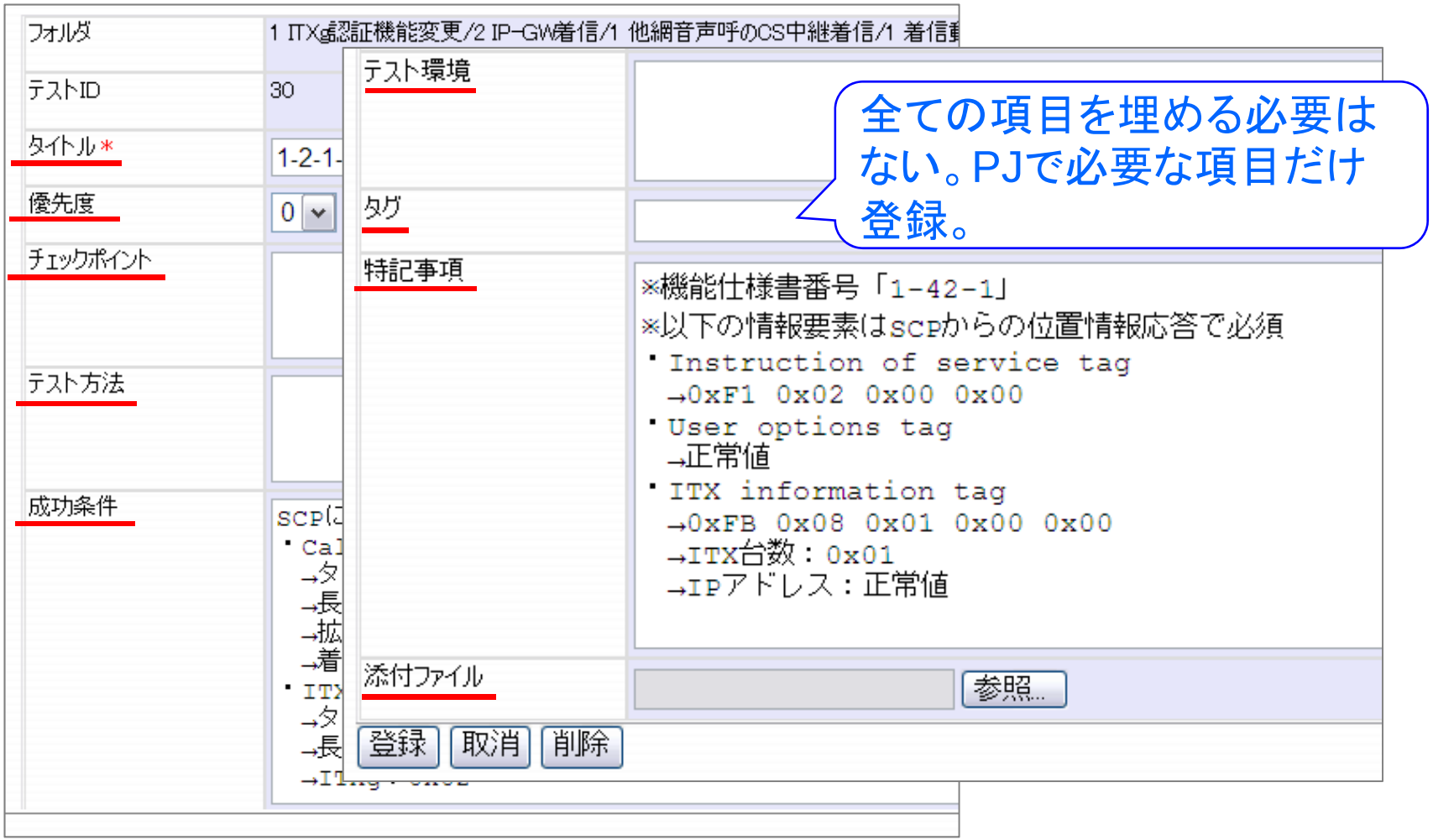


Web+DataBaseによりテスト項目とテスト結果を一元管理する

TETRAPLUSのユーザインタフェース

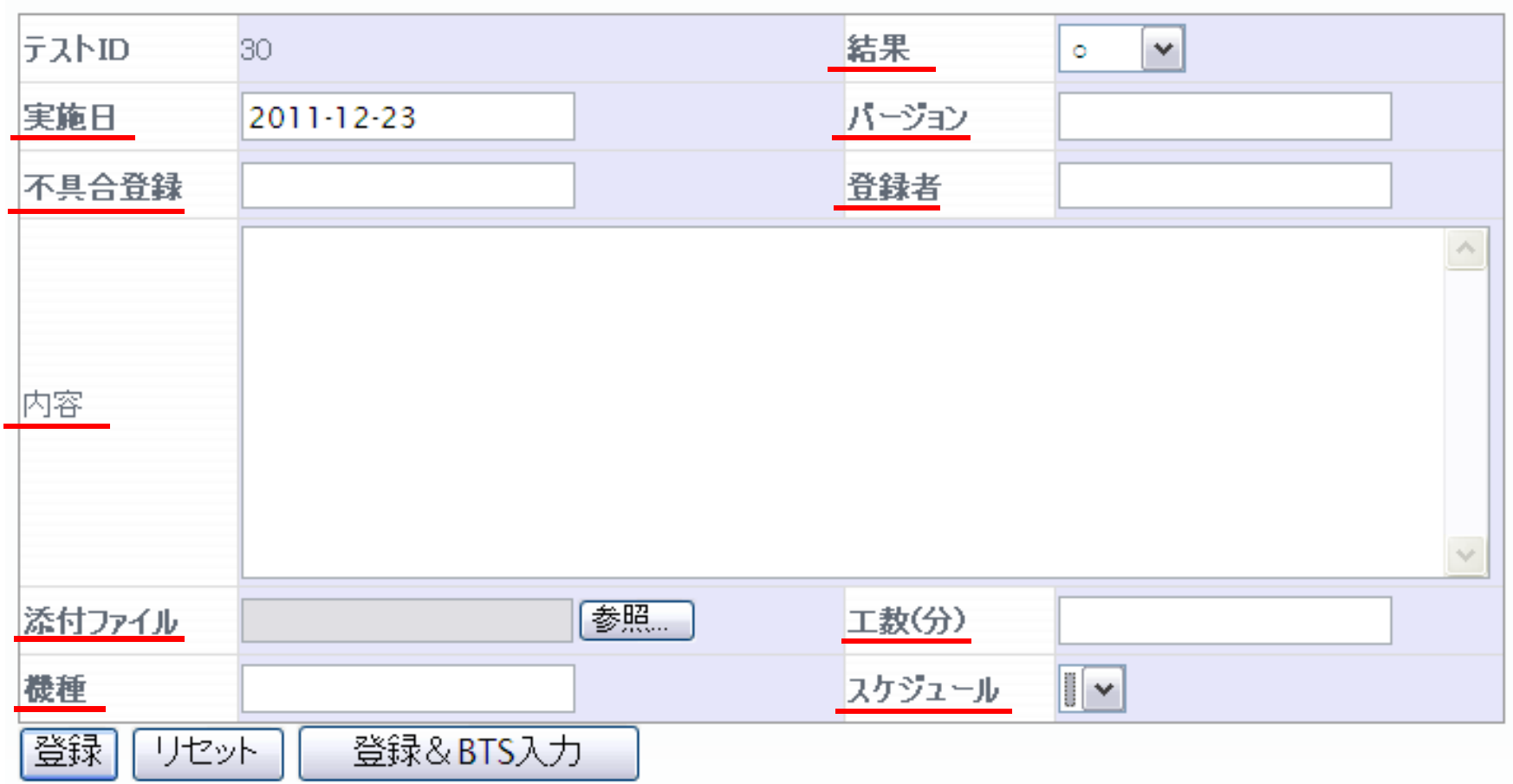
テスト項目

タイトル(必須)、優先度、チェックポイント、テスト方法、成功条件、テスト環境、タグ、特記事項、添付ファイル



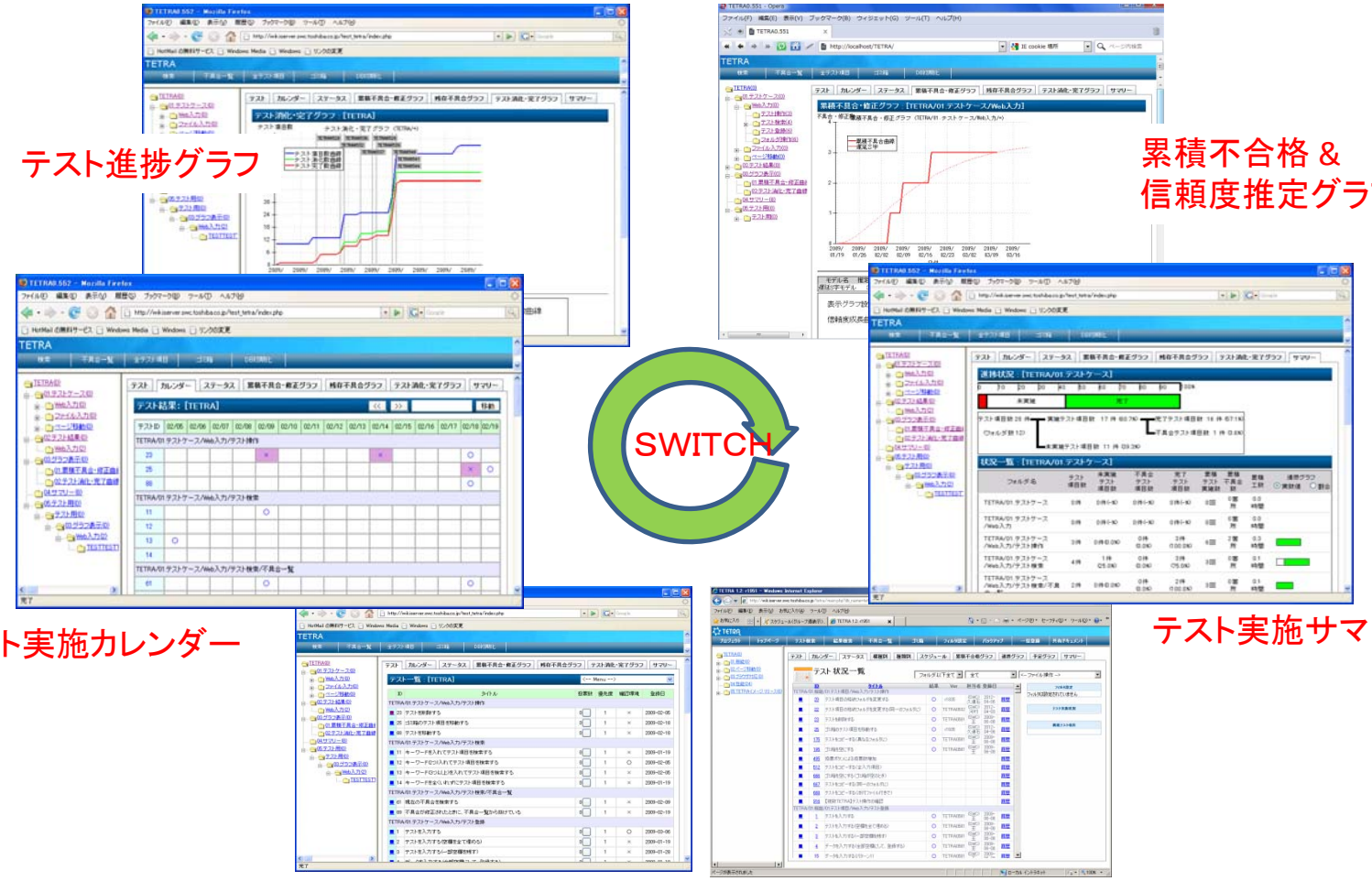
テスト結果

実施日、結果、不具合登録、バージョン、登録者、内容、添付ファイル、機種、工数、スケジュール



テスト実施状況の分析

- 担当者、用途ごとにビューを切り替えテスト状況を確認・分析する



TETRAPLUSと自動化ツールとの連携

仕様、スクリプト、テスト結果を一元管理し、可視性・保守性の向上

ID	タイトル	チェックポイント	テスト方法	成功条件	テスト環境	特記事項	優先度	タグ	ファイル

・TETRAPLUSのIDと紐付けてテストを実装することで保守性を向上

試験仕様

テスト管理ツール

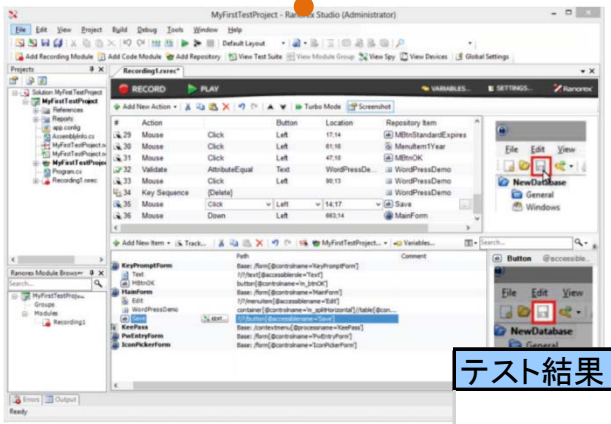
TETRAPLUS

試験成績書

試験仕様
＋
試験結果

テスト
スクリプト

自動テストツール



・サマリー機能で自動実行の結果を即座に確認

テスト結果	登録者	実施日	バージョン	不具合登録	内容	工数	添付ファイル	機種

試験結果

・TETRAPLUS連係ツールで、
試験結果をTETRAPLUSに 登録


TOSHIBA
Leading Innovation