納得するシステムテストをするための原則、実践アンチパターン

JaSST Tokai2023　10 Feb. 2023 10:45- 12:15

株式会社 ytte Lab 湯本 剛 (twitter @yttelab)

© Copyright 2019-2023 ytte Lab Ltd. The information contained herein is subject to change without notice.

自己紹介　湯本剛

❑経歴

1991年　工作機器メーカーで経理部に所属し、生産管理システムの受入テストに従事
1995年　メーカー系列ソフトハウスにてテストリーダーとしてアプリ開発に従事
- 財務会計のパッケージソフト、プリンタ、スキャナ、製造業受託開発（在庫管理、生産管理など）、顧客向けプロダクト販売Webサイト構築など

2003年　ソフトウェアテストコンサルタントとなりテストプロセス改善コンサル、テストツール導入支援、テスト教育に従事
2010年　日本ヒューレット・パッカード(HP)　ソフトウェア部門にて、ALM製品のプリセールスに従事
2014年　日本HPのサービス部門にて、デリバリー案件のテストマネジメント、QA組織立ち上げのコンサルテイング&デリバリー
2017年　AIG損保　ソリューションデリバリー部テスティング課　課長

❑現在（2023年2月）

freee株式会社 QAマネージャー　／（株）ytte Lab 代表 & コンサルタント
NPO法人　ASTER：ソフトウェアテスト技術振興協会　理事
JSTQB(認定ソフトウェアテスト技術者資格制度)技術委員
テスト設計コンテスト審査員
ISO/IEC JTC1/SC7 WG26　幹事（ISO29119：テストプロセス標準の策定）

❑著書

- 書籍（6冊）：　ビジネス主導のテストプロセス改善、現場の仕事がバリバリ進むテスト手法、ソフトウェアテストの基礎、JSTQBソフトウェアテスト教科書、基本から学ぶソフトウェアテスト、ソフトウェアテスト293の鉄則
- 学術論文：「データ共有タスク間の順序組合せテストケース抽出手法」電気学会　論文誌C　Vol. 137　No. 7
- 国際学会：　A Test Analysis Method for Black Box Testing Using AUT and Fault Knowledge　など多数
- 雑誌、ネット記事：　日経ITProなど多数

© Copyright 2019-2023 ytte Lab Ltd. The information contained herein is subject to change without notice.

アジェンダ

- ✓ ソフトウェアテストの原則
- ✓ システムテストの位置付け
- ✓ システムテストの基本
- ✓ システムテストの実践
- ✓ システムテストのアンチパターン

© Copyright 2019-2023 ytte Lab Ltd. The information contained herein is subject to change without notice.

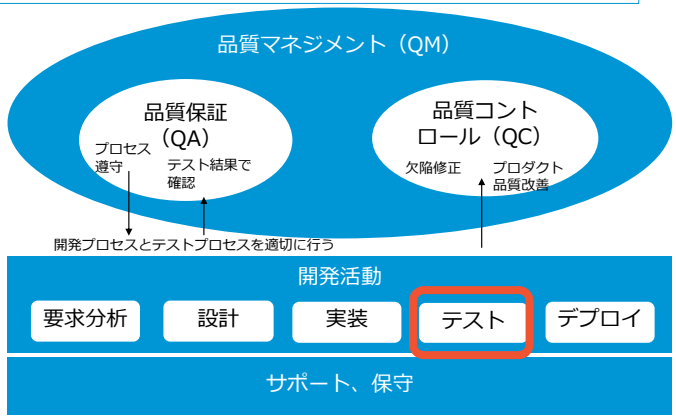
ソフトウェアテストの原則

ソフトウェアテストの大きな二つの役割
上手いテストをするとは？
ソフトウェアテスト7つの原則

© Copyright 2019-2023 ytte Lab Ltd. The information contained herein is subject to change without notice.

ソフトウェアテストには2つの大きな役割がある

品質保証
- 成果物が定めたレベルの出来栄えに到達するようプロセスを作り、それを遵守
  - 設計時の意識合わせ、セルフレビュー、仕組み化、自動チェックなど
- プロセス品質とプロダクト品質を確認
  - テスト結果は品質保証にとっての（とても鮮度の良い）基礎データ

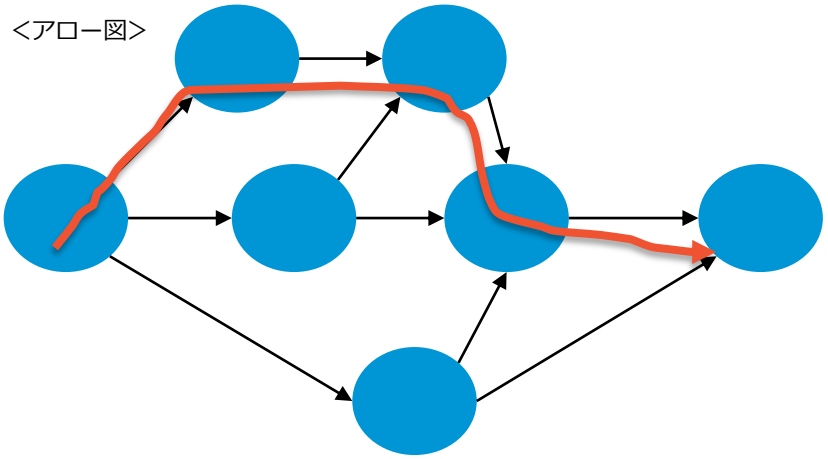


品質の良いものを世の中に送り出すための大事な手段

・QAのインプット情報として、鮮度がよく綺麗な情報を渡すこと

どれだけ正しく写す鏡か？

<アロー図>



開発ライフサイクルの主要な活動

・技術業として、決められたQCDを達成できるように仕事をすること

どれだけ効率的にできるか？

© Copyright 2019-2023 ytte Lab Ltd. The information contained herein is subject to change without notice.

上手いテストをするとは

テストの活動を**クリティカルパス**にしないこと

テストは品質マネジメントの一手段

- 他の品質マネジメント手段との住み分けをしてスコープを明らかにする
- 関係する人たちへの「迅速に」「正しい」情報提供

テスト技術次第

- テスト対象の確認をより少ない量で確実に行う技術（テスト設計、テスト戦略）
- 作業スピードを上げる技術（仮想化技術、自動化技術）
- テストでのアウトプットを再利用する技術（上記技術の適用、構成管理、メンテナンス）

開発ライフサイクル次第

- テストの成熟度を高めようとするほど、開発の状況が前提条件となる

© Copyright 2019-2023 ytte Lab Ltd. The information contained herein is subject to change without notice.

ソフトウェアテスト7つの原則 from JSTQB FLシラバス

いつの時代でも変わらないこと

テストは欠陥があることは示せるが、欠陥がないことは示せない

全数テストは不可能

「バグゼロ」の落とし穴

早期テストで時間とコストを節約

欠陥の偏在

殺虫剤のパラドックスにご用心

テストは状況（context）次第

「ちゃんと網羅して」
って言われたら、
言い返せるようになろう！

どんな開発でも絶対的に有効

満遍なく平均的に欠陥が潜んでる
ことはほぼ確実にない

そうなんだけど、、だからこそ
上手くやってほしいのがこれら

次ページでもうちょっと深掘り

© Copyright 2019-2023 ytte Lab Ltd. The information contained herein is subject to change without notice.

テストは状況次第：開発トレンドとテストへの影響

変わることと変わらないことをわきまえる

オープンソース

- **開発ベンダーに依存しない（コントロール可能）**
- **いろいろなものを組み合わせる**

- 開発に合わせたテストツール
- 組み合わせの変更をテスト
- 自分で責任が取れるようテスト

クラウド

- **インフラ調達が瞬時にできる（コントロール可能）**
- **運用と開発が分離しないのでリリーススピードUp**

- 本番環境でのテスト
- 機能トグルを使ったテスト
- 性能や信頼性をモニタリング

アジャイル開発

- **トライアル＆エラーな開発で少しずつ作る**
- **微調整を繰り返すことができるようにする**

- 意識合わせしながらテスト
- リグレッションテストが主役
- メトリクスが変わる

マイクロサービスアーキテクチャー

- **小さく頻繁なリリースをしても他に影響を与えない**
- **重複を許可して結合点を最小限にする**

- ユニット→サービス→E2E
- 結合点の保証方法をテスト
- 適応度関数で保証する

© Copyright 2019-2023 ytte Lab Ltd. The information contained herein is subject to change without notice.

開発のトレンドに合わせてテストをどう変えるか？

開発とQAが別チームというアプローチ	**QAがチームに内在するアプローチ**（チーム自身が品質に対する判断をする）
Traditionalな品質リスクの許容度合い	**いまどきの品質リスク許容度合い**（すぐに修正／リリースできる欠陥は許容する）
ビッグバンテスト（リリース前に大量にテストをする）	**予防テスト**（リリース前に欠陥修正による手戻りがないようにテストをする）
品質保証的判断（リリース可能な品質にあることを様々な証拠をもとに第三者が判断する）	**継続的品質改善**（リリースとは関係なく常にチーム全員が品質をよくしていく）

© Copyright 2019-2023 ytte Lab Ltd. The information contained herein is subject to change without notice.

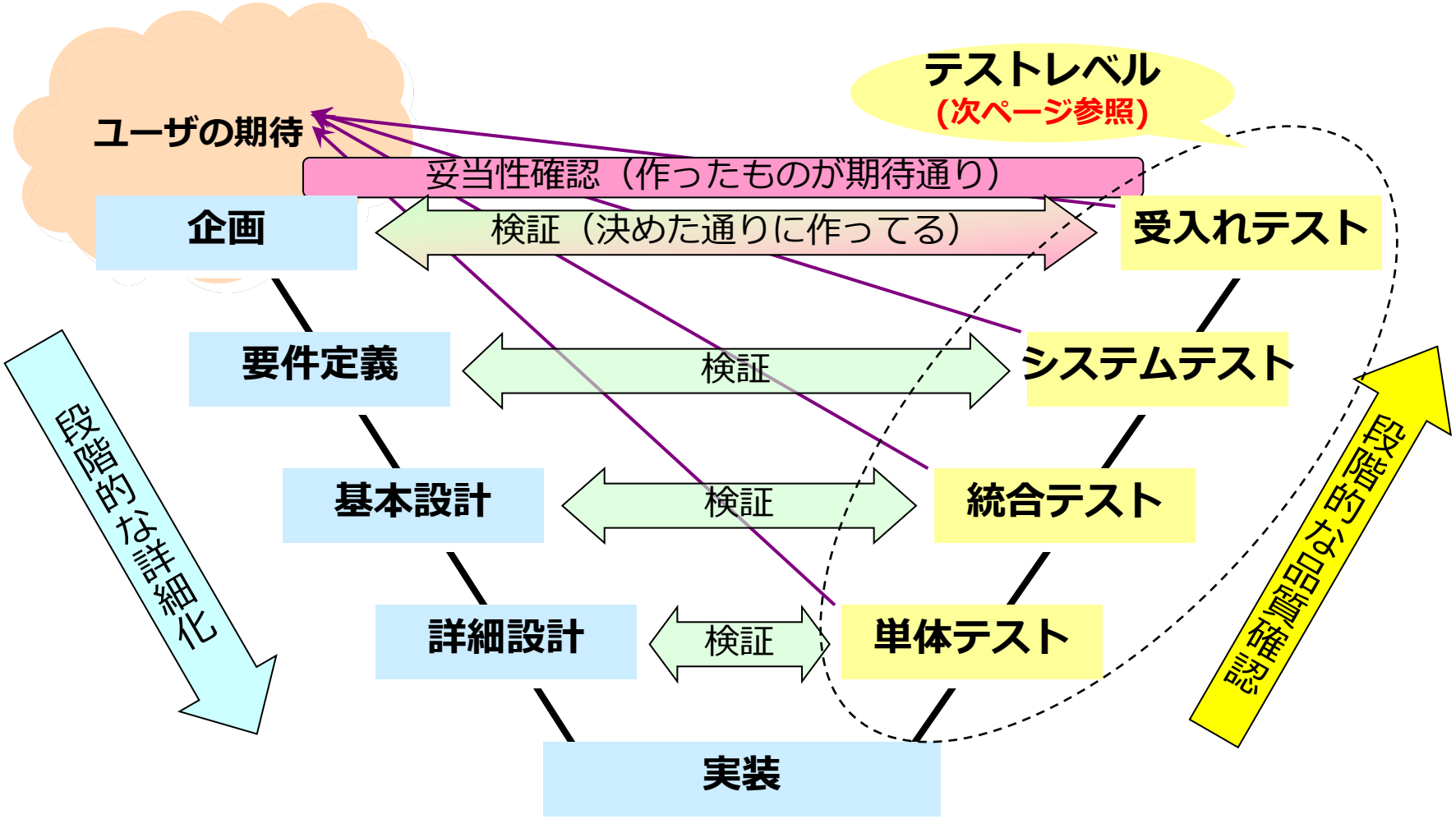
システムテストの位置付け

V字モデル
テストレベル
要求提示者の成果物＝システムテストのテストベース

© Copyright 2019-2023 ytte Lab Ltd. The information contained herein is subject to change without notice.

V字モデル（典型的）

各開発工程とテストレベルの関係を表したモデル



© Copyright 2019-2023 ytte Lab Ltd. The information contained herein is subject to change without notice.

典型的なテストレベル

単体テスト

開発者が作成する最小単位(モジュールやクラス)を対象としたテスト

統合テスト (結合テストともいわれる)

単体テストしたものを統合したものを対象にしたテストであり、インターフェース確認が主眼

システムテスト

システム全体を対象としたテストであり、指定された要件が満たされているか否かの確認が主眼

受け入れテスト

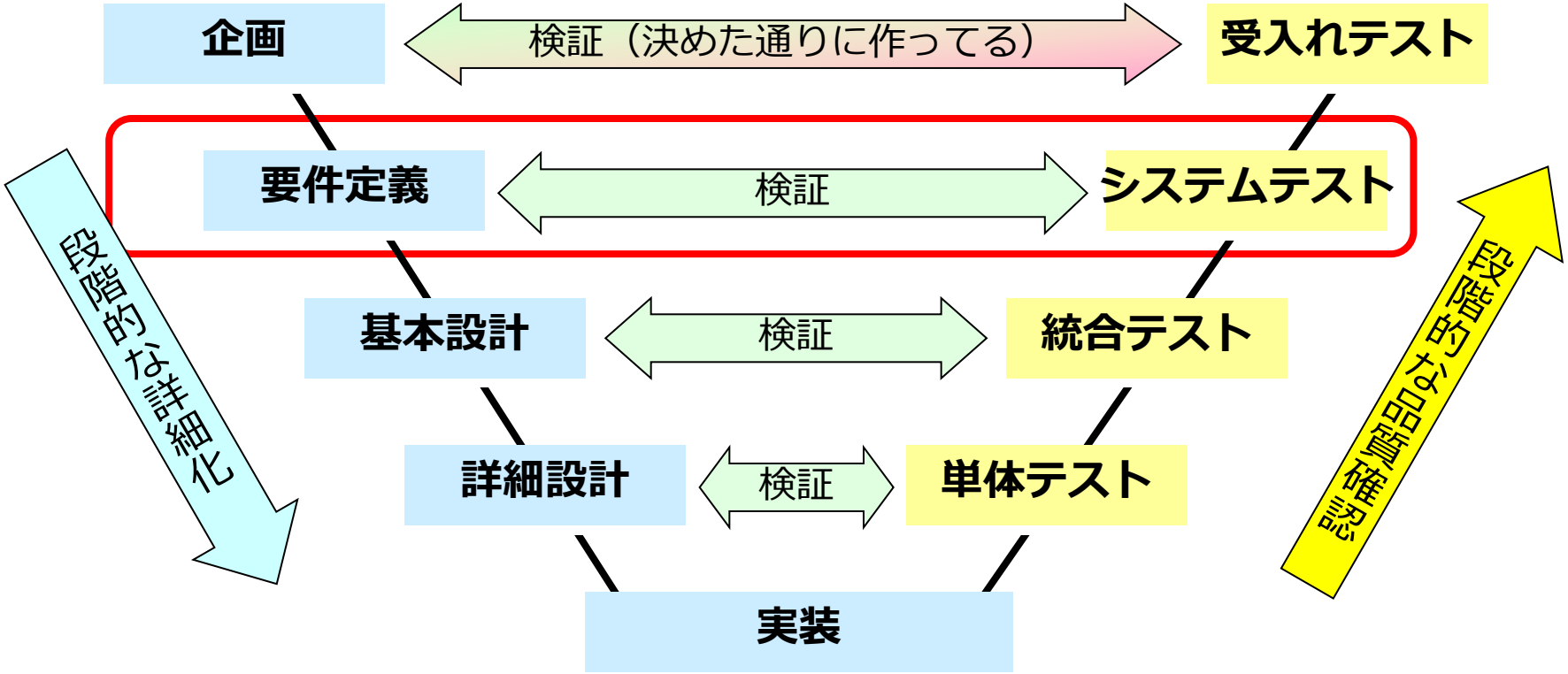
顧客がソフトウェアを受け入れるか否かを判定するためのテスト

© Copyright 2019-2023 ytte Lab Ltd. The information contained herein is subject to change without notice.

システムテストの位置付け

JSTQB FLシラバスに書いてあるシステムテストの定義

- システムが実行するエンドツーエンドのタスクと、タスクの実行時にシステムが示す非機能的振る舞いといったシステムやプロダクト全体の振る舞いや能力に焦点をあてる

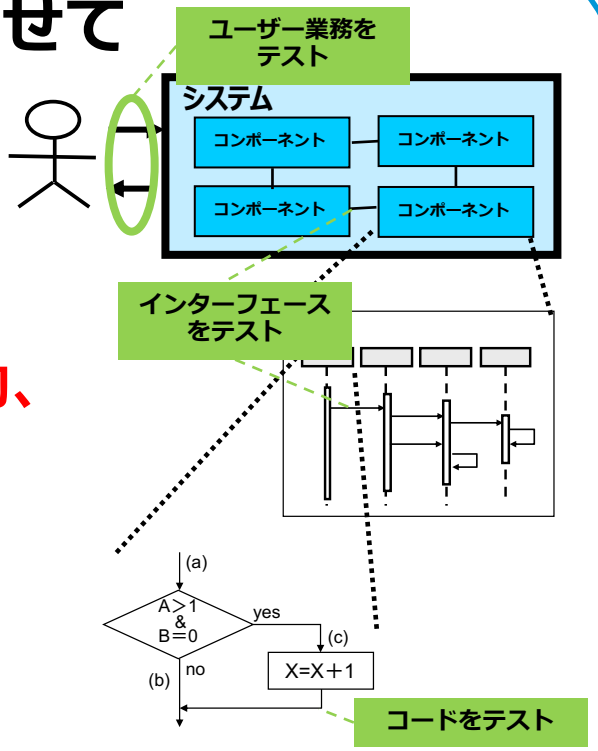


© Copyright 2019-2023 ytte Lab Ltd. The information contained herein is subject to change without notice.

テストレベルごとにテストするのはとても重要

テストは、テスト対象の詳細レベルに合わせて行うことが必要となる

・コードをテストする
・インターフェースをテストする
・ユーザー業務が遂行できるかをテストする…etc.



テストレベルを考慮することでテスト対象を効果的、効率的にテストすることができる

・網羅するための母数を混ぜると危険
- ユーザー業務が遂行できたかをテストする際に、コードのパスを母数にするとテスト量が膨大になる

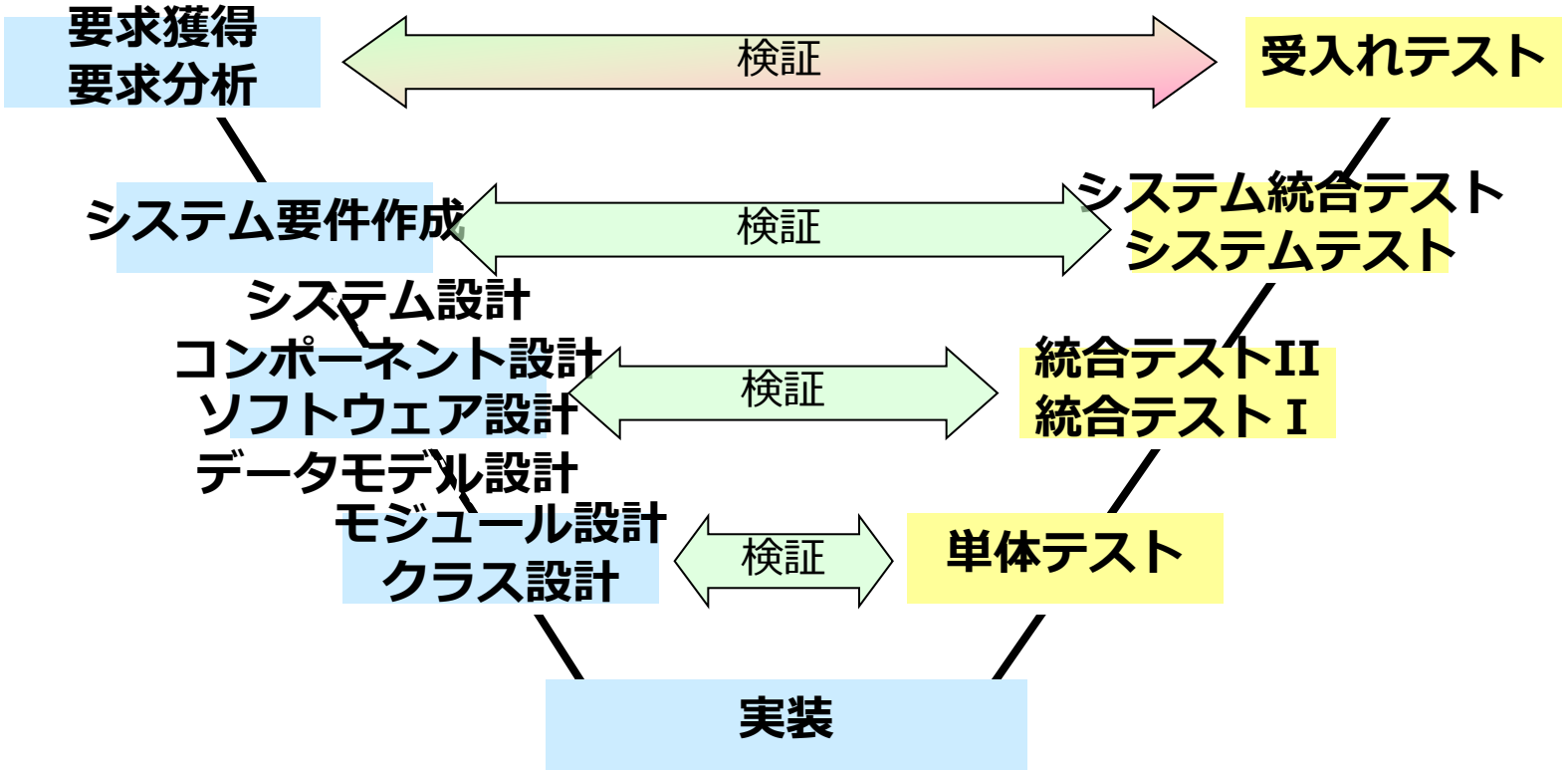
・テストの仕方が非効率になる
- 開発環境でコードのif文通過のテストでユーザー業務遂行可能か判断するのは難しい
- 本番相当の大規模なテスト環境で、多くの人が関与してユーザー業務が遂行できたかをテストする際に、コードのif文通過を確認するテストを行うのは難しい

note参考「フェーズ、レベル、タイプ、技法」

© Copyright 2019-2023 ytte Lab Ltd. The information contained herein is subject to change without notice.

テストレベルは、実際はもっと複雑

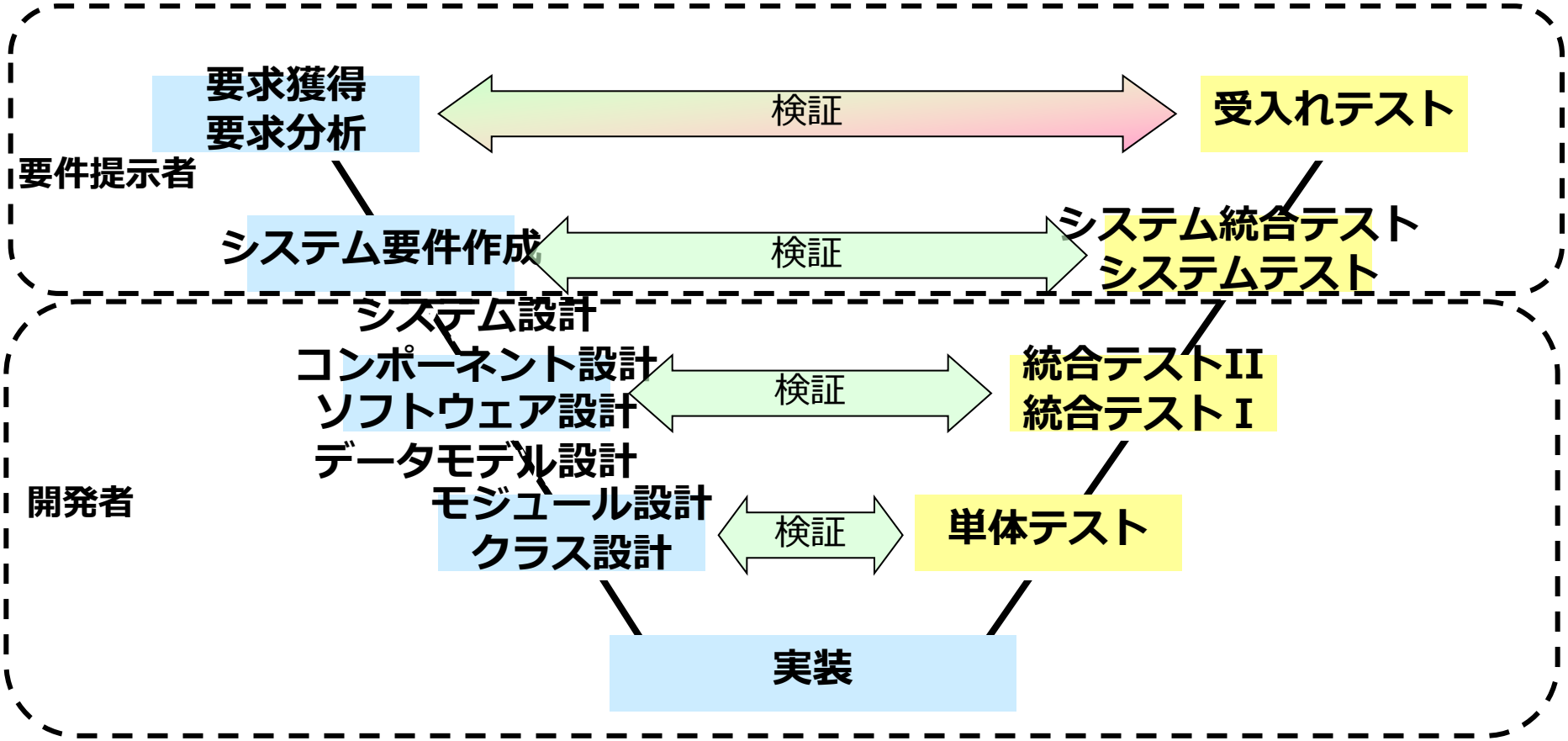
各開発工程とテストレベルの関係を表したモデルの具体例



© Copyright 2019-2023 ytte Lab Ltd. The information contained herein is subject to change without notice.

テストレベルは、実際はもっと複雑

各開発工程とテストレベルの関係を表したモデルに責任分解点追記



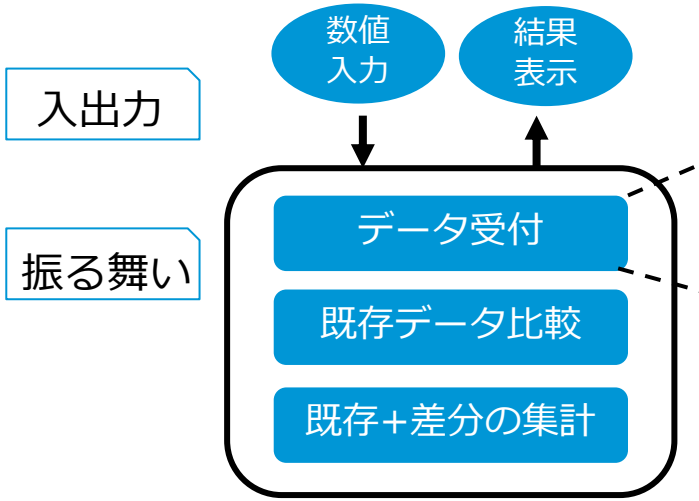
© Copyright 2019-2023 ytte Lab Ltd. The information contained herein is subject to change without notice.

要求提示者の成果物＝システムテストのテストベース

システムテストのテストベースの特徴

- システムの内部には言及しない
  - システムの入出力を具体的に決めたもの
  - システムの振る舞いを論理的に決めたもの
  - 利用シナリオ

- システムテストのテストベースのイメージ



統合テストのテストベースは…

システムテストのテストベースではないので、統合テストまででテストしてあることを確認すること

「論理的な振る舞い」を実現できていることをシステムテストで確認する

Point!

・P-Vの特定に統合テストのテストベースを使う場合はある（P-Vについては後述）

© Copyright 2019-2023 ytte Lab Ltd. The information contained herein is subject to change without notice.

システムテストの基本

システムテストの概要
システムテストにおけるテストアプローチの3原則
システムテストにおけるテスト設計の4原則

© Copyright 2019-2023 ytte Lab Ltd. The information contained herein is subject to change without notice.

システムテストの概要 1/2

システムテストの目的 from JSTQB FLシラバス

リスクの軽減

- 該当のテストレベルに相当するリスクの軽減

システムの機能的/非機能的振る舞いが設計および仕様通りであることの検証

システムが完成し、期待通りに動作することの妥当性確認

システムの全体的な品質に対する信頼の積み上げ

欠陥の検出

- 該当のテストレベルに相当する欠陥の軽減

欠陥がより高いテストレベルまたは運用環境まで見逃されることの防止

Point!
自分が入るプロジェクトで、ここに書かれた全てのシステムテストの目的を満たさなくても良いので、プランで取捨選択すること

© Copyright 2019-2023 ytte Lab Ltd. The information contained herein is subject to change without notice.

システムテストの概要 2/2

テストアイテム

- ユーザーが使う時と同等に使える状態のシステム

テスト環境

- できる限りユーザーが使う時と同等にした環境

テストベース

- システムの外部からの入力操作と外部への出力結果の仕様（機能仕様）
- 論理的なシステムの振る舞い
- 利用シナリオに基づいた操作

Point!
テストベースは必ずしも仕様書である必要はないが、上記3つの情報を何かしらの方法で入手してテストケースを作っていく必要がある

© Copyright 2019-2023 ytte Lab Ltd. The information contained herein is subject to change without notice.

システムテストにおけるテストアプローチの3原則

1. 外部からの1システム操作を軸に操作の事前条件をP-Vにしてテストする

- 期待と違う結果の時に、調査しやすいようにテストを進めていく
  - 期待結果が独立して確認できる単純なP-Vを先にテストする
  - その後、現実的な値、複数のP-Vの組み合わせをテストする

2. ２つの操作をつなげてテストする（事前条件のP-Vに前の操作を入れる）

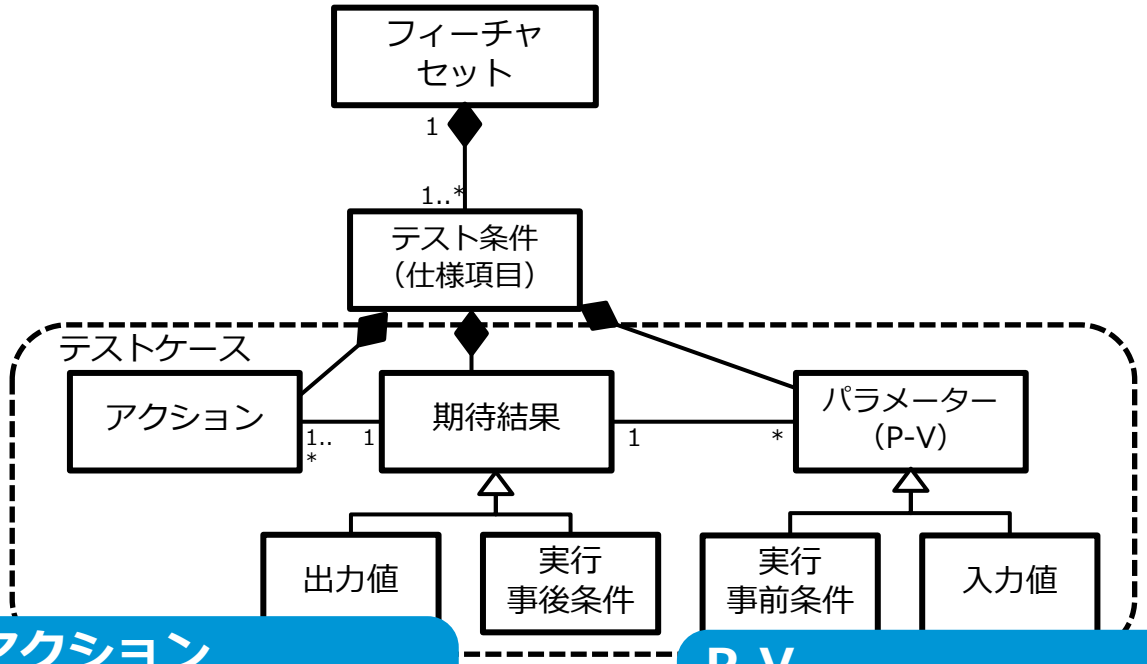
- 場合の数が爆発しないようにテストを進めていく
  - コンポジットの中だけテスト→外だけテストと段階的にテストする
  - 「無視する遷移＝機能テスト」「あり得ない遷移＝信頼性テスト」と目的で分ける
  - 履歴（ex.再起動後）→アトミック性（ex.割り込み）→時間（ex.タイムアウト）は分けてテストする

3. 利用シーンを想定して動作させても問題ないことをテストする

- 利用シナリオ単位でテストする
- 各利用シナリオを連結させた業務フローをテストする
  - シナリオを実現する際に必要となるP-Vに限定する
  - 業務フローシナリオをテストする際に、「1.」と「2.」のP-Vはできる限り混ぜない

Point!
一回のテストに対するP-Vをできる限り限定するため、だんだん組み合わせること

[参考]仕様項目、テストケース、P-Vについて



アクション
・ボリュームを設定
・音を再生

P-V
・ボリューム1の状態
・ボリューム2を入力

期待結果
・スピーカーから10dBの音が出力

仕様項目

例えば、音楽プレーヤーのサウンド設定にて以下のように定義したもの

「ボリュームは１から10の間で設定できる。１は消音であり、10は100デシベルであり、10デシベルずつ大きくできる」

© Copyright 2019-2023 ytte Lab Ltd. The information contained herein is subject to change without notice.

システムテストにおけるテスト設計の4原則

1. テストの目的、リスク、テストタイプから、仕様項目とP-Vを限定する

- 目的は何か？どんなリスクがあるか？どのテストタイプをテストするのか？をはっきりさせてテスト設計に臨まないといけない＝テストプランが大事

2. 仕様項目（テスト条件）は3種(後述)のテストベースから識別する

- テスト分析に使うテストベースをあらゆる手段で収集する

3. P-Vの組み合わせが爆発しないようにテスト設計する

- 仕様項目（テスト条件）の期待結果ごとにP-Vを組み合わせる
  - 非機能テストは非機能要件の期待結果を見るためのP-Vを組み合わせる
- 「1.」から考えて重要でないP-Vは任意にする
- 状態遷移は段階的にテストすることでP-Vを限定的にする
- 内部遷移、内部操作の期待結果がわかる他の外部出力を期待結果にして、P-Vを限定的にする

4. 想定外のP-Vを追加する

- 経験的に不具合が出そうなP-Vを追加する
- 利用シナリオに基づいたP-Vを追加する
- 想定外だったP-Vもできる限り業務フローシナリオのテストの前のテストに加える

© Copyright 2019-2023 ytte Lab Ltd. The information contained herein is subject to change without notice.

システムテストの実践

実践時に考慮してほしいTipsを３つ紹介

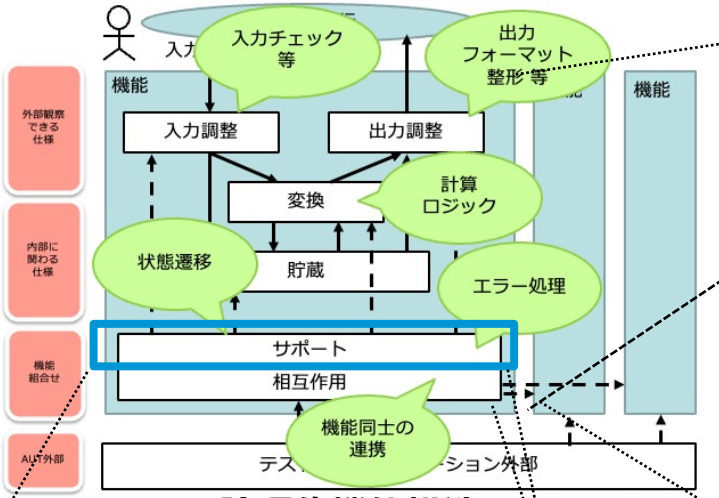
・テスト分析、テスト設計 : テスト対象を抽象的にイメージしながら作業する
・テスト実装：テスト環境の準備を徹底して行う
・テストマネジメント：関係者とコミュニケーションする

© Copyright 2019-2023 ytte Lab Ltd. The information contained herein is subject to change without notice.

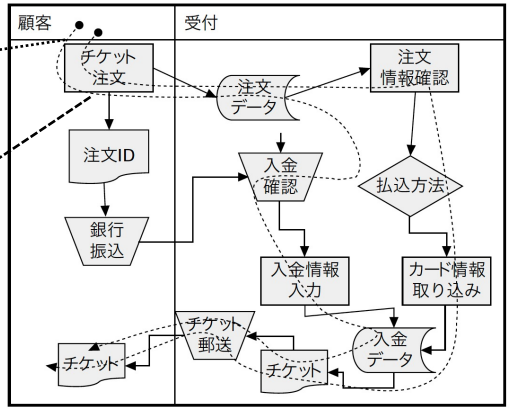
テスト分析、テスト設計（システムテスト）

テスト対象を抽象的にイメージする

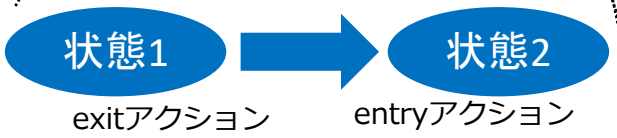
いきなり細かな理解に走らずに、３種類のモデルでテスト対象の全体像をイメージしながら、ざっくりでいいので内部的振る舞いまで理解する



論理的機能構造



業務フロー



アトミック性が必要なので割り込み禁止になる

論理的な状態遷移モデル

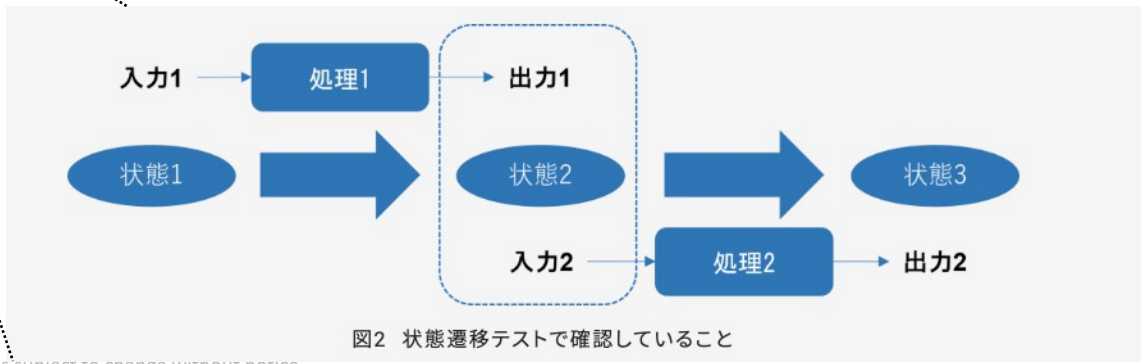
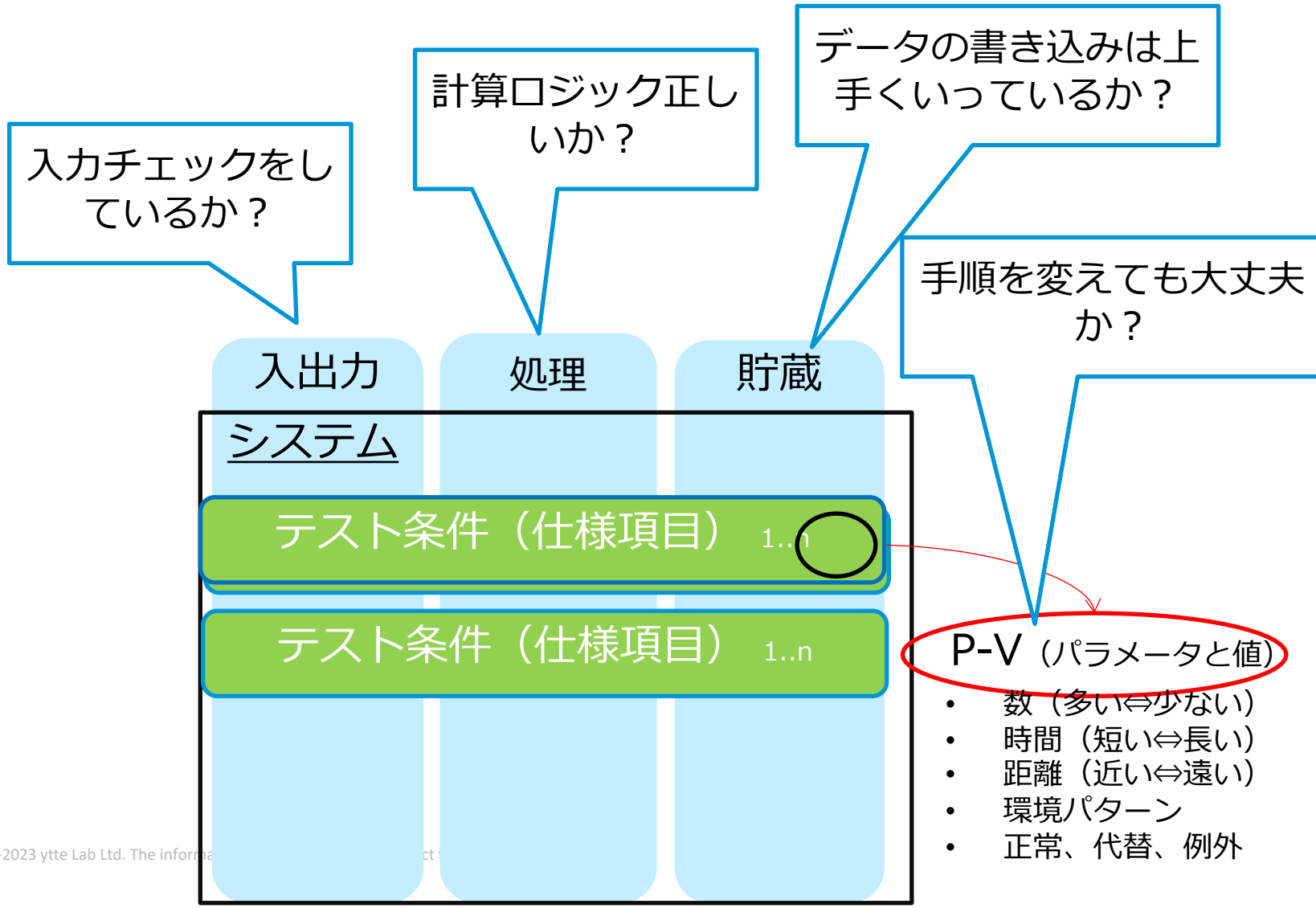


図2　状態遷移テストで確認していること

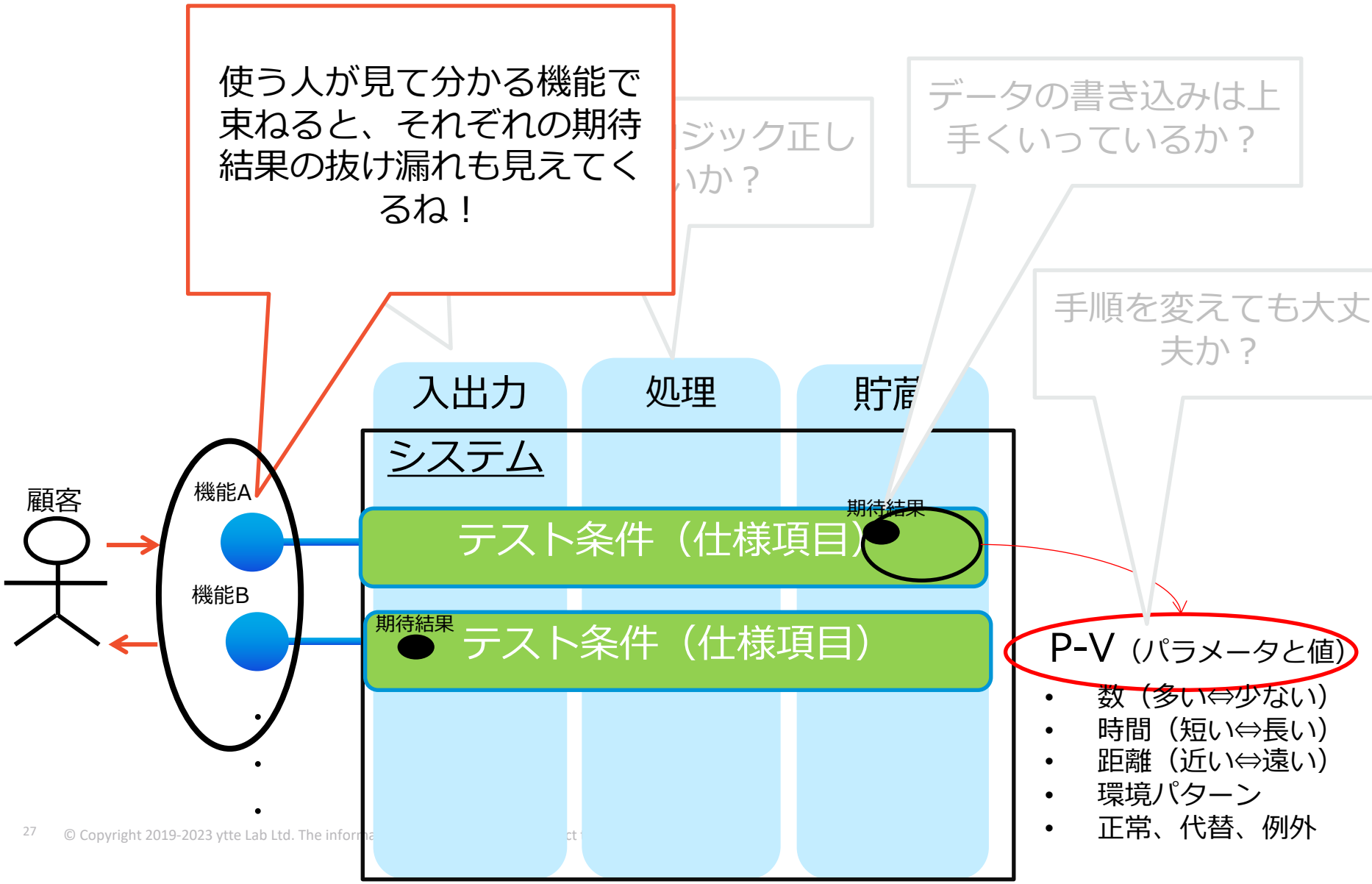
© Copyright 2019-2023 ytte Lab Ltd. The information contained herein is subject to change without notice.

外部からのシステム操作を軸に操作の事前条件をP-Vにしてテストする場合（4.想定外のP-V）



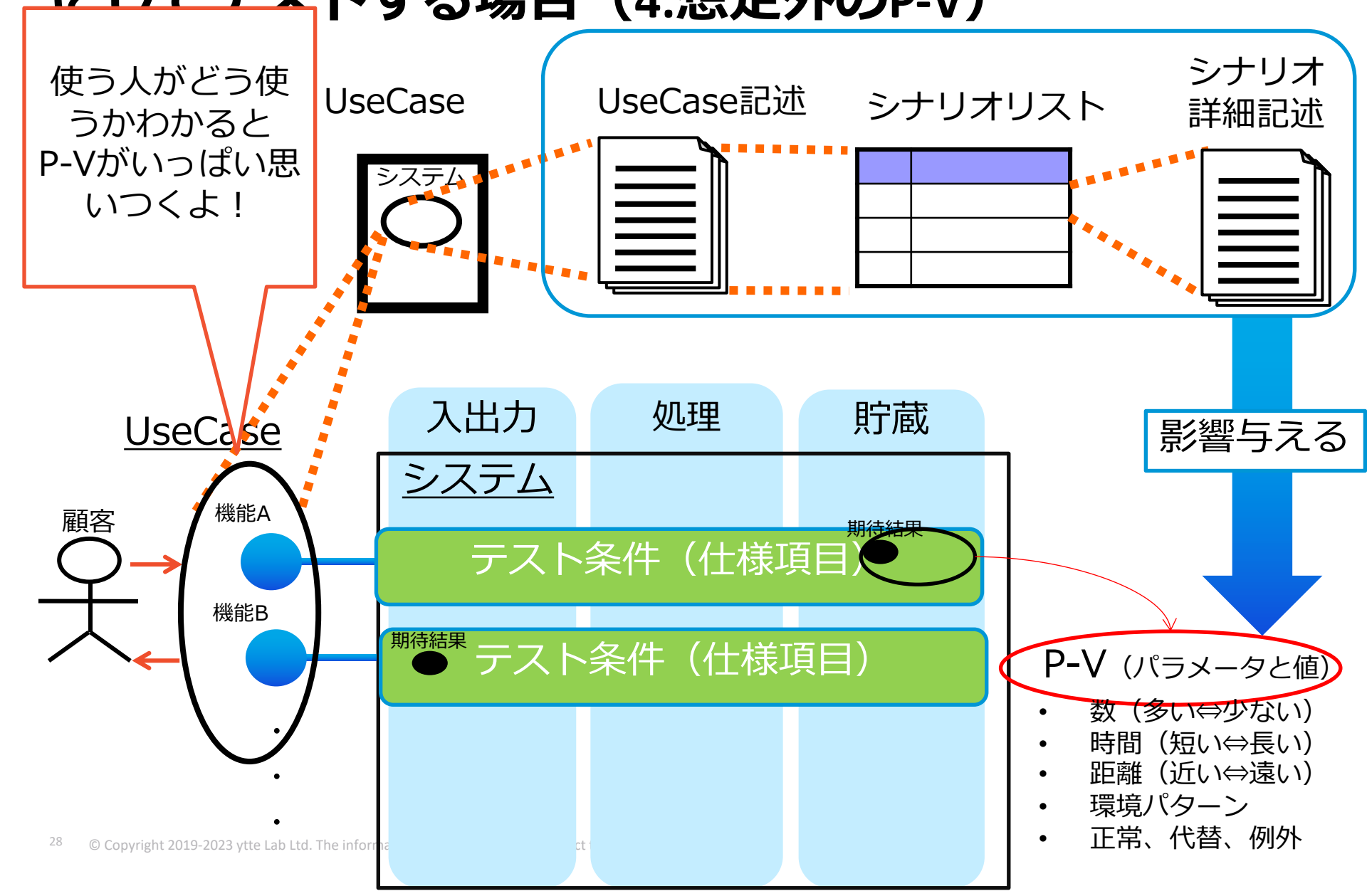
© Copyright 2019-2023 ytte Lab Ltd. The informa

外部からのシステム操作を軸に操作の事前条件をP-Vにしてテストする場合（4.想定外のP-V）



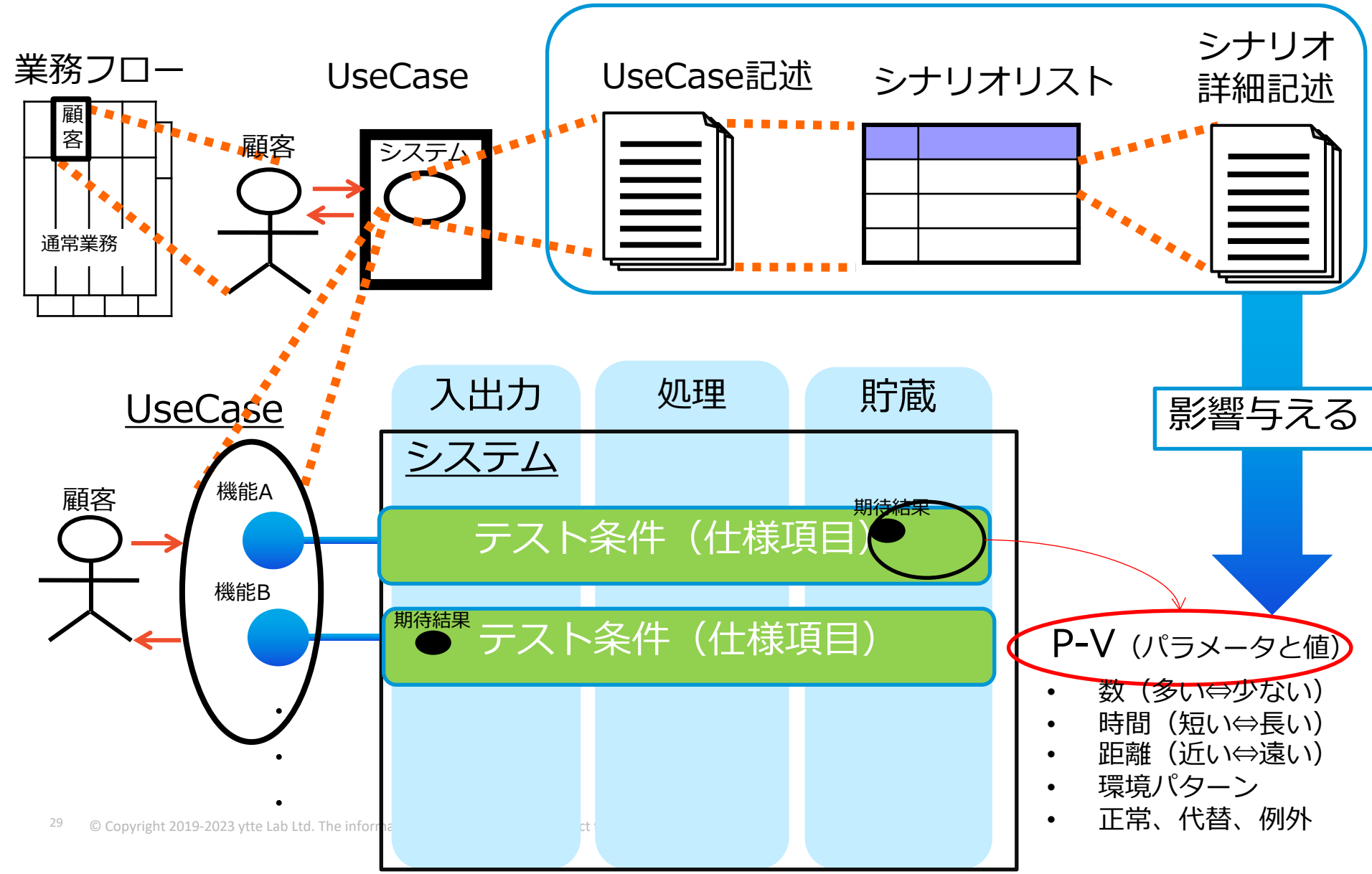
© Copyright 2019-2023 ytte Lab Ltd. The informa

外部からのシステム操作を軸に操作の事前条件をP-Vにしてテストする場合（4.想定外のP-V）



© Copyright 2019-2023 ytte Lab Ltd. The informa

外部からのシステム操作を軸に操作の事前条件をP-Vにしてテストする場合（4.想定外のP-V）



© Copyright 2019-2023 ytte Lab Ltd. The informa

テスト環境の準備を徹底して行う

準備タスクをリストアップする

- 準備完了日を決めて、確認作業をスケジュールに入れる
- 本番と変わるものがどれなのかわかるようにする
  - ex. コードブランチ、環境変数、サーバースペック、機能トグル、外部サービスエミュレーター
- スケジュール通りに活動が進んでいることを確認する

複数の開発チーム／外部サービスを統合する場合は、統合して確認する担当を置く

- 統合スケジュールを作る
  - ブランチ分けて開発してる場合のマージ確認をスケジュールに入れる
  - 段階的な環境構築の確認作業をスケジュールに入れる
- スケジュール通りに活動が進んでいることを確認する

環境を複数使う場合は、同じものをテストできるようにする

- デプロイに使うブランチを合わせてテストするバージョンを合わせる
  - テスト対象のブランチに入ってはいけないPRが入らないような仕組みをつくる
  - テスト環境へのデプロイの仕組みを用意して間違いが入らないようにする

© Copyright 2019-2023 ytte Lab Ltd. The information contained herein is subject to change without notice.

関係者とコミュニケーションする

プロジェクト体制に合わせたコミュニケーションパスを確保する

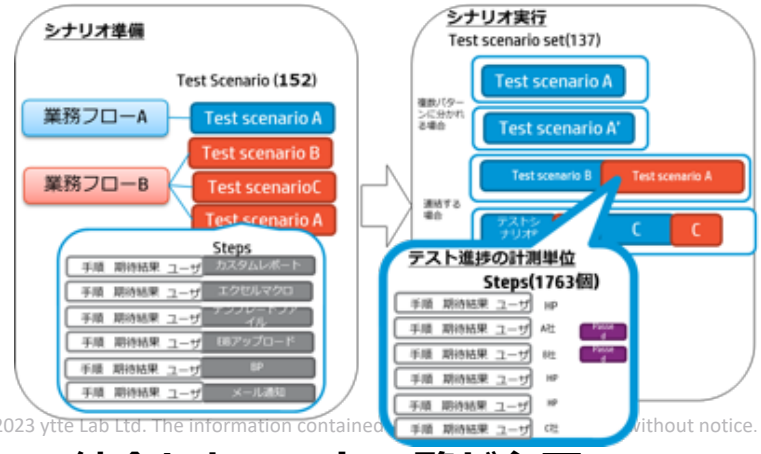
- アジャイルな場合、トラディショナルな大規模開発の場合など、場合にあったパスを作る

プロダクトリスク、テストプラン、テスト分析結果は必ず伝えて合意する

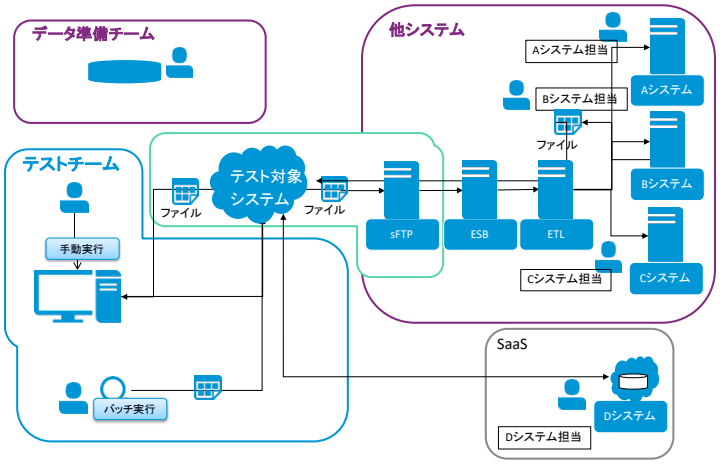
- チーム（PM、開発者、デザイナー、テスト担当者）みんなに伝えて合意する
- 要点は必ずアウトプット＆会話するようにして、「空中戦」にしない

複数チームがまたがる場合はさらに意識合わせに気を配る

- マスターテストプラン、統合したテスト一覧を作る



統合したテスト一覧が必要



このような複合システムの場合

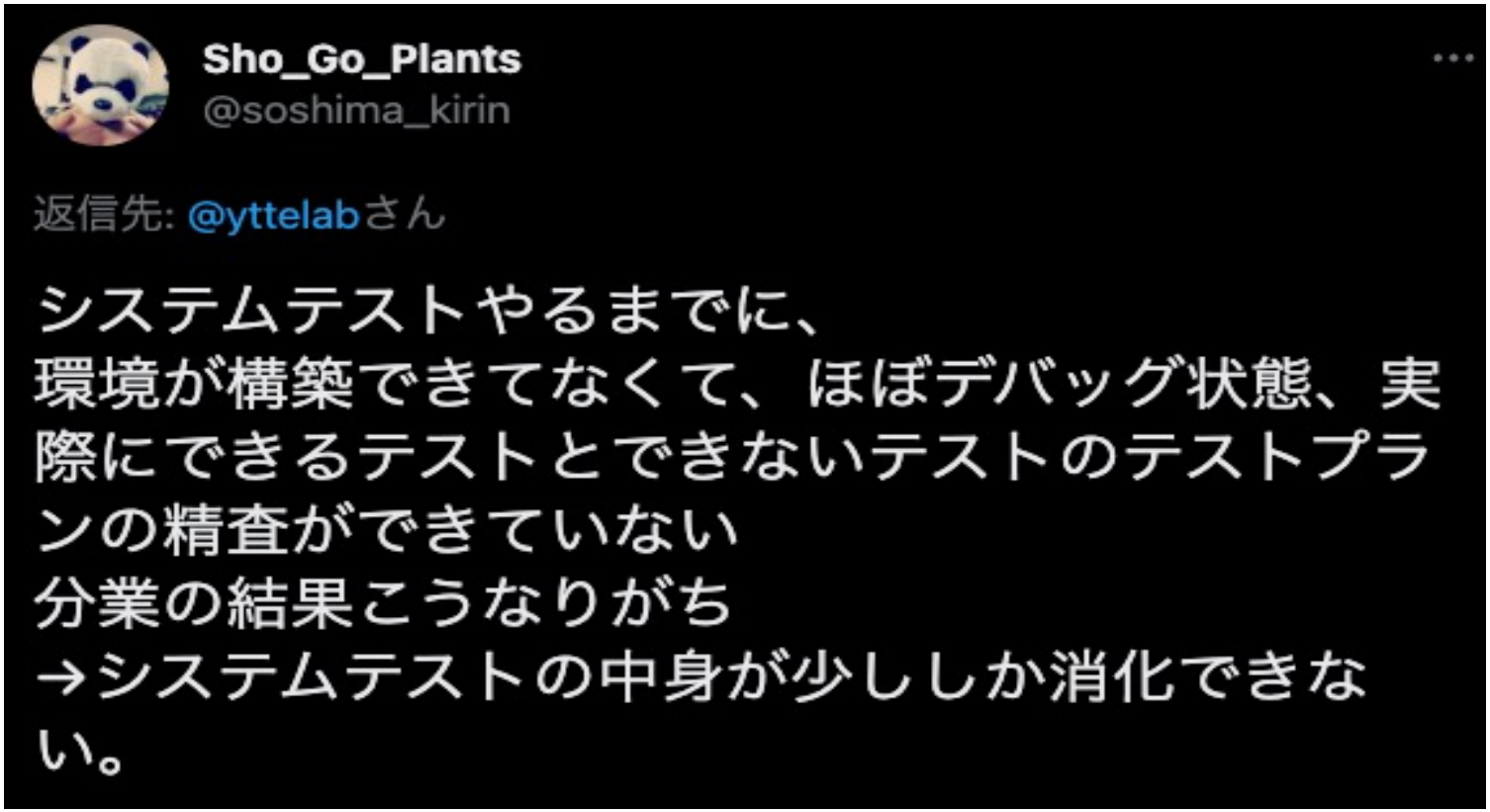
© Copyright 2019-2023 ytte Lab Ltd. The information contained ... without notice.

システムテストのアンチパターン

システムテストのアンチパターン
なぜ欠陥は入るのか？

© Copyright 2019-2023 ytte Lab Ltd. The information contained herein is subject to change without notice.

アンチパターン その1



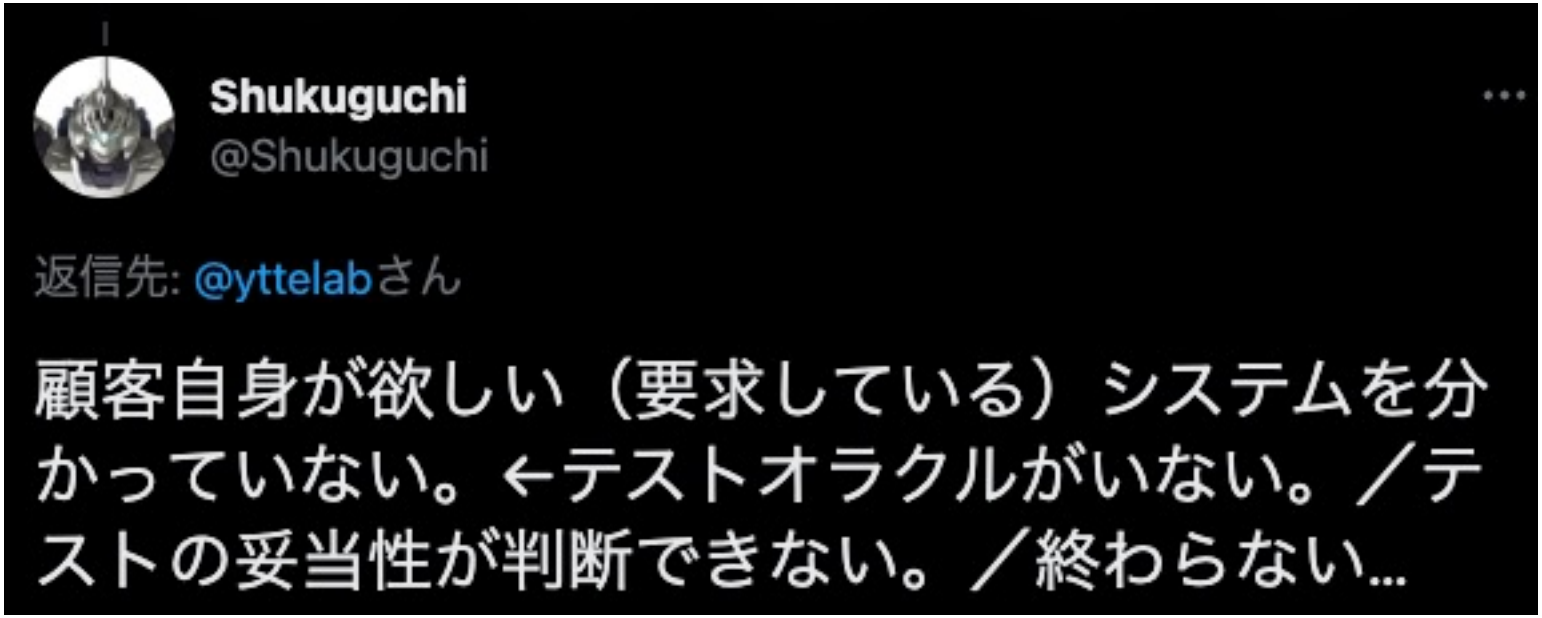
テストマネジメントの問題

状況を全体的に見てて判断したり調整するテストマネジメントの役割がいない

- **システムテストの実践Tipsの2（環境）と3（コミュニケーション）を誰かがやる**

© Copyright 2019-2023 ytte Lab Ltd. The information contained herein is subject to change without notice.

アンチパターンその2



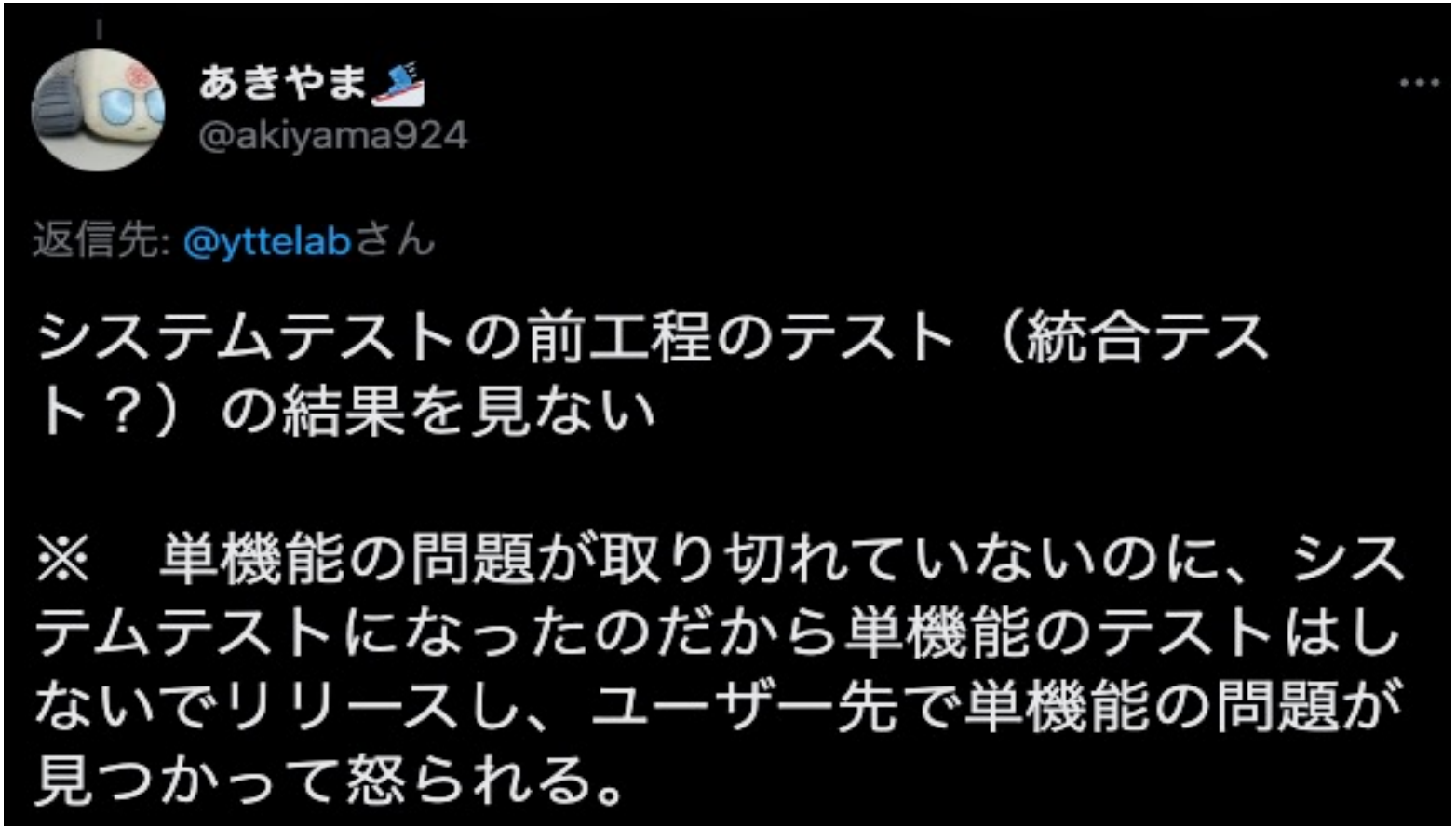
テストマネジメントの問題

今日のここまでの話で出てきていませんが…「どこまでやるか決めましょう」と提案してプロジェクト全体をリードする

- **（ここまで大問題ではないですが）仕様や要求の欠陥が見つかった時の迅速な決定はとても重要なので、テストする側が判断を前進させるための対応をするのは重要**
- **ただし…本当にどうしようもなければ、プロジェクトを中止するしかない**

© Copyright 2019-2023 ytte Lab Ltd. The information contained herein is subject to change without notice.

アンチパターン その3



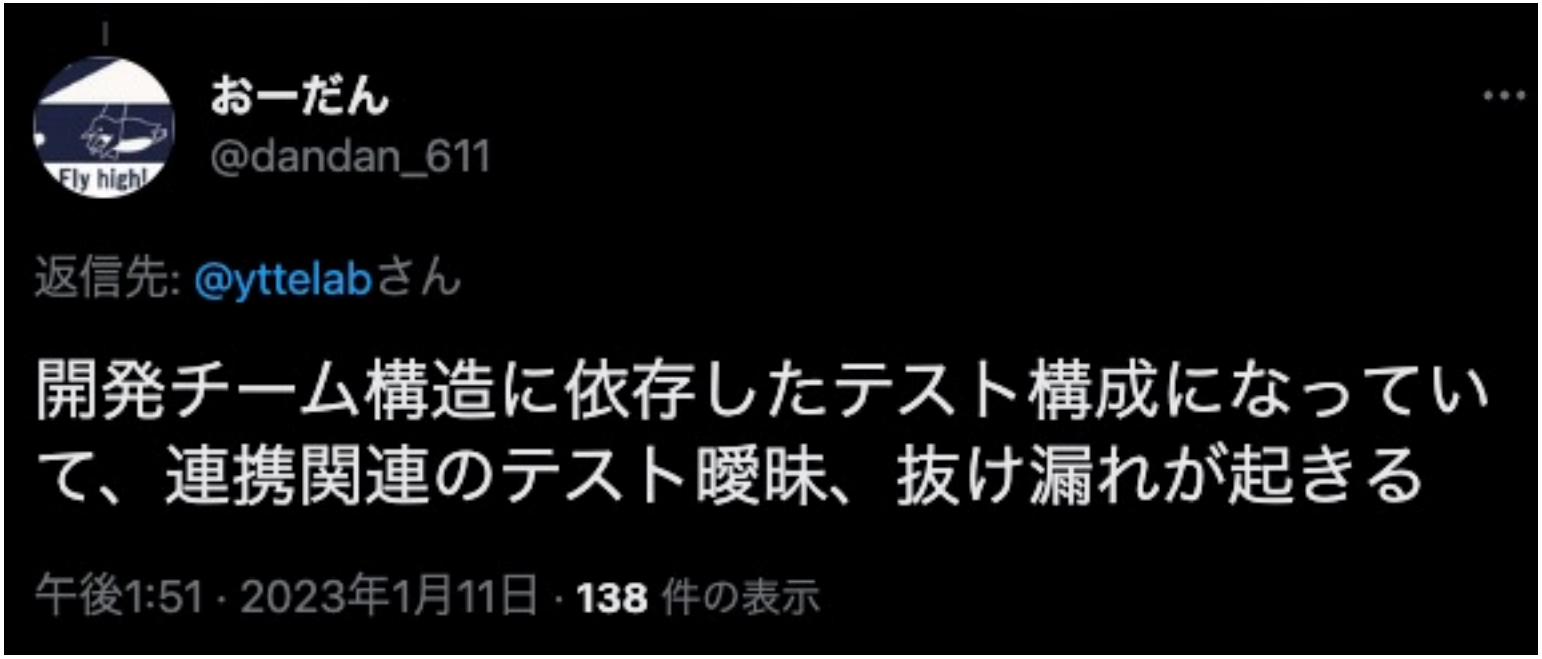
テストマネジメントの問題

テストレベルの責任範囲、テストアプローチ原則の合意不足の問題

- **誰がどこまでテストすべきかが曖昧な可能性**
  - 本講演の文脈だと、システムテストアプローチの原則1と原則3の齟齬が起きてる

© Copyright 2019-2023 ytte Lab Ltd. The information contained herein is subject to change without notice.

アンチパターン その4



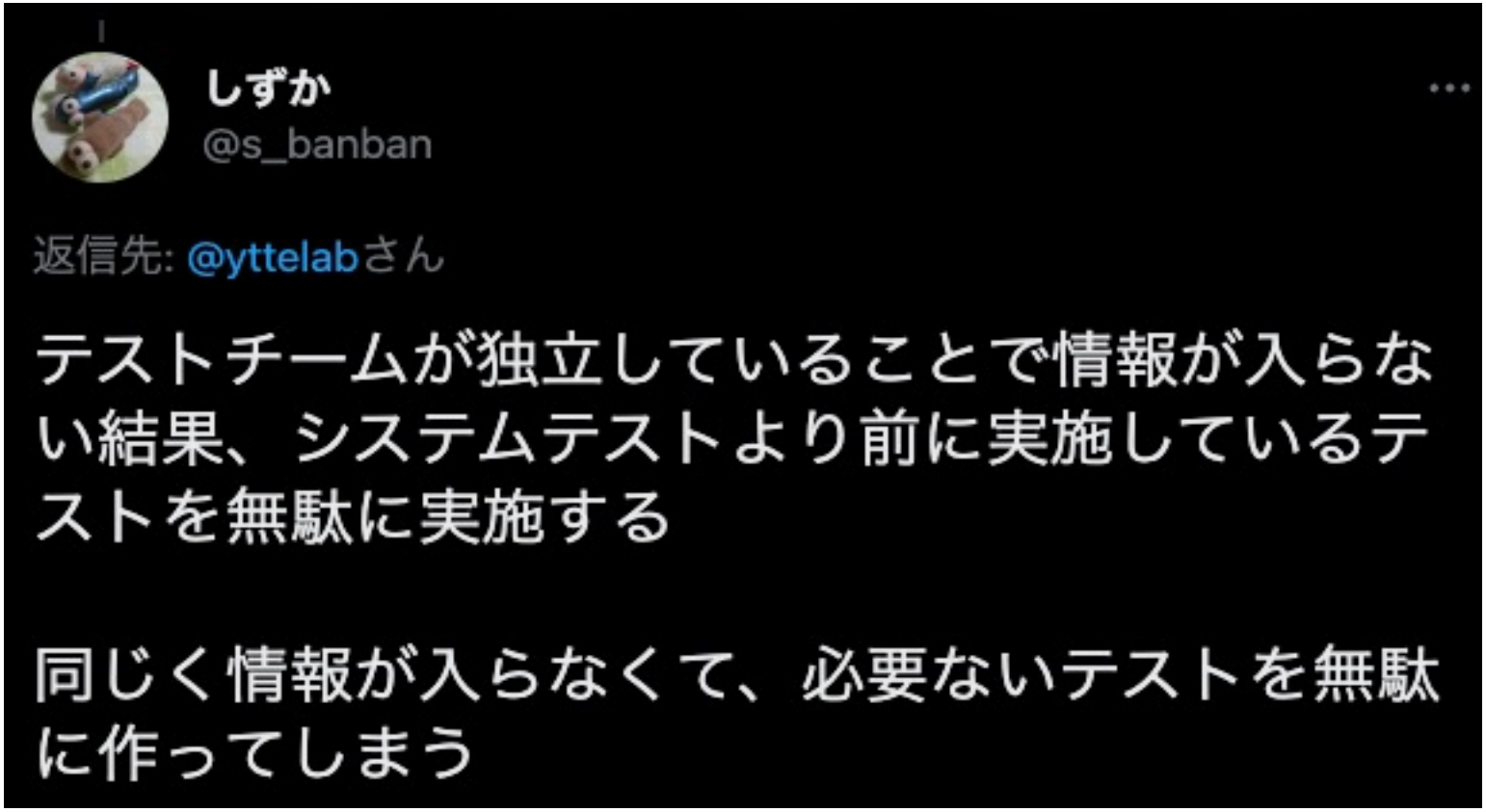
テスト分析、テスト設計の問題

システムテストアプローチの原則2、原則3ができてない

- **テスト分析に必要なテストベースはあらゆる手段で入手する**
- **テスト対象を抽象的にイメージしながら作業する**

© Copyright 2019-2023 ytte Lab Ltd. The information contained herein is subject to change without notice.

アンチパターン その5



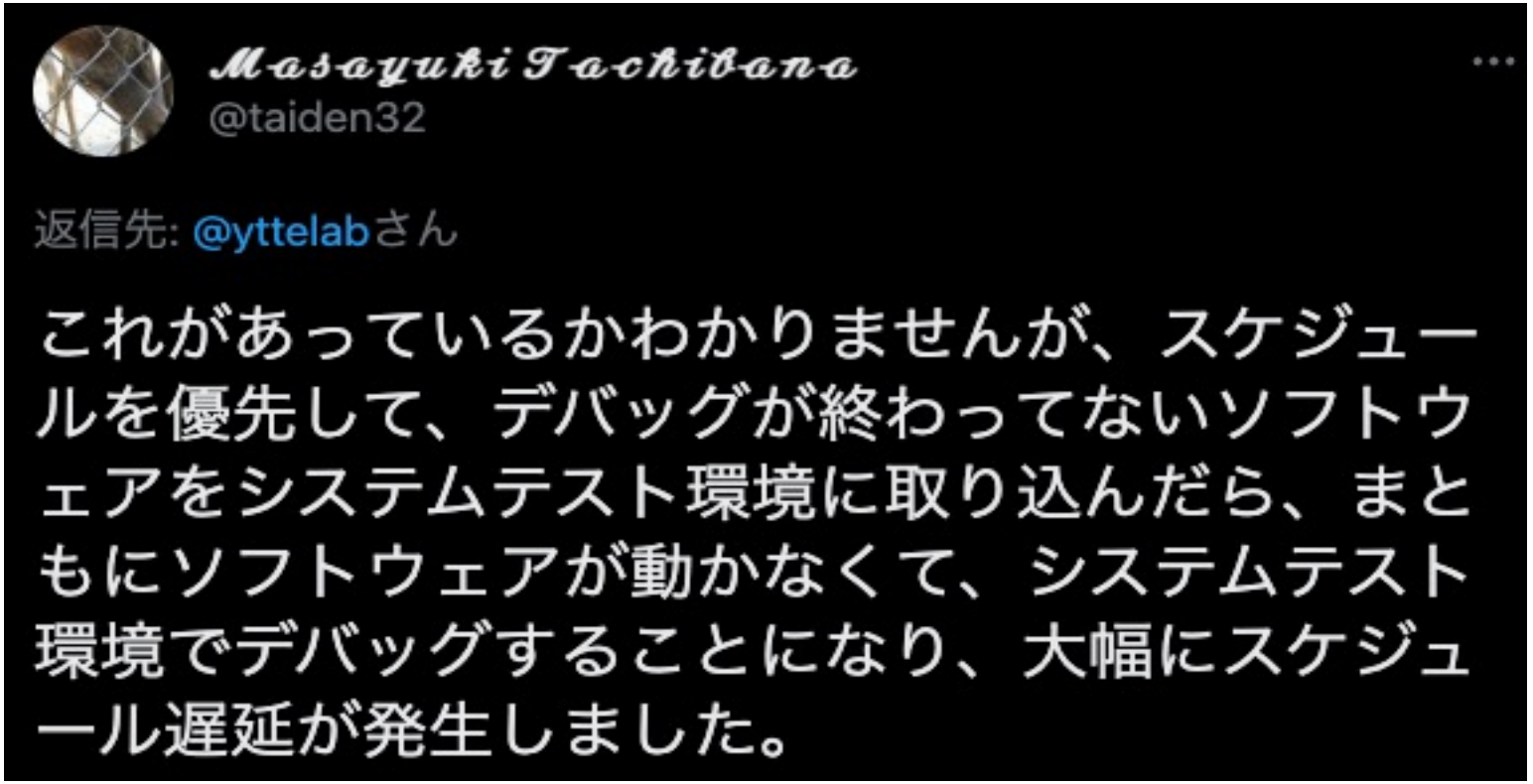
テストマネジメントの問題

システムテストの実践Tips３（コミュニケーション）ができてない

- **プロダクトリスク、テストプラン、テスト分析結果は必ず伝えて合意する**

© Copyright 2019-2023 ytte Lab Ltd. The information contained herein is subject to change without notice.

アンチパターン その6



テストマネジメントの問題

ソフトウェアテストの原則(テストレベル)に則ってない問題もあるが…コミュニケーションをしていればスケジュール遅延が起こることは自明なので、そこで何ができるかをあらかじめコミュニケーションするのも必要

- **テストプランの開始基準、中断基準を合意する**
- **プロジェクトリスクのマネジメントをしてトリガーポイント設定してスケジュールの遅延を早期に決断する**

© Copyright 2019-2023 ytte Lab Ltd. The information contained herein is subject to change without notice.

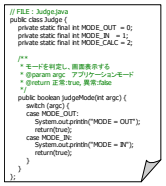
なぜ欠陥が入るか？（テストにもろ影響する）

プログラマーの技術力の問題だけではない

プログラマー

実装

コード



うっかり勘違いする
・ex.足すところを引いてしまった

アルゴリズムの使い方間違える
・ex.検索、ソート

使える値を間違える
・ex.境界値、null値

条件分岐を間違える

型（整数か？文字か？）を間違える

間違った入力があったときのこと考えない
・ex.在庫がない時の発注指示

違うデータを使ってしまう
・ex.受注データ、発注データ

同時に複数の処理をする対処を忘れる

大量データの処理をする対処を考えない

単体テスト

自分でちゃんと作ったか
確認すれば間違いは見つかる

次のページ

© Copyright 2019-2023 ytte Lab Ltd. The information contained herein is subject to change without notice.

なぜ欠陥が入るか？（テストにもろ影響する）

今日、これまで説明したようなソフトウェア開発は一人では行わない

規模の増大（人間の理解を超える）、コミュニケーションロス

エンドユーザー

誤解や漏れ

Sier
（PM、アナリスト）

誤解や漏れ

ソフトハウス
（PM、設計者）

誤解や漏れ

ソフトハウス
（設計者）

誤解や漏れ

ソフトハウス
（実装、テスト）

口頭だけで話を決めて開発をしてしまう

こういう問題が起こります

© Copyright 2019-2023 ytte Lab Ltd. The information contained herein is subject to change without notice.

なぜ欠陥が入るか？（テストにもろ影響する）

開発の段階的詳細化に対する考え方が違う（背景の違い/規模の違い）

- 外部設計、内部設計
- 論理設計、構造設計

VS　基本設計、詳細設計

※　各自が一度覚えてしまった考え方をひとつに合わせていくのは非常に大変（チームとしてバラバラ）

ソフトウェア開発で行うことは、活動としては不変であるが…

実際はこれらの活動の手順を決めてプロセスとなるように設計しないといけない

- 反復型開発（イテレイティブ型開発）
  - アジャイル
- 派生開発（バージョンアップ開発）

VS　ウオーターフォール
（シーケンシャルな新規開発）

意味を理解せず、言われたことだけやると間違いに気づかない

そもそも原理原則として必要な活動がおざなりのまま開発を行っている

- 要求獲得、要求分析をせずに開発をする（言われたことしかしない）
- 設計して実装するといった、開発には段階があることを知らない（いきなり作る）
- テストが「実行だけではない」ことを知らない（先人から学ばない）
- アウトプットして確認しあわない（他人と共有しようとしない、身近な人にしか伝えない）
- マネジメント活動を行わない（整理整頓することを無駄だと思っている）

© Copyright 2019-2023 ytte Lab Ltd. The information contained herein is subject to change without notice.

まとめ

✓ ソフトウェアテストの原則

- 全部テストはできないので、テストレベルなど棲み分けが大事
- 基本変わらない原則と、コンテキストでHowを変える原則

✓ システムテストの位置付け

- 要件として提示したシステムの論理的な振る舞いをテストする

✓ システムテストの基本

- だんだんテストを広げる、P-Vを爆発させない

✓ システムテストの実践

- 3つのモデル、ユーザーの利用シーンからP-V追加
- テスト環境、コミュニケーション

✓ システムテストのアンチパターン

- 原理原則をおざなりにし、コミュニケーションロスで問題が起きる

© Copyright 2019-2023 ytte Lab Ltd. The information contained herein is subject to change without notice.

参考文献

湯本剛「開発を楽しくするソフトウェアテスト」（ JaSST東北2013 基調講演）

湯本剛「これからの開発に役立つソフトウェアテスト」（ JaSST四国2019 基調講演）

湯本剛「開発プロジェクトにおけるテストマネジメントの基本と実践」（ JaSST北陸2019 基調講演）

湯本剛「納得できるテストをするためのモデリング講座 」(yttelabセミナー)

湯本剛「テスターのプロフェッショナリズムについて」（yttelab講演資料）

湯本剛 「（2021年度版）ソフトウェアテストの上流設計」（note magazine）

湯本剛「バージョンアップ開発や長谷開発の時の影響範囲をテストしてますか？」(sqripts連載)

「ソフトウェアエンジニアリング講座　テスティング実践編」（豆蔵）

ISTQB 「ISTQBテスト技術者資格制度Foundation Levelシラバス日本語版 Version2018.J01」
(NPO法人ASTER ,JSTQB)2019

JaSST東北実行委員&湯本剛「ソフトウェアテスト分析手法実践セミナー」（日科技連セミナー）

湯本剛 「テストのマネジメント・技法」（日科技連オンデマンドセミナー ）

Twitterで問いかけたら教えてくれた@akiyama924 @s_banban @Shukuguchi @taiden32 @soshima_kirin @dandan
ありがとうございました！

© Copyright 2019-2023 ytte Lab Ltd. The information contained herein is subject to change without notice.

ありがとうございました!

© Copyright 2019-2023 ytte Lab Ltd. The information contained herein is subject to change without notice.