

やってみました！探索的テスト

～探索的テスト導入から運用にこぎつけるまでの道のり～

2018/9/7

金谷 和博

東京エレクトロン テクノロジーソリューションズ（株）
ソフト技術部



目次

1. はじめに	3
2. 実施の効果	7
3. 探索的テスト実施フロー	9
4. 工夫しているポイント	16
・チャーター作成	
・試験環境	
・振り返りの実施	
・仕様書の廃止	
・Daily朝会の実施	
・手順化	
5. まとめ	23

はじめに

はじめに

- 探索的テストとは

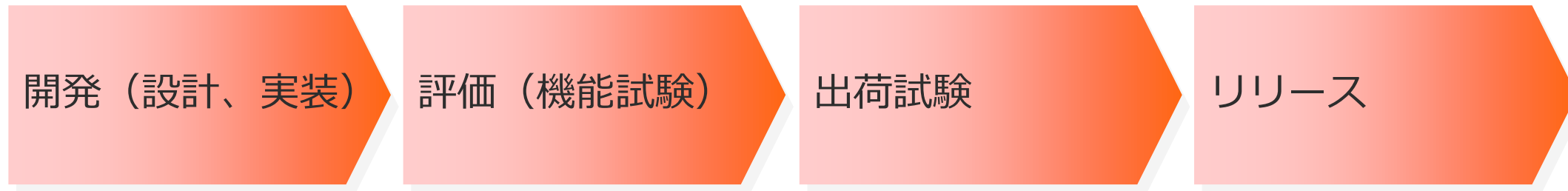
探索的テストとは、様々な視点やテスト技術者の経験から気になるところを「探索」しながら「テスト」することを言う。テストを網羅的におこなうのではなく、怪しい箇所を探し出し、順次絞り込んでいくアプローチのテスト技法であり、アドリブでおこなうテストとも言える。
チャーター（原則）と呼ばれる文書でテストの方向性を指定することもある。

発表内容

- 我々が扱う製品の出荷後不具合がなかなか減らず、根本的な対策が必要な状況でした。
 - そこで、他グループで実施している探索的テストを導入することに決めました。
 - メンバーも品質に危機感を持っており、導入はスムーズにいきました。
 - 実際にやってみたら出荷前に効果的に不具合を発見できました。
 - 探索的テストは定着し、さらなる改善をしながらよりよいテストフローを模索しています。
-
- 本日は、探索的テストフローのポイント、定着させるコツなどをお伝えします。

テスト全体像

ソフトウェア開発フロー



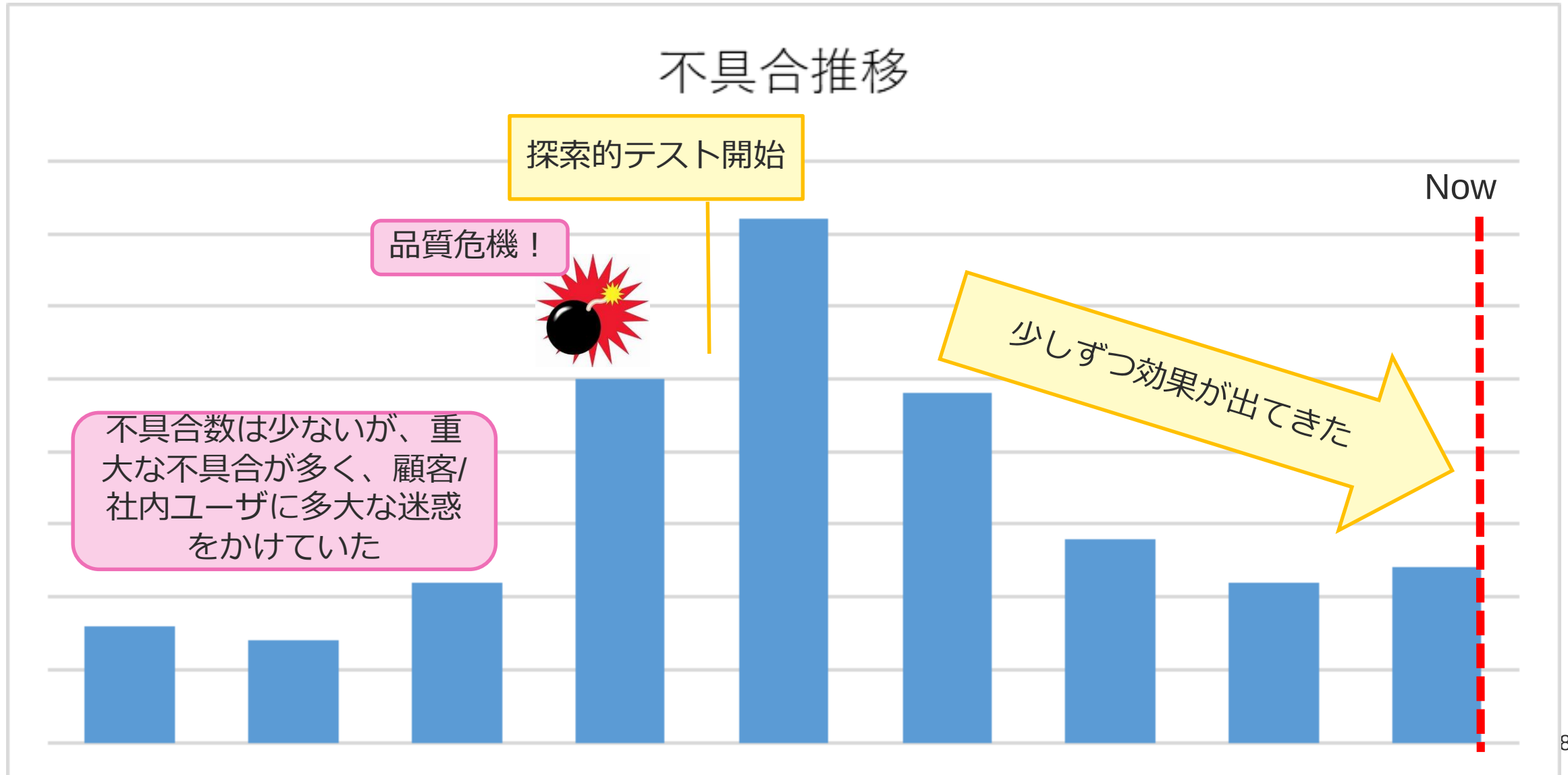
テスト概要

フェーズ	テスト種	目的	実施者
評価	機能試験	機能が仕様通り動作することを機能試験仕様書にそって実施する	開発者
出荷試験	定型試験	基本機能の動作確認、回帰試験、など	開発者or SQA
	探索的テスト	弱いエリア、リスクの高い機能、異常系を集中的にテストする	開発者and SQA
	QA第3者試験	QA観点で抽出した機能テスト（異常ケース、他機能への影響、装置運用観点）	SQA

実施の効果

実施の効果

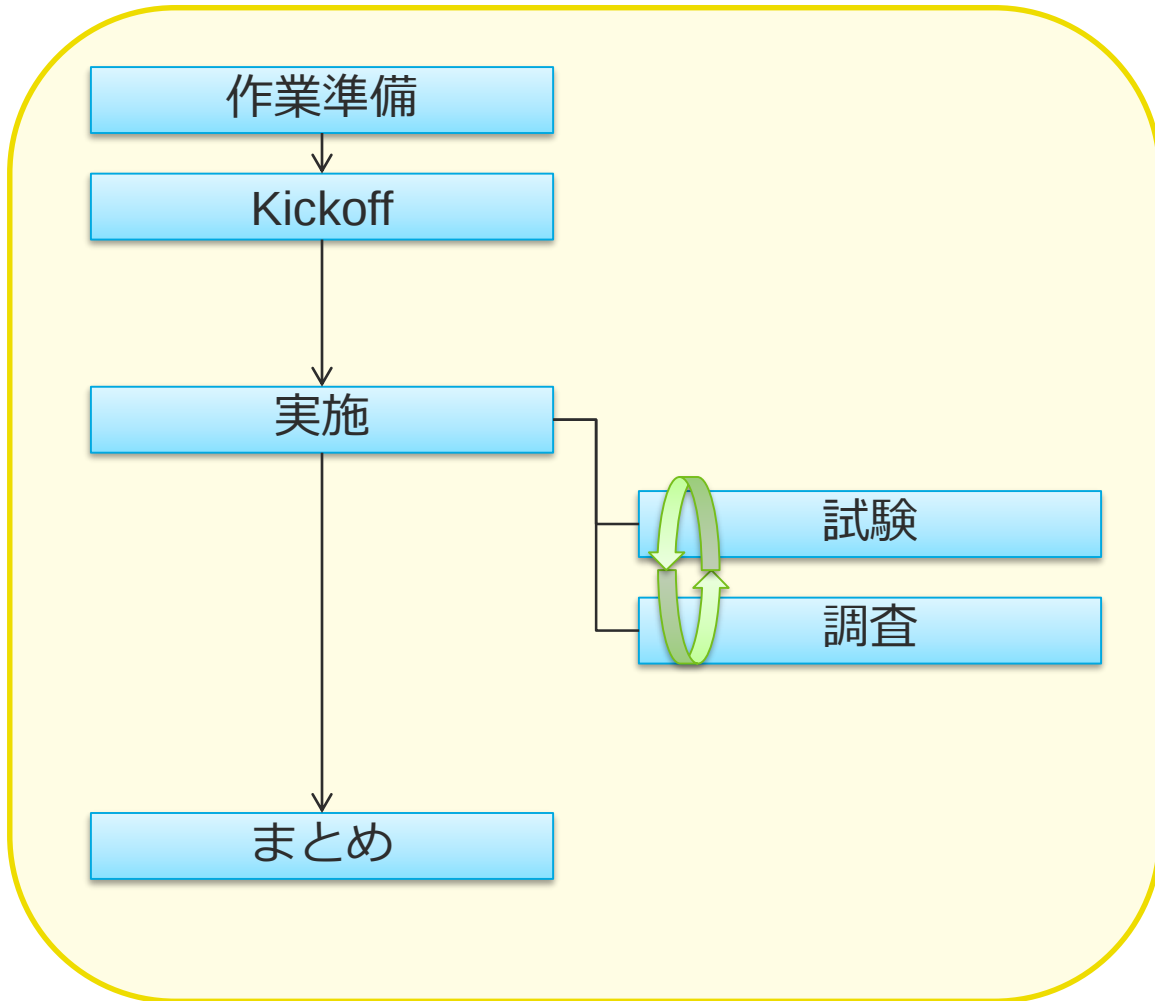
* 詳細情報割愛



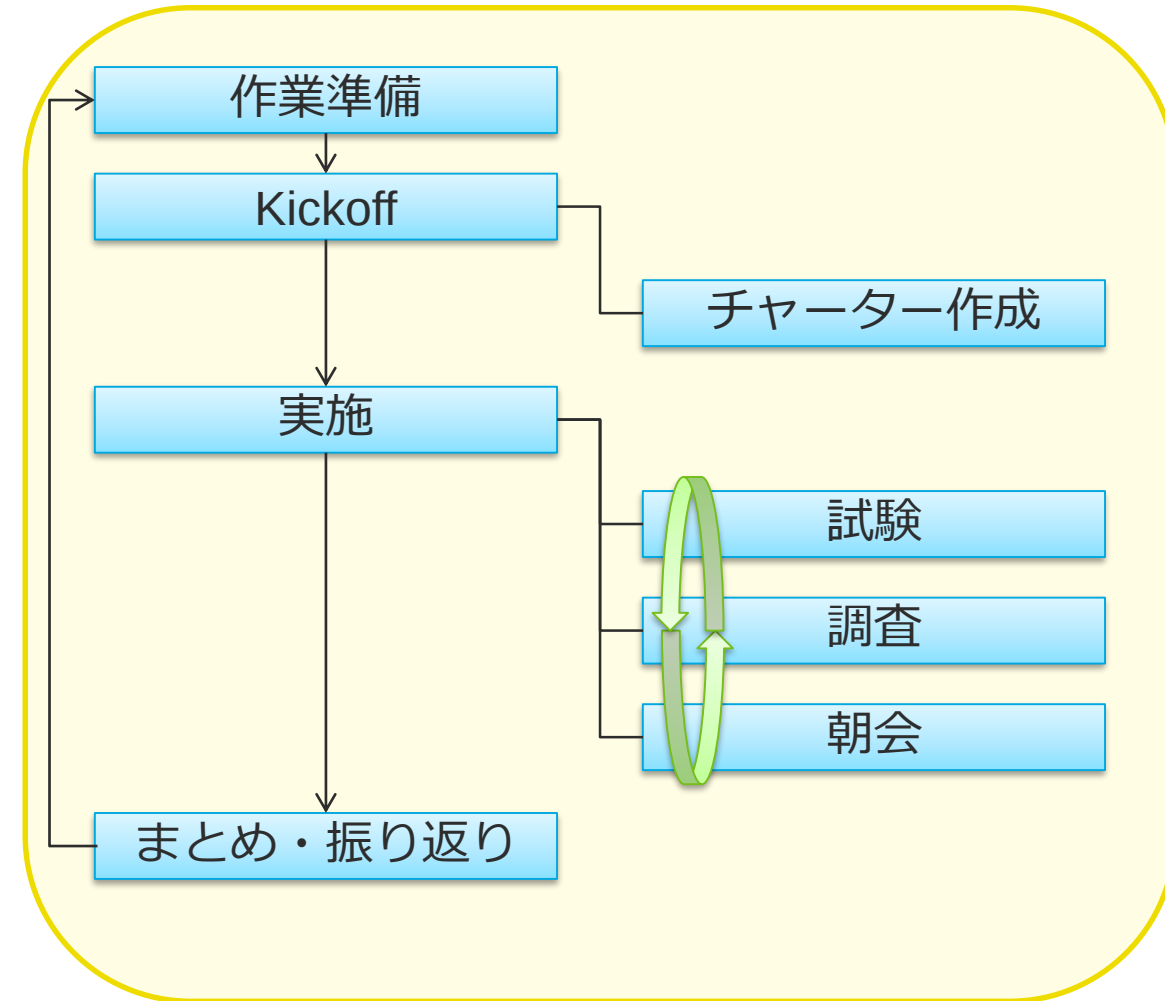
探索的テスト実施フロー

探索的テスト実施フロー（導入当初と現在）

導入当初

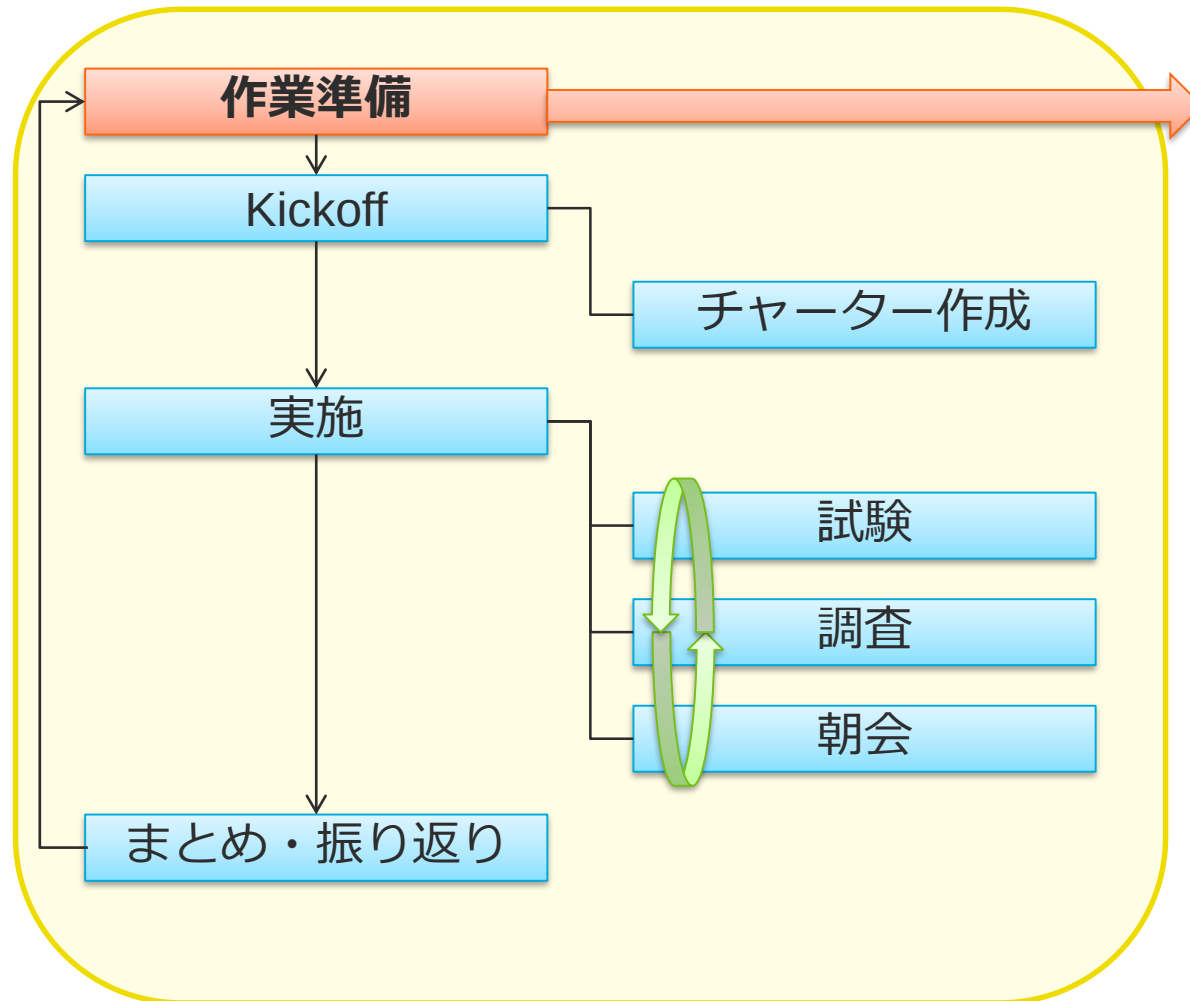


改善後（現在）



導入当初は他グループのやり方をそのまま模倣。その後、改善を重ねて、現在のフローに。

探索的テスト実施フロー（詳細）

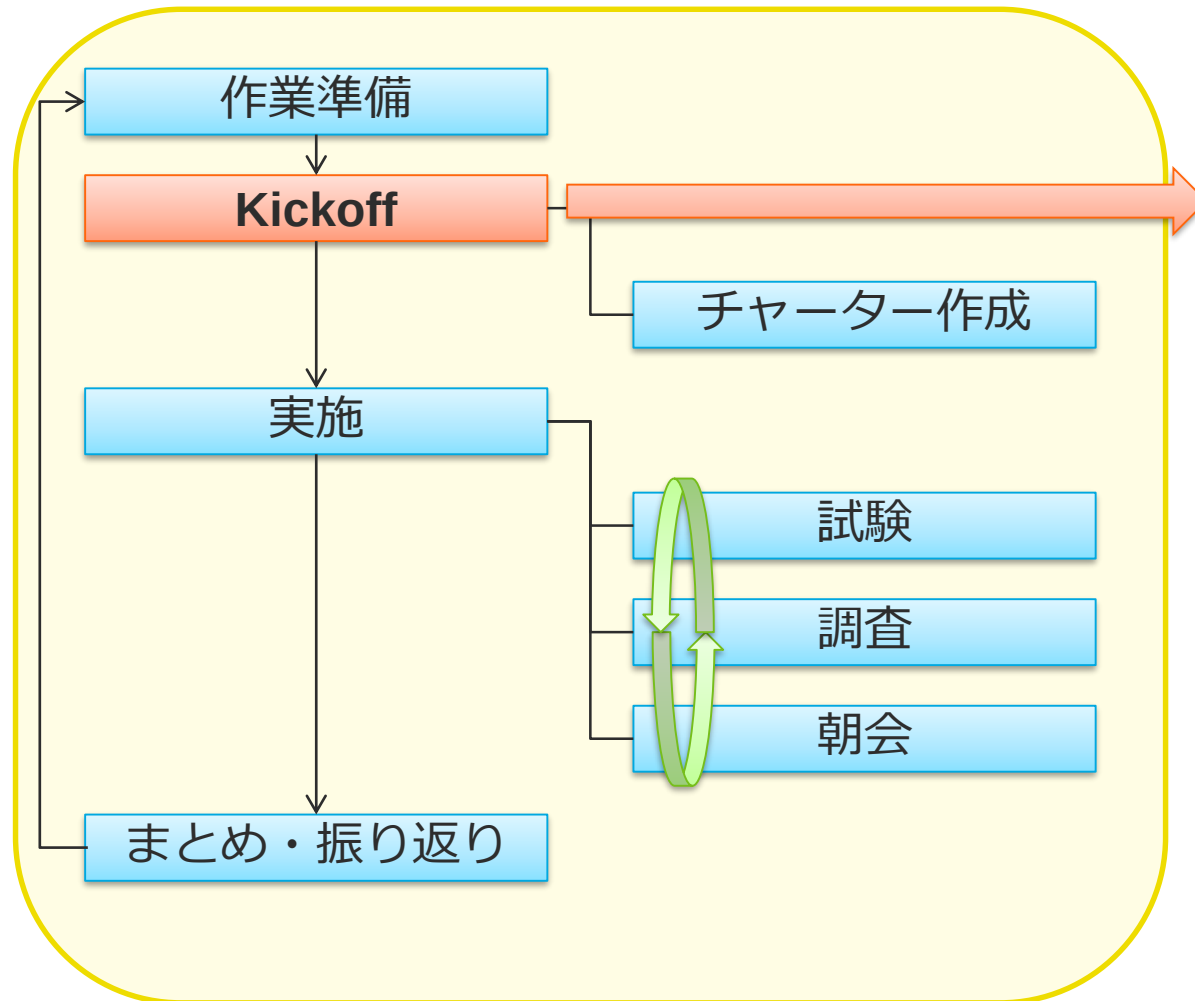


【作業準備】

- 試験計画（案）の作成
 - ・ 対象バージョン、テスト期間
- 各種ファイルの保存場所作成
 - ・ ログ
 - ・ チャーター
- 試験環境の作成
 - ・ VirtualBox★
- 不具合チケット起票の準備
 - ・ Redmine

SQAがテストの段取りを実施することで、テスターは迷わずテストに入れる。

探索的テスト実施フロー（詳細）

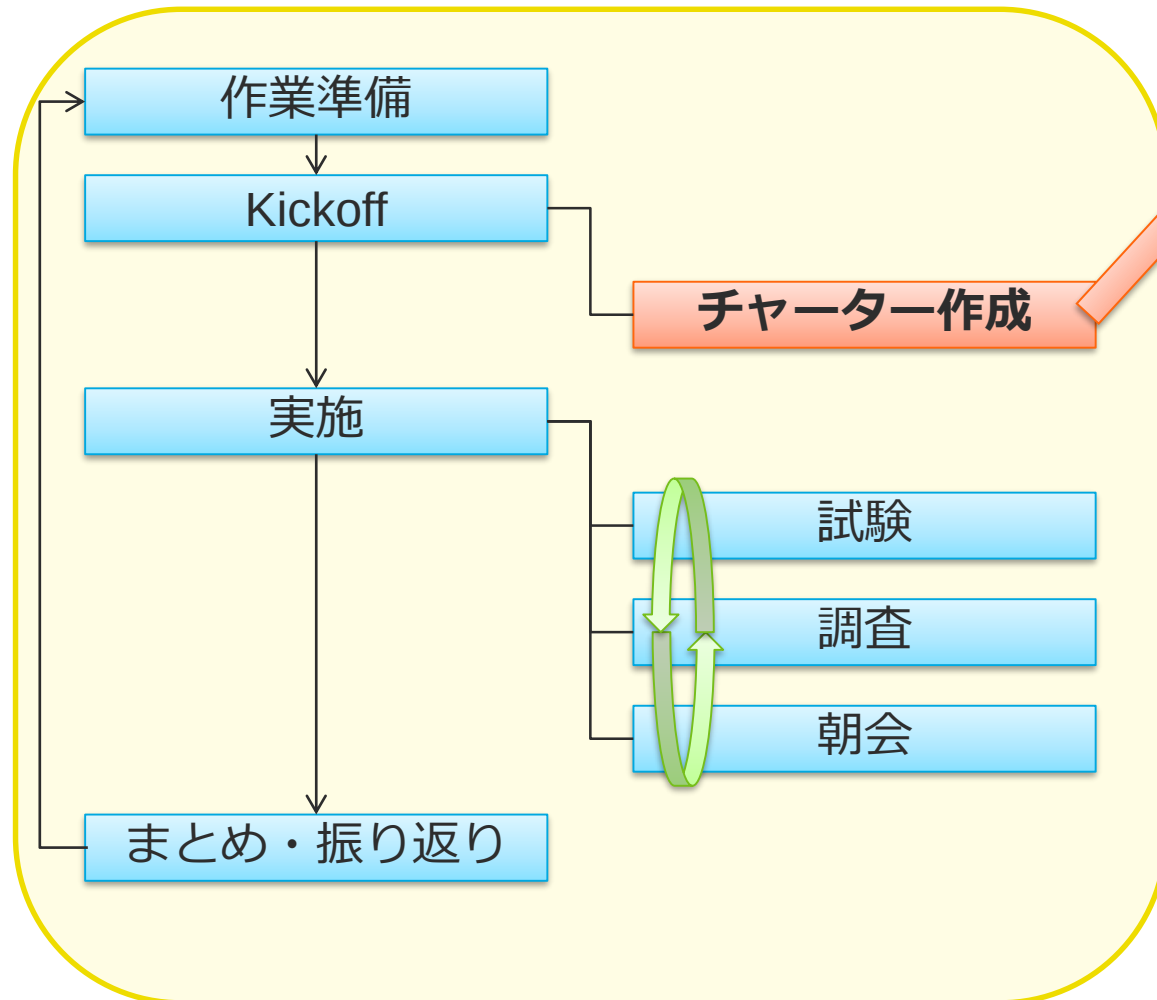


【Kickoff】

- 試験計画の共有
 - ・ 試験期間/対象バージョン/試験環境/ログの保存場所
- 狙いどころを決める
 - ・ 対象の機能/案件（開発者、マネージャー、SQA、全員の意見を聞く）
- 作業分担
 - ・ 誰がどの機能をテストするか
 - ・ メンター（機能開発者）はだれか
- 宿題を出す
 - ・ チャーターの作成

KickOffで関係者全員のベクトルを合わせる。

探索的テスト実施フロー（詳細）



【チャーター作成】★

テスト担当者はメンター（機能開発者）と話ながら、チャーターを作成

*チャーターとは、テストの指針や方針を決めるもの（道しるべ）であり、書き方/粒度に決まりは無い。

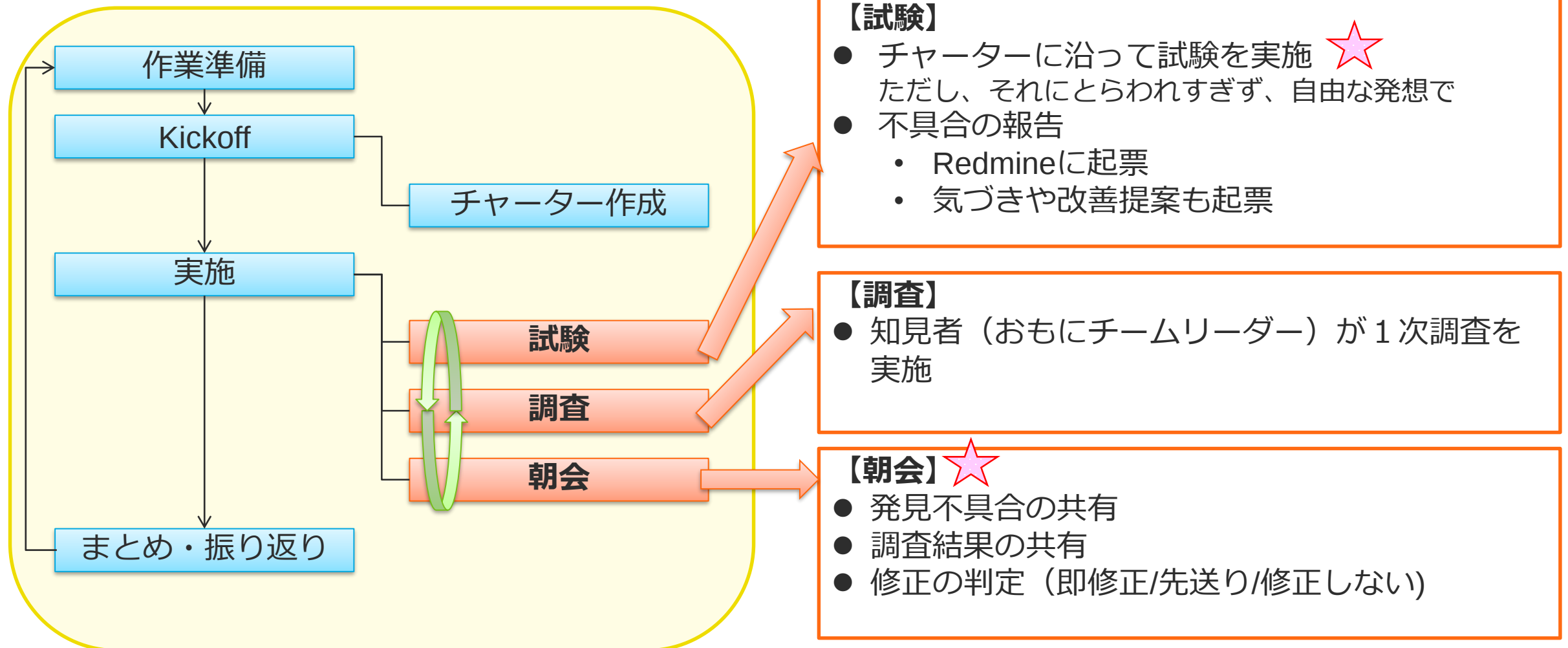
機能のポイントからのテスト

- 1) 設定パラメータは動的に変更可能
⇒ パラメータをいろいろなタイミングで変えてみる
- 2) ダミー/製品間で処理排他が可能
⇒ コンディショニングを絡めてみる
- 3) ウェーハの搬出待ち時間は先行ウェーハの滞留に応じて動的変化
⇒ 搬入先の状態をいろいろ変えてみる
⇒ 搬送に関わるパラメータを変えてみる

過去不具合の再テスト

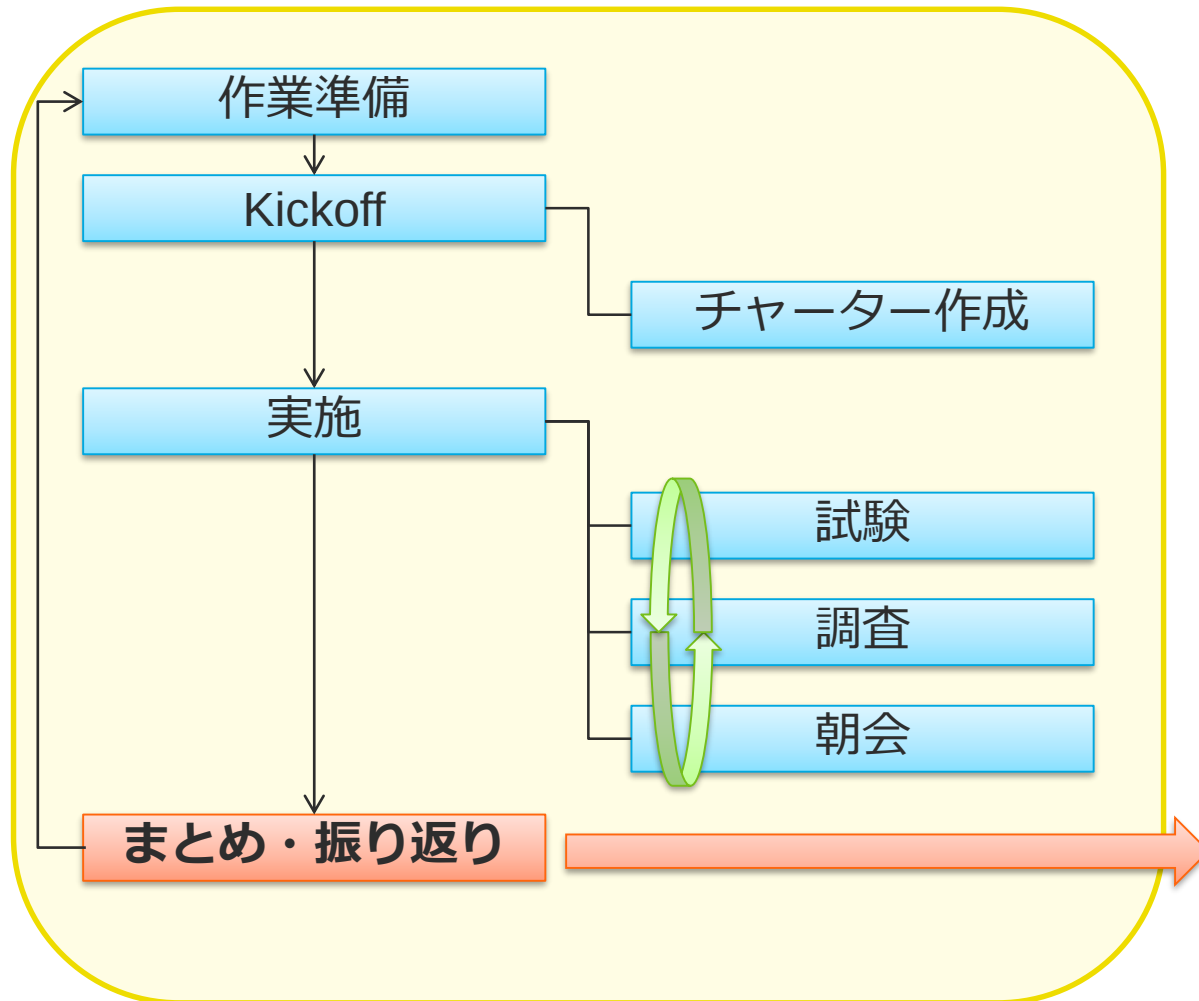
- ・ウェーハを残した状態で再起動
- ・起動後エクストラウェーハ出現
- ・ウェーハ削除

探索的テスト実施フロー（詳細）



Dailyの朝会実施は、情報共有以外にも様々な効果あり。詳細は後ほど

探索的テスト実施フロー（詳細）



【まとめ】

- 品質指標データまとめ
 - ・ 報告件数、不具合件数、リリースまでに修正する不具合件数など
 - ・ テスト工数

【振り返り】★

- 意見収集
 - ・ 試験対象（ソフトウェア）の質、探索的テストのプロセスのKPTを関係者から事前に収集
- KPT = Keep / Problem / Try
- 振り返り
 - ・ 収集した意見をもとに実施
 - ・ 次回へのアクション決め
 - ・ グループメンバー全員と共有

テストの振り返りを実施することで、改善につなげる。詳細は後ほど

工夫しているポイント

工夫している点 【チャーター作成】

- メンター（機能開発者）とテスター（テスト担当者）でチャーター作成

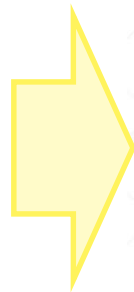
効果！

メンター

テスター



チャーター



- ・ テスターは開発者の中から機能開発者とは**別の人をアサイン**する。
- ・ 機能開発者がテストしてほしいところをテスターに**アドバイス**する（自信の無いところ、心配なところ、追加テストしてほしいところ、、、etc）

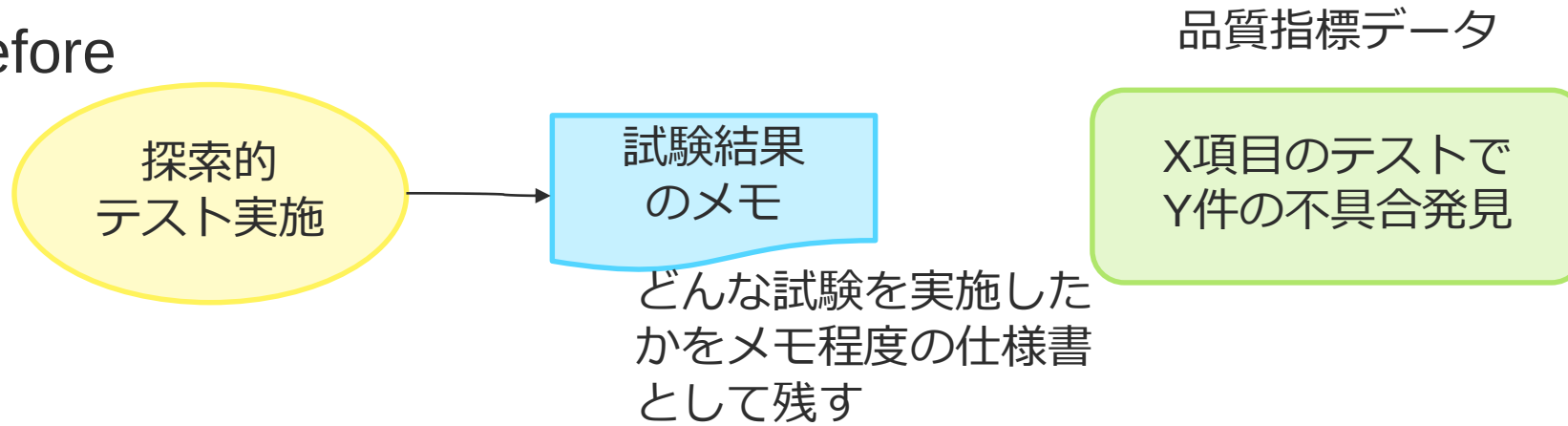
- ・ 開発者は、自分の作成した機能は優しくテストしてしまう。第3者が**厳しくテスト**して不具合流出阻止！
- ・ テスターは、**自分の知見もかけあわせて**様々な視点でテスト実施！
- ・ テスターは、機能開発者のアドバイスでテスト範囲を**絞り込むこと**で効率的に不具合発見！



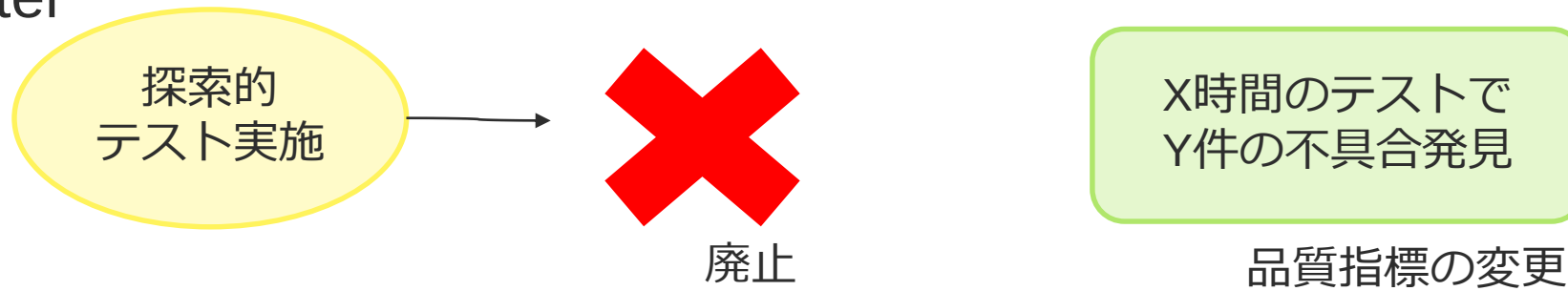
工夫している点 【試験の実施】

■ 仕様書（試験結果メモ）の廃止

Before



After



効果！

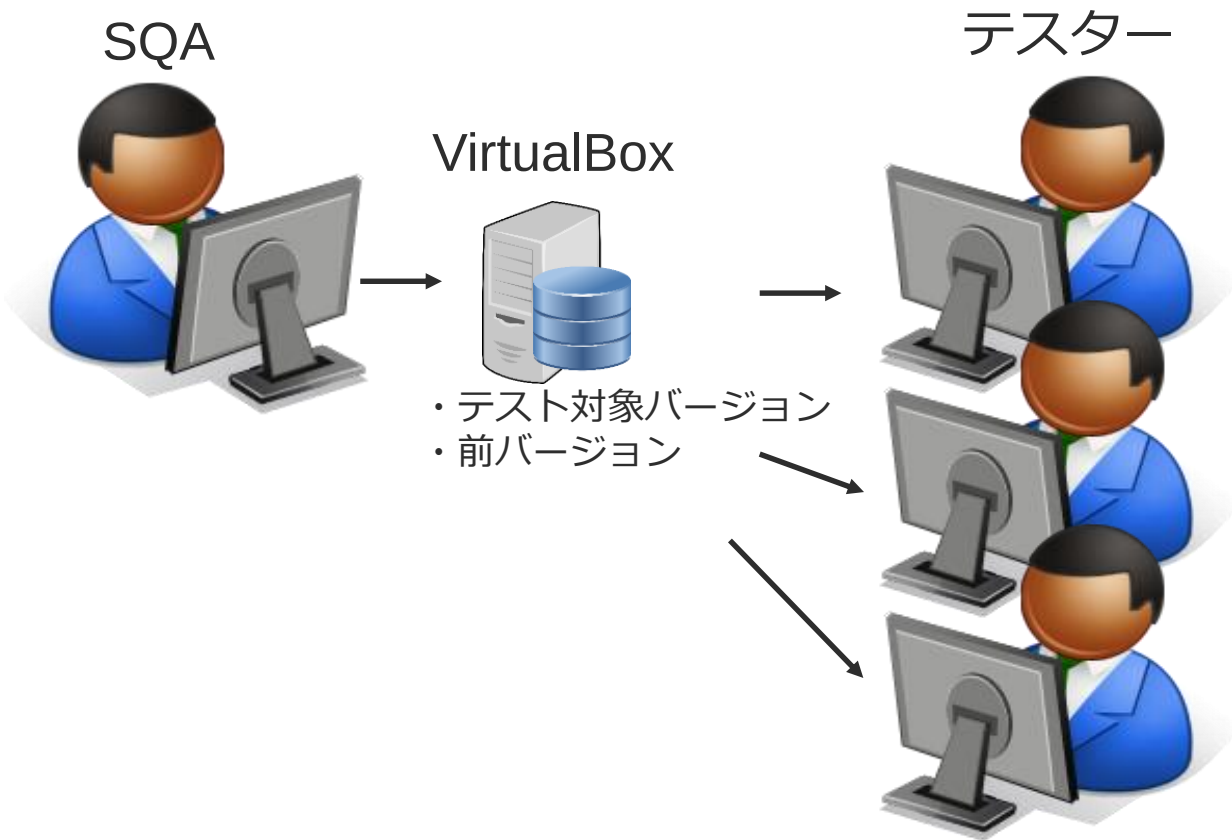
- ・ **試験に集中**可能
(流用されない仕様書作成の時間を廃棄)

SOZAI GOOD



工夫している点 【試験環境】

- VirtualBoxで試験環境を提供
- 前バージョンの環境も同時に提供



効果！

- ・ テスターと不具合調査する人が同じ環境を使用することで、**調査工数削減**
- ・ **バージョンアップ前後での動作確認**が可能となり、バージョンアップによる挙動変更がわかる
- ・ 不具合作りこみがテスト対象バージョンなのか**潜在不具合なのか素早く判断**可能

SOZAI GOOD



工夫している点 【朝会の実施】

■ Dailyの朝会実施

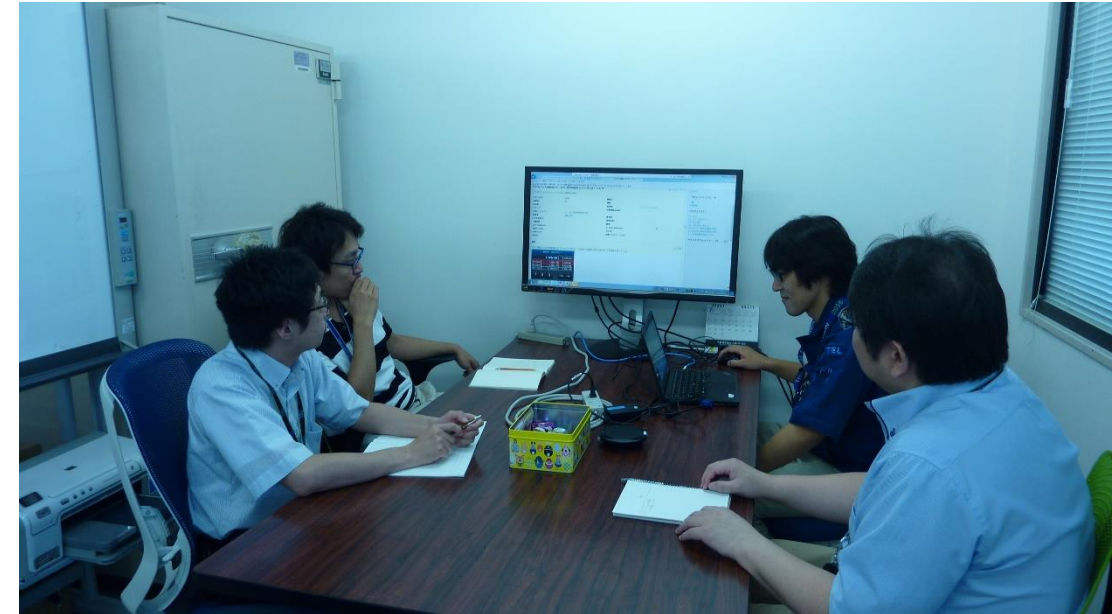
- 前日あがった不具合（チケット）の内容確認と切り分け
- 不具合発見時は「イイネ！」と褒める

- ・ **アクションを即判断**（リリース前に修正／次バージョン以降で対応／不具合ではない）
- ・ 他人の不具合報告を聞いて、**自身のテストの参考**にできる
- ・ テスター間の**競争意識**が芽生える

SOZAI GOOD



↓こんな感じでやってます



効果！

工夫している点 【振り返りの実施】

■ 探索的テスト終了後に振り返り実施

Input

振り返り

Output

効果！

テストプロセス
KPT

事前収集

プロダクト
KPT



次回の
アクション

定量的/定性的
品質測定

- ・ 次回のテストの**効率UP**
- ・ 発見不具合数による**品質レベルの測定**



プロセス⇒

プロダクト⇒

その他⇒

	Keep↓	Problem↓	Try↓
テストプロセス	良かった		

まとめ

まとめ

- メンバー全員が気持ちよく作業出来る環境（雰囲気）作りが大事
 - テストに関わる全員でのベクトル合わせ
 - メンターとテスターの共同作業を通してのチームワーク
 - 不具合報告には“イイネ！”と褒める文化
 - メンバーの意見をもとに改善していく組織風潮
- 探索的テストを継続させる意思が大事
 - SQA、マネージャーの導入から継続につなげる意思
 - 目的と効果を全員で共有することで、全員が必要性を認識
 - 改善を続けることが継続の近道

ご清聴ありがとうございました。



TOKYO ELECTRON