リスク・ベース・テストに基づく テスト自動化戦略の策定

Automated Testing Strategy based on Risk-based Testing

日本アイ･ビー･エム GBS. EA&T, SOA技術 上甲 貴広
（EA&T, SOA Engineering, IBM Japan Takahiro JOKO）

日本アイ･ビー･エム GBS. EA&T, インダストリーアプリケーション 太田 健一郎
（EA&T, Industry Applications, GBS, IBM Japan Kenichiroh OHTA）

Agenda

- はじめに
- GUIテスト自動化の失敗
- GUIテスト自動化における課題
- テスト自動化戦略策定プロセスの外観
- 品質リスクの分析手順
- テスト自動化範囲の決定
- テスト自動化戦略の策定
- プロジェクトへの適用事例
- まとめ

はじめに

テスティングの重要性は高まる一方

- ✓ 構築するシステムの大規模化・複雑化
- ✓ すべてを開発するのではなくフレームワークやアセットの組合せ

テスト作業を支援する各種ツールが一般化

- ✓ 単体テストツール
- ✓ 静的コード解析ツール
- ✓ バグトラッキングツール
- ✓ 負荷（性能）試験ツール，etc

特に「GUIテスト自動化ツール」の期待度は高い

- ✓ キャプチャ・リプレイ機能で，テスト作業を記録
- ✓ テスト結果を何度でも繰り返し実行して、結果を比較可能
- ✓ デグレードの検出 ／ テスト工数の削減

GUIテスト自動化の失敗

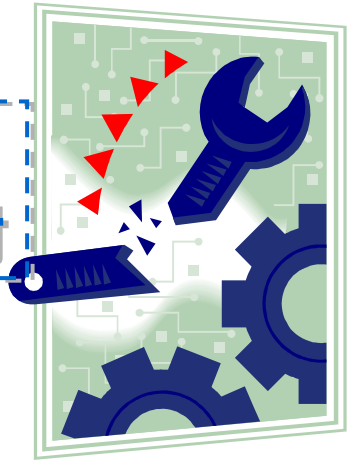
GUIテスト自動化プロジェクトの失敗

- ✓ 当初の期待ほどには効果があがらない・・・
- ✓ 保守工数が負担となり自動テストスクリプトを廃棄・・・
- ✓ 不適切なテストを自動化したためにテストに漏れが発生・・・

テスト自動化の失敗は「テスト戦略」の不備も一因

- ✓ ツールの機能は高機能化する一方
- ✓ ベンダーからツールの **Tips** といった情報も取得可能
- ✓ しかし！　どのようにテスト自動化を適用すべきかのノウハウが不足

リスク・ベース・テストを適用した、
「テスト自動化戦略」の策定方針を検討

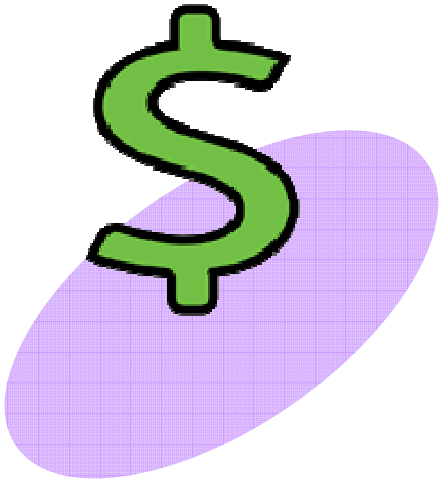


GUIテスト自動化における課題（１／３）

プロジェクトのリソース（お金や時間）が無くなってきたね・・・
よし！　**テストを自動化することでコストを削減**しよう！

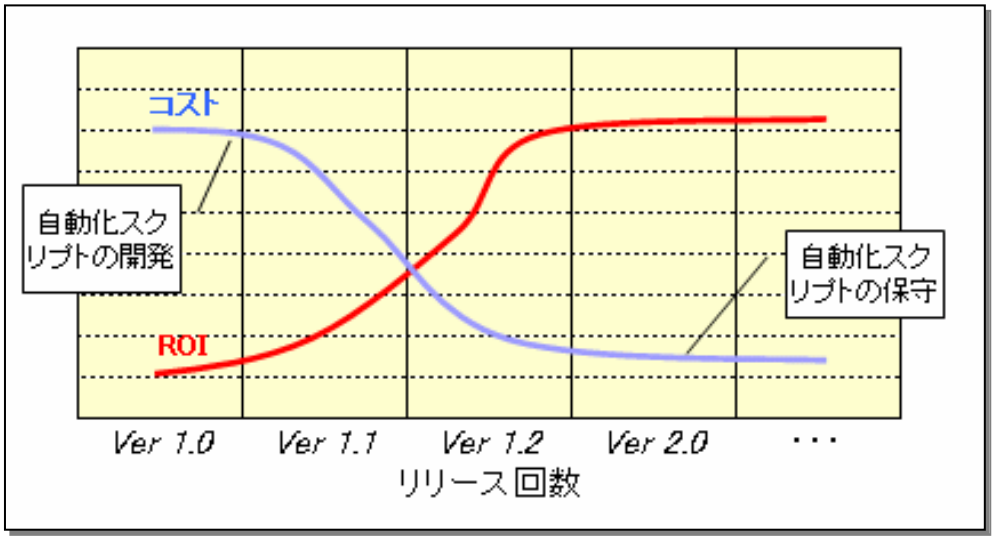
GUIテスト自動化に必要なコストはさまざま・・・

- ✓ ツールの購入コスト
- ✓ ツールの学習コスト
- ✓ テストスクリプトの開発コスト
- ✓ 自動テスト専用のテスト環境の構築と保守・運用コスト
- ✓ テストスクリプトの保守・運用コスト

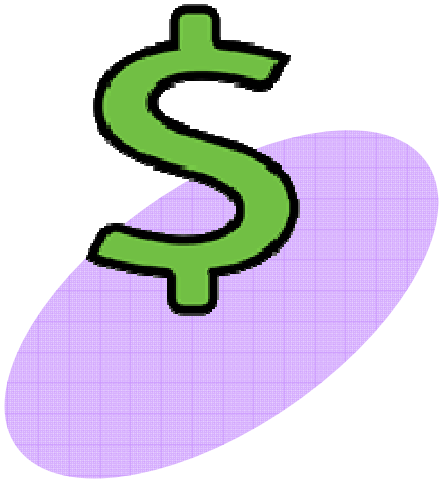


GUIテスト自動化における課題（２／３）

- システムの初期リリースでは工数削減につながらない
  - ✓ はじめにテストスクリプトの開発が必要
  - ✓ システムを改修するたびにテストスクリプトの保守が必要
  - ✓ システムが安定すれば保守工数も低下して投資利益率が向上



- 単純にテスト工数の削減を目的とすべきでない！

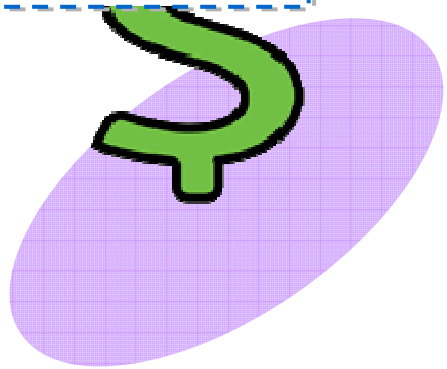


GUIテスト自動化における課題（３／３）

- 自動テストの主目的はテスト対象の品質の向上
  - ✓ 手動テストと自動テストは同一対象に対する異なる検査手法（テストで得られる情報の質や量が、両者で異なる）
  - ✓ 手動テストと自動テストが補完関係を築けば、品質向上に大きな効果アリ
  - ✓ 欠陥発生に起因するコストを抑制可能！

- テスト自動化戦略の狙い

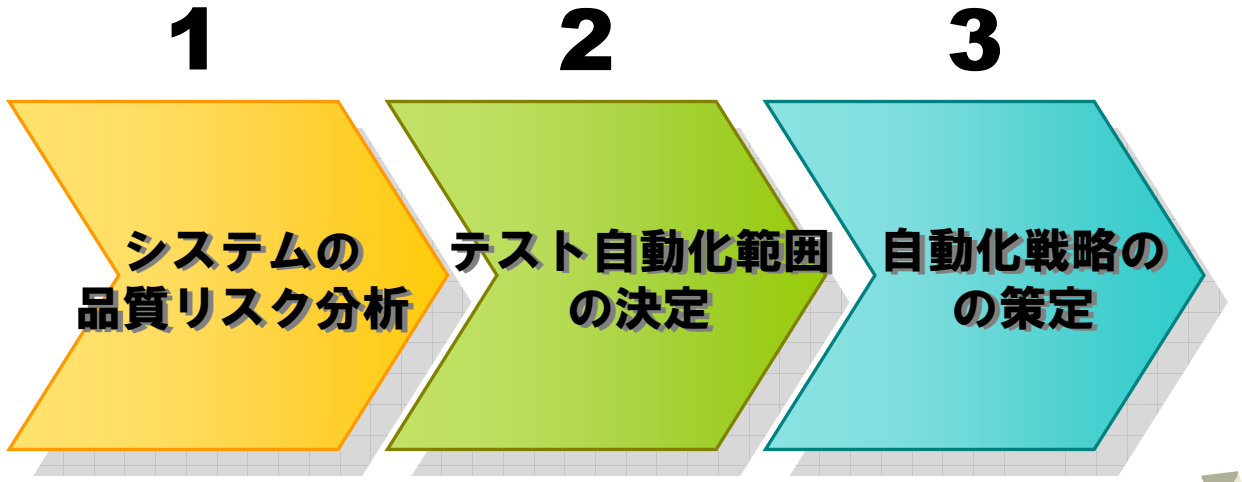
1. テストスクリプトの開発コストは大きいので、適切に自動化対象のフォーカスを絞るべき
2. システムの品質向上に、特に効果のある領域を狙う

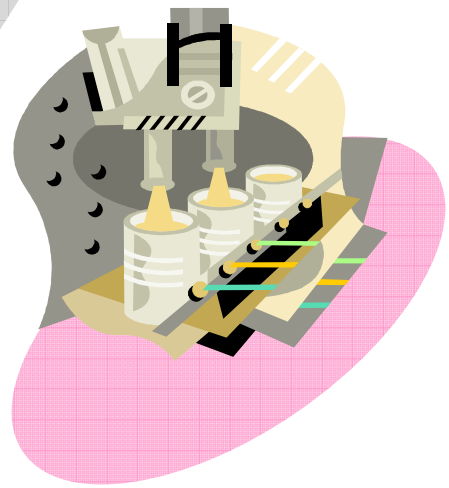


テスト自動化戦略策定プロセスの概観

テスト自動化戦略を策定するためのアプローチ

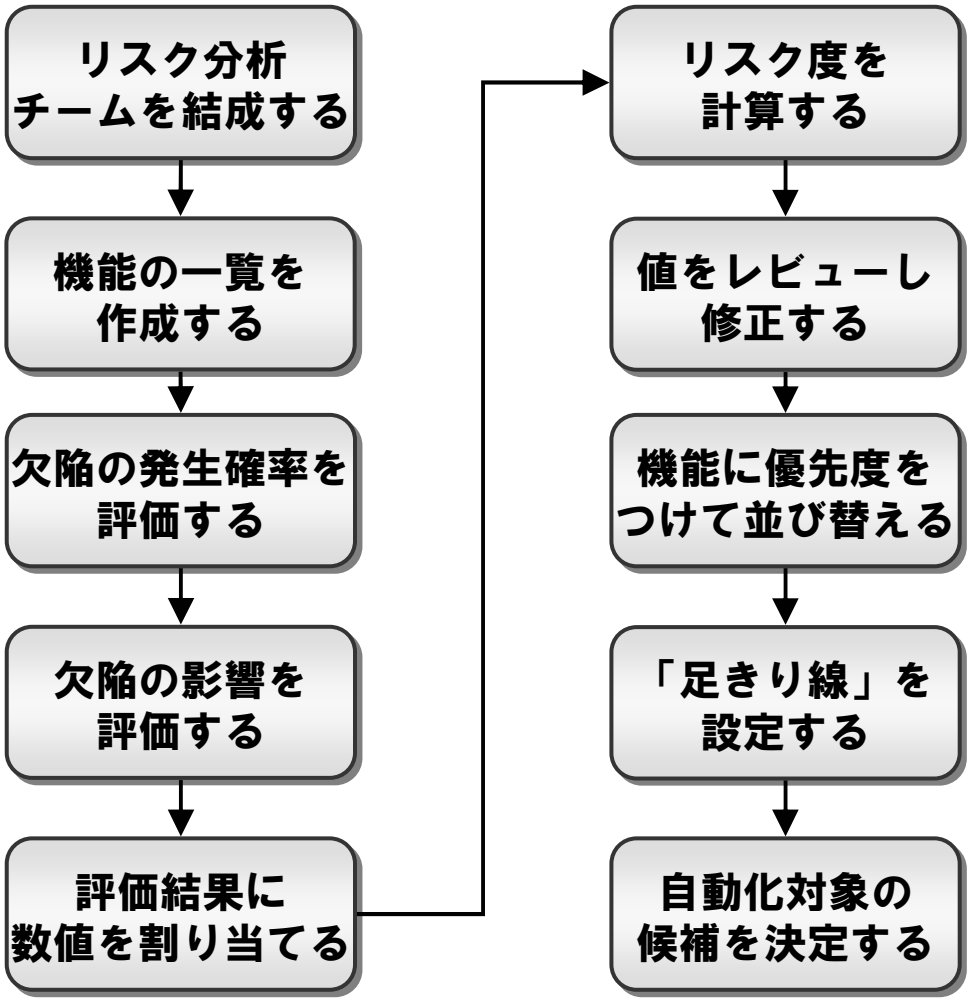
- ✓ 品質リスクを分析して、自動化すべき領域のリスク順位を明確化
- ✓ 分析結果から、自動化範囲とテストスクリプトの開発優先順位を決定
- ✓ テスト自動化プロジェクトの各種方針を確定





品質リスクの分析手順（１／４）

- システムの品質リスク分析手順
  - ✓ Rick Craig らによって提唱される、リスク・ベース・テストにおけるリスク分析手法を適用



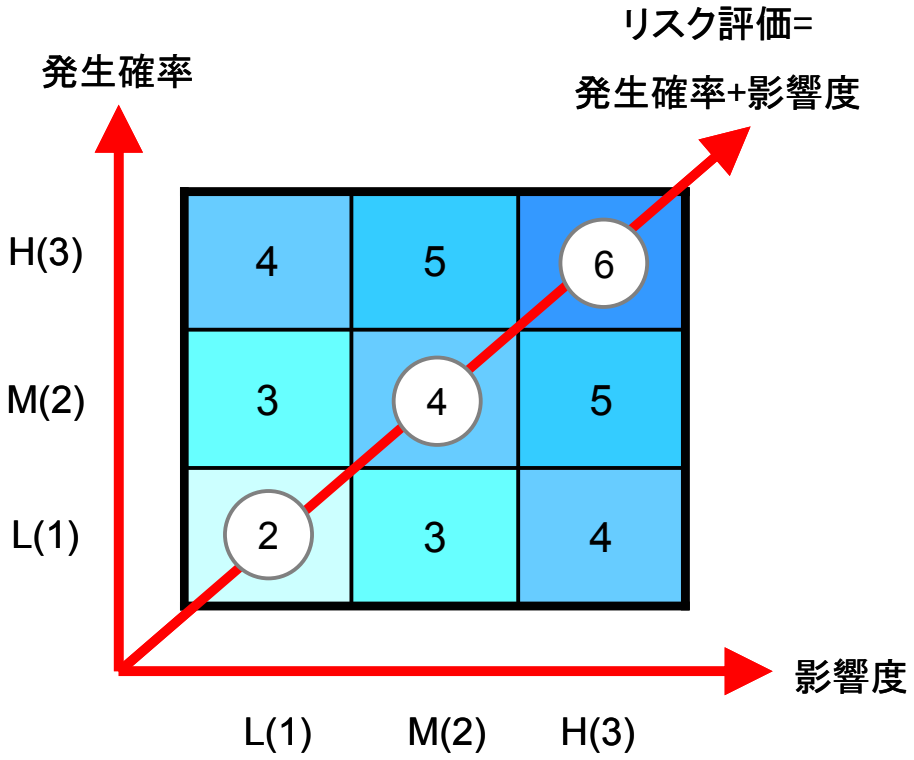
品質リスクの分析手順（２／４）

- **Step.1** リスク分析チームを結成
  - ✓システムと業務に詳しいメンバーでリスク分析のためのチームを結成
- **Step.2** 機能一覧を作成
  - ✓機能，業務，ユースケースなどシステム全体をカバーする機能一覧を作成
- **Step.3** 欠陥の発生確率を評価
  - ✓「欠陥発生の相対的な確率」を評価して指標化
- **Step.4** 欠陥の影響を評価
  - ✓「欠陥がビジネスに与える相対的な影響度」を評価して指標化

CDレンタルシステム	発生確率	影響度	リスク評価
機能/業務			
1. CDを検索する	Low	Medium	
2. CDを借りる	Low	High	
3. CDを返却する	Low	High	
4. CDを予約する	High	Low	
5. 予約をキャンセルする	Low	Low	
6. 延滞料を払う	Medium	High	

品質リスクの分析手順（３／４）

- **Step.5　評価結果に数値を割り当て**
  - ✓評価結果の指標（High，Middle，Low）を数値に変換
- **Step.6　リスク度を計算**
  - ✓「発生確率」と「影響度」からリスク度を算出
- **Step.7　リスク度をレビューして修正**
  - ✓リスク度の算出結果を、必要に応じて修正



品質リスクの分析手順（４／４）

- **Step.8　機能一覧を並び替え**
  - ✓機能一覧をリスク順で並び替え
- **Step.9　「足切り線」の設定**
  - ✓このライン以下は、自動テストしないという基準を設定
- **Step.10　自動化対象候補の決定**
  - ✓足切り線をもとにテスト自動化対象の候補を決定

CDレンタルシステム	発生確率	影響度	リスク評価
機能/業務			
6. 延滞料を払う	Medium	High	5
2. CDを借りる	Low	High	4
3. CDを返却する	Low	High	4
4. CDを予約する	High	Low	4
1. CDを検索する	Low	Medium	3
5. 予約をキャンセルする	Low	Low	2

テスト自動化範囲の決定

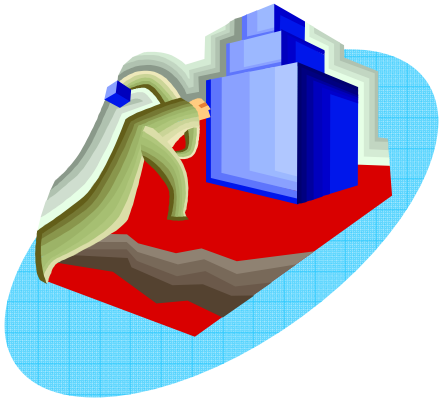
- リスク分析結果をもとにテスト自動化の範囲と優先順位を決定

✓ 自動化**対象外**／スクリプトの開発優先順位が**低い**

- **GUI** の安定度が低い
- **GUI** が特殊／極めて複雑
- 該当機能を実行可能なテスト環境に強い制約アリ
- 次期リリースでの大幅な改訂を予定

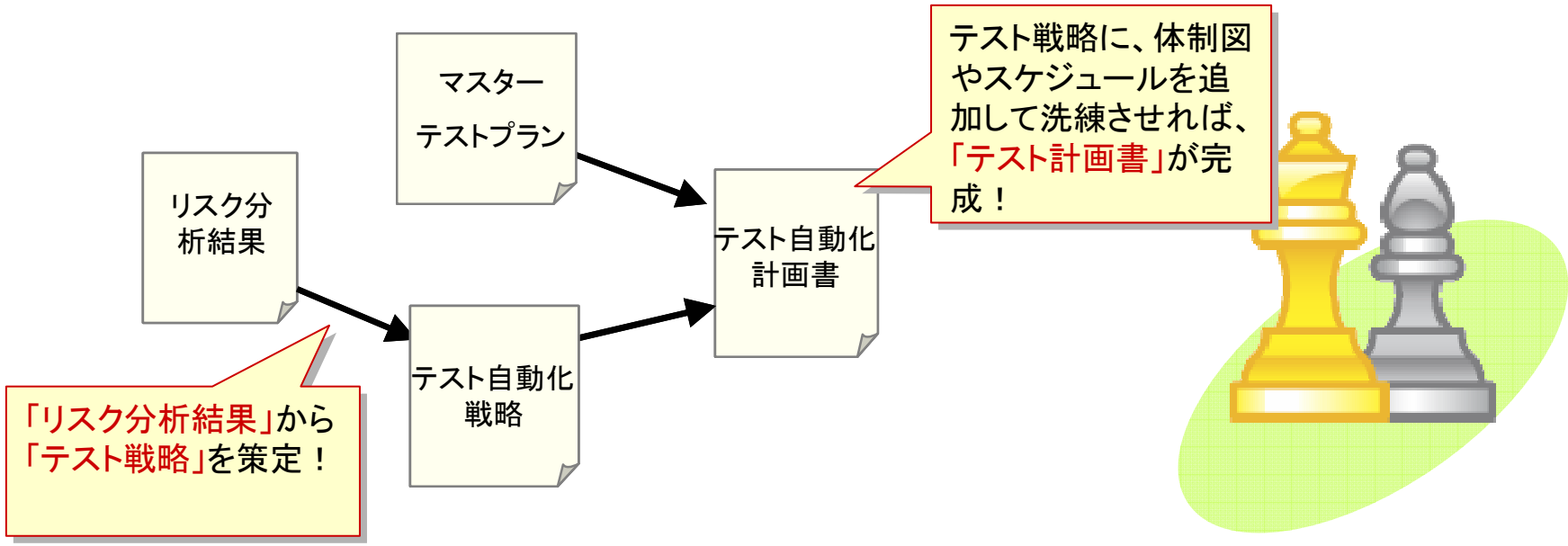
✓ 自動化**対象**／スクリプトの開発優先順位が**高い**

- システムの最も基本的な機能
- デグレードによるビジネスへの影響度が大
- **GUI** の安定度が高い
- 単純あるいは長時間の作業を繰り返すテスト
- 次期リリースでの改訂の予定が無く、保守運用フェーズ でもテストスクリプトが継続適用可能



テスト自動化戦略の策定（１／４）

- リスク分析結果をもとに以下のような方針を文書化
  - ✓ リスク分析結果に基づく自動化範囲
  - ✓ テストスクリプトの開発優先順位
  - ✓ 採用するテスト自動化ツール
  - ✓ テストスクリプトの開発方針
  - ✓ テストスクリプトの管理方針
  - ✓ 自動テストの実施と運用の方針
  - ✓ 自動テストケース一覧
- テスト自動化プロジェクトの成果物関連図



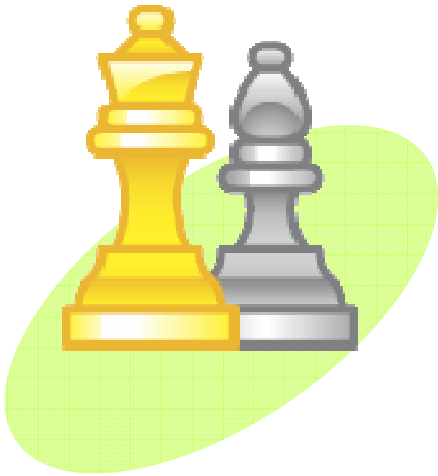
テスト自動化戦略の策定（２／４）

採用するテスト自動化ツール

- ✓ 利用する自動化ツールの基本的な機能や利用方法を記述
- ✓ 自作ツールをテストスクリプト組込む場合は、その仕様や利用方法も記述
  （例：データベースのデータダンプの比較ツール、など）

テストスクリプトの開発方針

- ✓ 均質で保守容易なスクリプトのためのルール
- ✓ 以下のような方針を決定
  - テストスクリプトの命名規則
  - テストスクリプトの粒度の基準
  - テストスクリプトのコーディング標準
  - 検証ポイントの設定標準
  - テスト実施ログの出力標準
  - 共通コンポーネントの API 利用方法
  - サンプルのテストスクリプト



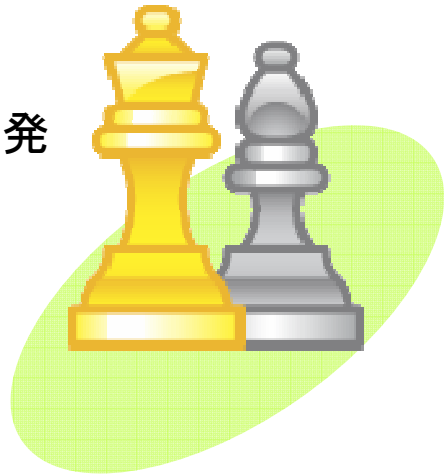
テスト自動化戦略の策定（３／４）

テストスクリプトの管理方針

- ✓ テストスクリプトの構成管理ツールによる管理
- ✓ テストスクリプト自体の欠陥のバグトラッキングツールによる管理、など

自動テストの実施・運用方針

- ✓ 自動テスト専用環境の運用手順
- ✓ 自動テストを実行するタイミング
  （夜間に実行する、ビルド終了毎に実行する、など）
- ✓ 実行するテストスクリプトの種類
  （ビルド終了毎に主要なテストを実行、夜間に全スクリプトを実行、など）
- ✓ 自動テスト結果の報告手順
- ✓ 自動テスト失敗時の原因分析手順
- ✓ テスト自動化と協業する他チーム（ライブラリ管理チーム、開発者チーム、手動テストチーム）の主管代表者

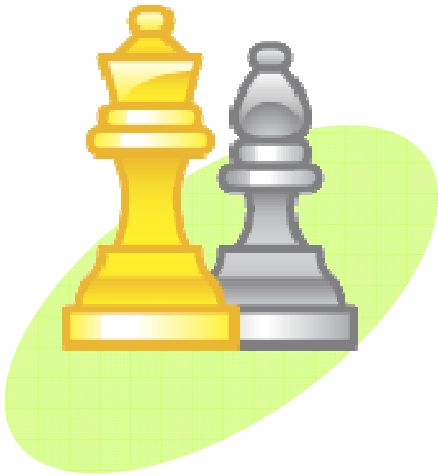


テスト自動化戦略の策定（４／４）

自動テストケース一覧

- ✓ 自動テストのためのテストケース一覧を記述
- ✓ テストケースの詳細は「自動テストケース仕様書」などの別仕様書に記述

テスト自動化プロジェクトの各種方針を定めた
「テスト自動化戦略」が完成！



プロジェクトへの適用事例（１／３）

- 大規模金融系システム構築プロジェクトの回帰テスト自動化
  - ✓ 最大で開発者が数百人の大規模プロジェクト
  - ✓ テスト自動化チームは10人以下で、サービスインまでの期間も僅か
  - ✓ 人的・時間的なリソースの制約が強い

効果的かつ効率的な自動化を実現するためには
投資対効果の高い領域にフォーカスを絞る必要アリ

プロジェクトへの適用事例（２／３）

品質リスク分析の結果

- ✓ マトリクスの縦軸は、金融商品の年間取引数
- ✓ マトリクスの横軸は、基本的業務と応用（例外的）業務
- ✓ 縦軸で「欠陥の発生確率」と「影響度」をまとめて評価
- ✓ 年間取引数が少ない金融商品でも、基本的業務だけは自動テストの対象に

横軸は、業務シナリオのバリエーション

商品種別		年間取引数	基本ケースID		応用ケースID			
大分類	小分類		業務１	業務２	業務３	業務４	業務５	業務６
グループA	A1	2,952	A1-1	A1-2	A1-3	A1-4	A1-5	A1-6
	A2	75,694	A2-1	A2-2	A2-3	A2-4	A2-5	A2-6
	A3	30,288	A3-1	A3-2	A3-3	A3-4	A3-5	A3-6
グループB	B1	7,177	B1-1	B1-2	B1-3	B1-4	B1-5	B1-6
	B2	112	B2-1	B2-2				
	B3	382	B3-1	B3-2				
グループC	C1	20	C1-1	C1-2				
	C2	13	C2-1	C2-2				
	C3	4	C3-1					
グループD	D1	0	D1-1					
	D2	6	D1-2					
グループE	E1	2	E1-1					
	E2	4	E1-2					
	E3	9	E1-3					
	E4	3	E4-1					

縦軸は、金融商品の年間取引数

グレーの領域については、自動テストの「対象外」！

プロジェクトへの適用事例（３／３）

テスト戦略策定と自動テスト実施

- ✓ 分析結果をもとに、自動化範囲やテストスクリプトの開発優先順位を決定
- ✓ 優先順位の高いものから反復的に自動化に取り組む
- ✓ 取引数が多い金融商品を対象としたテストスクリプトは、１日に数回実行
- ✓ 全テストスクリプトは、１日に1回（夜間）実行
- ✓ サービスイン後の保守運用フェーズでも、自動テストはリリースプロセスに含めて運用

費用対効果（自動テストによる欠陥の検出・修正コストと自動テストスクリプトの開発・保守のコスト）を考慮すると、おおむね満足いく結果を得た

まとめ

- テスト自動化の成功にはツールの機能より**戦略が重要**
- 効果的かつ効率的な自動化を実現するためには**フォーカスを絞る**（テスト自動化の実現にはコストを要する）
- テスト自動化には**リスク・ベース・テスト**が有効
- リスク分析結果をもとに各種方針を決定して**「テスト自動化戦略」**を策定
- 大規模金融系プロジェクトへの適用例のご紹介