ソフトウェアテスト、一番最初にやるべき大事なこと

～テストの目的をプロジェクトで共有する～

自己紹介

湯本 剛

Consultant at HPSoftware Japan

ソフトウェアテスト現場で10年ほど経験を積んだ後、テストプロセス改善のコンサルティング、教育に約7年ほど従事

2010年8月よりHP Softwareのテストツール導入支援コンサルタントに転職し、主にテストケース管理ツール、キャプチャリプレイツールの導入支援に従事

外部活動

NPO法人ASTER　理事

- JSTQB 技術委員
- ISTQB CTAL WPメンバー
- ソフトウェアテストシンポジウム（JaSST）東京実行委員

ISO/IEC JTC1 SC7 WG26 エキスパート

日科技連SQiPステアリング委員

書籍、Web記事や雑誌への執筆や翻訳

得意分野

テストマネジメント、テスト分析

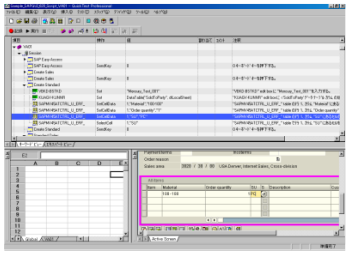
HPのアプリケーション品質管理ツール群

HPのテストツール全体像

キャプチャリプレイツール

QuickTestProfessional

①アプリケーション操作を記録(スクリプト化)
②スクリプトを再生し、自動検証
データ駆動、キーワード駆動テスト



手動テスト支援ツール

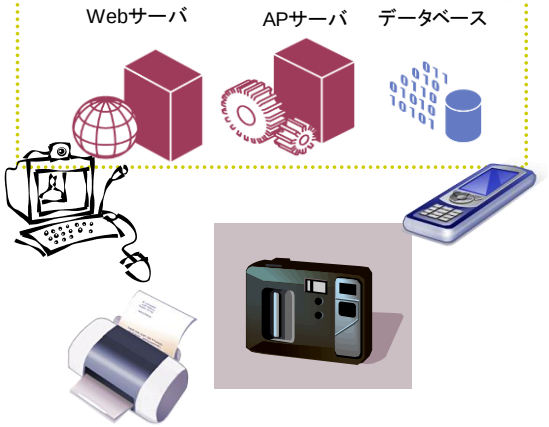
Sprinter

スタブツール

Service Virtualization

外部サービスをシミュレーションして、実際のサービス稼動を想定したテスト環境を作成

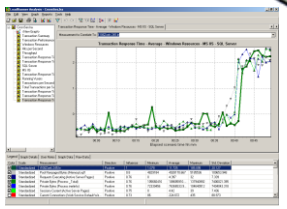
さまざまなテスト対象

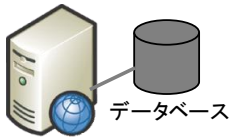


負荷テストツール

PerformanceCenter &LoadRunner (with Diagnostics)

①アプリケーション操作を記録(スクリプト化)
②別PC上で多重実行
③パフォーマンスデータの収集
④結果分析





セキュリティテストツール

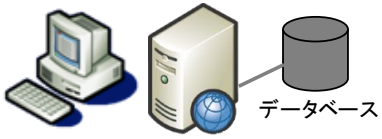
WebInspect,Fortify

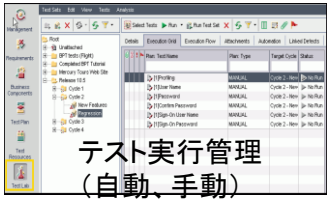
Webアプリケーションに対して様々なアタックをかけ、脆弱性を検証する

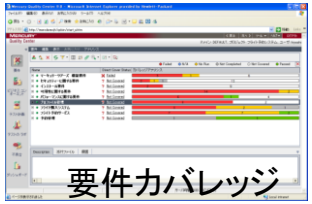
テストマネジメントツール

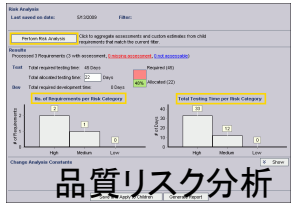
QualityCenter

リリース管理、要件管理
テスト条件、テストケースの管理
不具合管理
テスト実行管理(スケジューラ)
テスト結果収集、進捗参照









アジェンダ

ソフトウェアテストが直面する課題

- ソフトウェアテストはソフトウェア開発のボトルネック

ソフトウェアテストの全体像

- テストに必要な活動を知る＝何をよくしていけばよいかを知る

テストは何故やるのか？

- 目的と手段の階層構造

テストの目的とは？

- 最適なテストケースの集合を選定する基準

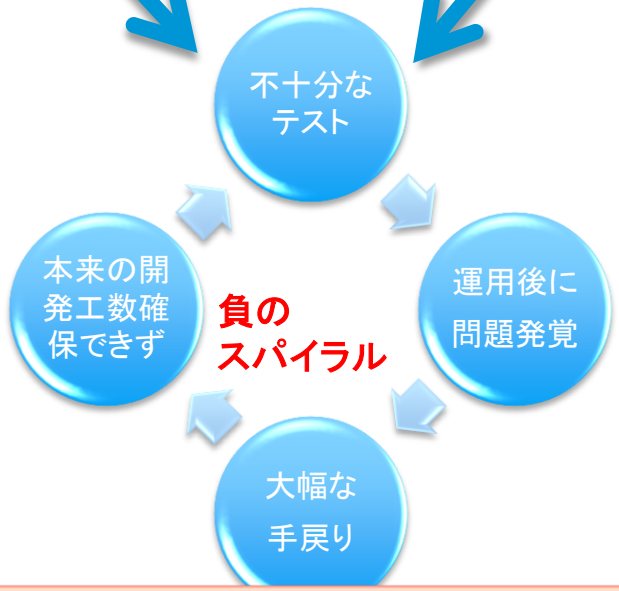
1. ソフトウェアテストが直面する課題

ソフトウェアテストはソフトウェア開発のボトルネック

ソフトウェアテストが直面している課題

ソフトウェアの大規模化 複雑化によるテスト増大

開発期間短縮化による テスト工数不足



テストを更に効率よく実施し、確実に品質を確保するには？

テストが開発のボトルネックに！

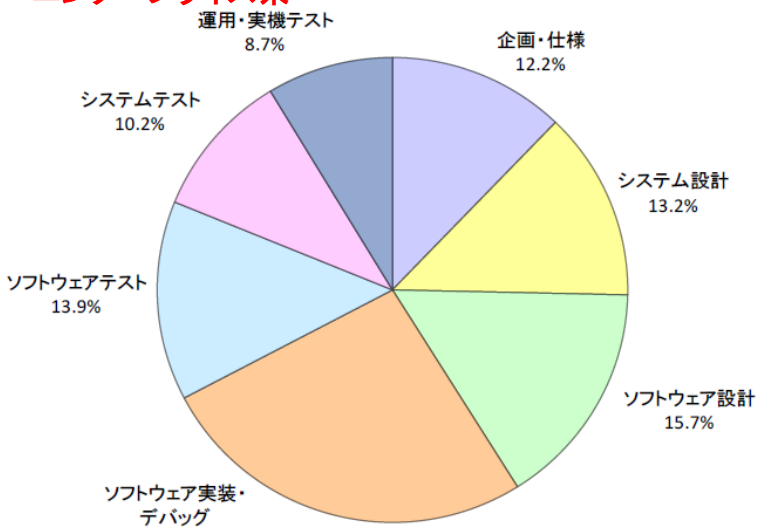
平均的にはテストは開発工数の約40%以上かかる（下記IPAデータより）

- ミッションクリティカルなシステムの場合、開発工数の90%がテストになる事例もある

ボトルネックであるテストの見直し→　開発全体を良くする相乗効果

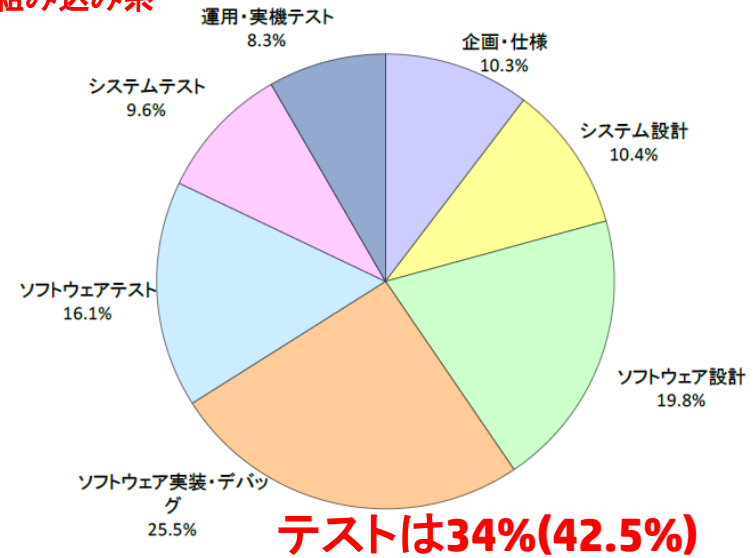
- 工数増になる3つの要因を解決する:「欠陥の特性」「開発効率化」「テストのやり方」

エンタープライズ系



テストは33.3%(42.1%)

組み込み系

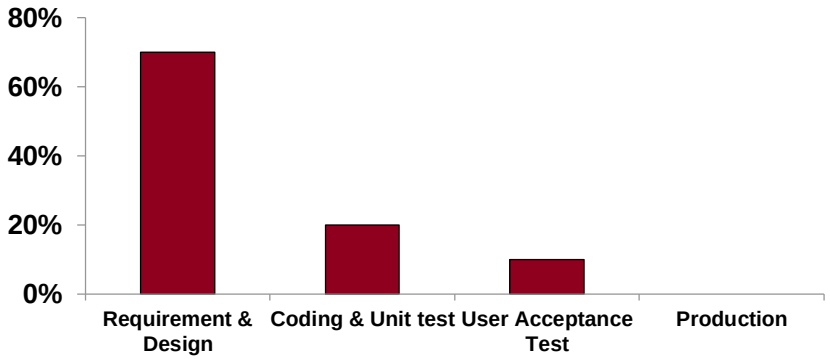


テストは34%(42.5%)

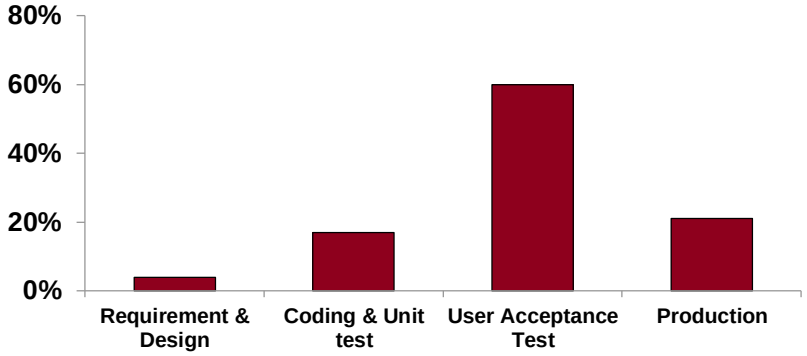
「ソフトウェア産業の実態把握に関する調査」2011年度版　IPA　より引用

テスト工数の増加要因(1) ---- 欠陥の特性

1.欠陥の多くは上流で発生

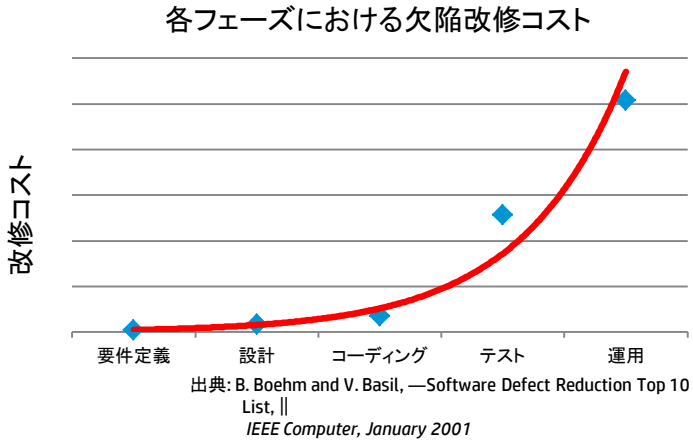


2.欠陥の多くは、下流で発見



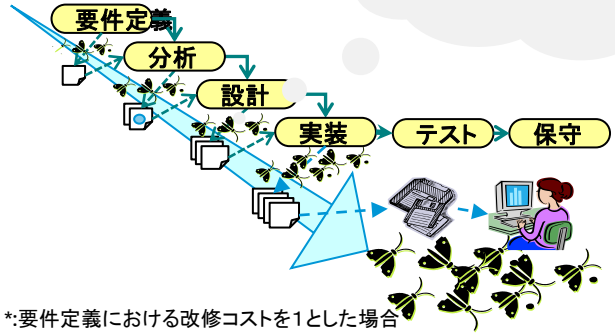
出典: NIST 2002 RTI Project 7007.011

3.欠陥改修コストは時間と共に増加



出典: B. Boehm and V. Basil, —Software Defect Reduction Top 10 List, ‖
IEEE Computer, January 2001

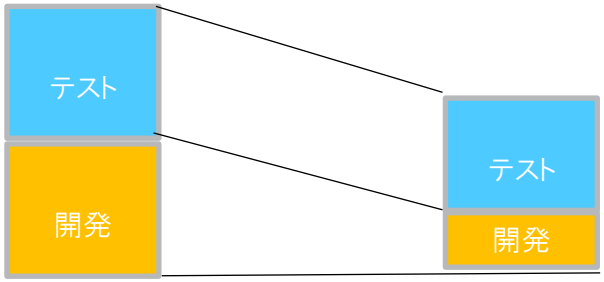
混入時は小さな問題でも
工程を経る毎に成長、
蓄積する。



*:要件定義における改修コストを1とした場合

テスト工数の増加要因(2) ---- 開発の効率化

改善前のアプリ開発全体の投資額



改善後のアプリ開発全体の投資額

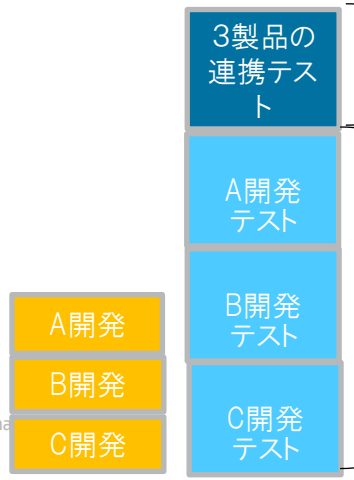
施策の例
・コンポーネントの調達
・基盤リソースのシェア

開発の効率化で同一のアプリケーション開発投資額でより多くのプロダクトを
アウトプットできるが、テストは減っていない。

開発規模を増加させると…

1プロダクトにおける
アプリ開発の投資額比率

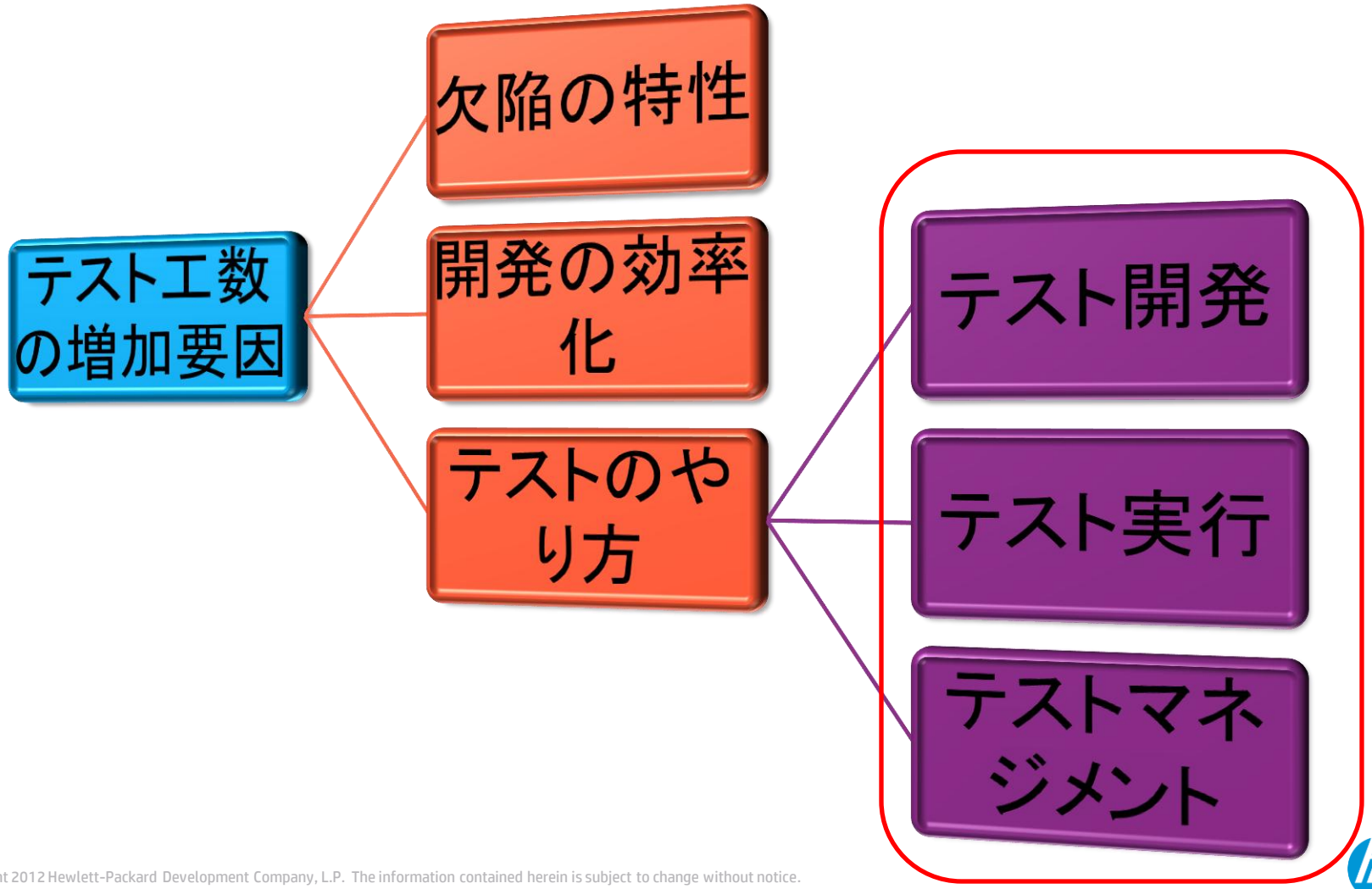
複数プロダクトにおける
アプリ開発の投資額比率



全プロダクトが連携し、
問題なく動くことも確認
しなければならないため、
更にテストが増える

テスト資産の再利用も
進まないため、
テストは減らない

テスト工数の増加要因(3)--- テストのやり方



ソフトウェアテストのやり方に起因

テストケースの作り方 = テスト開発技術

テストケースは開発するものであり、テストケースの品質を高めないといけない

- 「的を得た」テストケースを作る。的を得てないとテスト量が乗算で増加し、正しい判断を阻害する。
- 保守性の高いテストケースを作る。保守性が高くないとテストの再利用が困難になる。

テスト対象の操作や結果の確認の仕方 = テスト実行技術

テスト実行にも訓練が必要（実施順番の最適化、見落としの有無）

- 人間の見落としは、訓練しないと40%にも及ぶこともあると言われている
- 見落としを最小限にする仕掛け

テスト実行の自動化や、テスト環境作成の自動化などの技術を利用する

- 適用した箇所ではテスト実行工数が20倍～60倍の差となる
- テスト実行工数の最大50%を自動化した事例もあり

テスト開発、テスト実行を上手くやる方法 = テストマネジメント技術

マネジメント不備がムリ・ムダ・ムラを引き起こす

- テスト目的の明確化/優先度の継続的な見直し（テスト戦略、テスト計画）
- 資産の再利用に関する効率化（構成管理、トレーサビリティ管理）
- バグ発見時のやりとりの効率化（正しい情報のやりとり）
- テスト結果の見える化に関する効率化（モニタリング、報告書作成）

テストのやり方とテストの現状

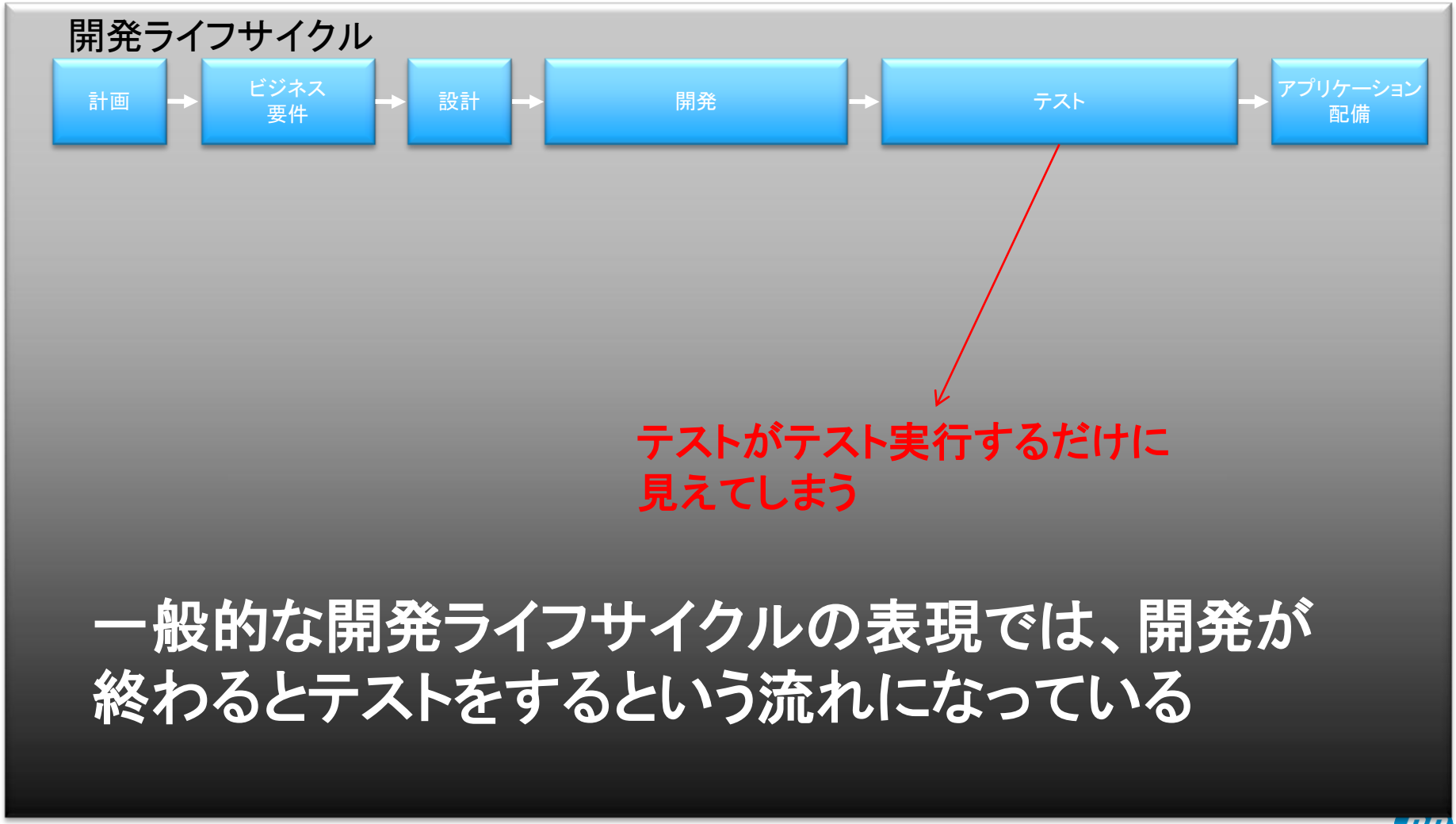
テストのやり方	テストのよくある現状
テスト開発技術 →	人海戦術 – 考える時間を惜しんで人を投入 – 時間のある限りひたすらテスト実行を行う
テスト実行技術 →	手動によるテスト – テストデータの準備 – テスト環境の準備 – テスト実行 – テスト結果のモニタリング、正解のジャッジメント
テストマネジメント技術 →	原始的なテストマネジメント – エクセルを使ったテストケースドキュメント – ファイルサーバーでのテストケース管理 – メールベースのバグ報告 – 報告書作成が手動転記

2. ソフトウェアテストの全体像

テストに必要な活動を知る＝何をよくしていけばよいかを知る

テストは実行だけではない

テストを実行だけでとらえていると何を改善すればよいかわからない



テストは実行だけではない

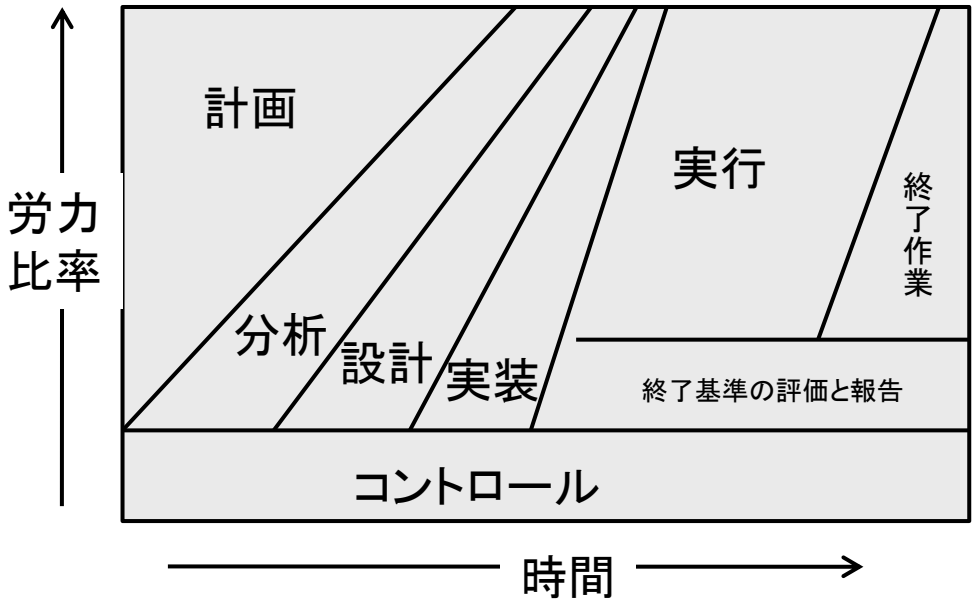
テストを実行だけでとらえていると何を改善すればよいかわからない

ISTQBが定義するテストプロセス

http://jstqb.jp/

ISTQBでは、テストプロセスは、以下の活動に分けて考えるとしている。

- 計画とコントロール
- 分析と設計
- 実装と実行
- テスト終了基準の評価と報告
- テスト終了作業



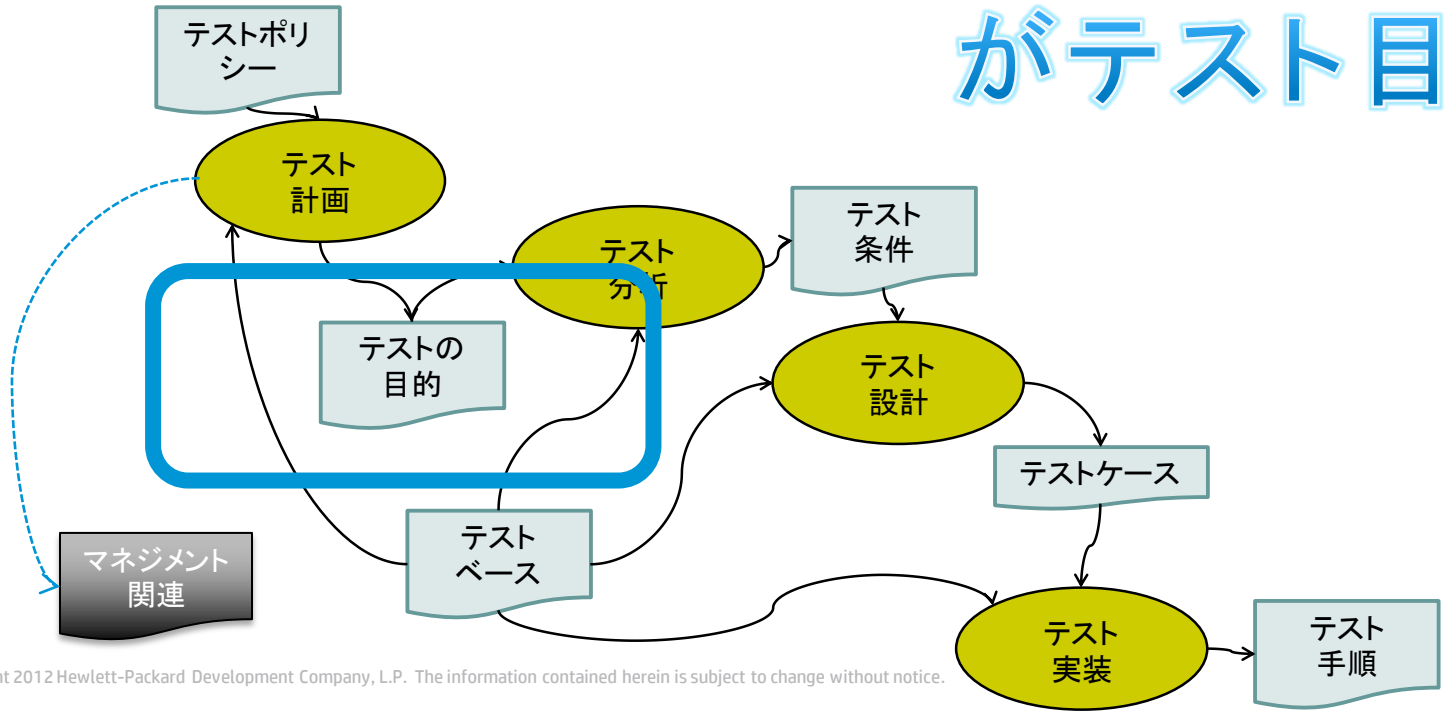
これらの活動はひとつづつ順番に行われるとは限らず、重なったり、並行したり、繰り返されたりする。

テストを開発するとは何か？

十分なテスト、目的に沿った「良い」テストを実行するための活動

- テストの目的やテスト対象を明らかにする（計画する）
- テストを造れるようにテスト対象を分解する（分析する）
- 目的に沿ったテストを造る（設計する）
- テストを実行できるようにする（実装する）
  - （手動なら手順を考える）（自動ならテストコードを作る）
  - 環境構築、テストデータ作成

テスト開発の最初のアウトプットがテスト目的



3. テストは何故やるのか？

目的と手段の階層構造

テストは何でやるんでしょう？

いろんな人に

いろんな想い

テストは何でするのでしょう？

ISTQBをベースに整理　(http://jstqb.jp/)

テストの必要性

Reduction（低減）

- システムやその文書をテストすることは運用時に問題が発生するリスクを軽減する

Evidence（証拠）

- 当初の目的であるビジネスの成立、または、産業界の規則、もしくは開発時の契約を満たしているかをチェックできる

Measurement（計測）

- テストを基にして、ソフトウェアの品質を測ることが可能

Effect（影響）

- テストプロセスを改善する中で、原因分析を行い開発にフィードバックすることでソフトウェア開発そのものの改善にもつなげる

テストの目的

Detection（発見）

- 欠陥の検出

Confidence（信頼）

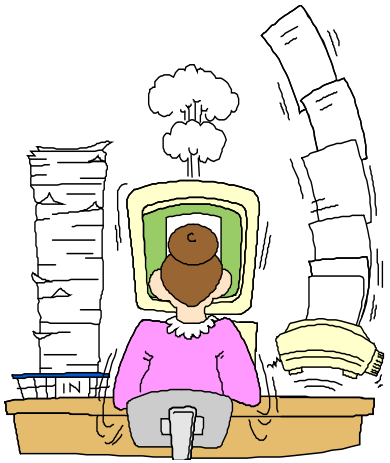
- 品質レベルが十分であることを確認

Instruction（情報提供）

- 判断のための情報を提示

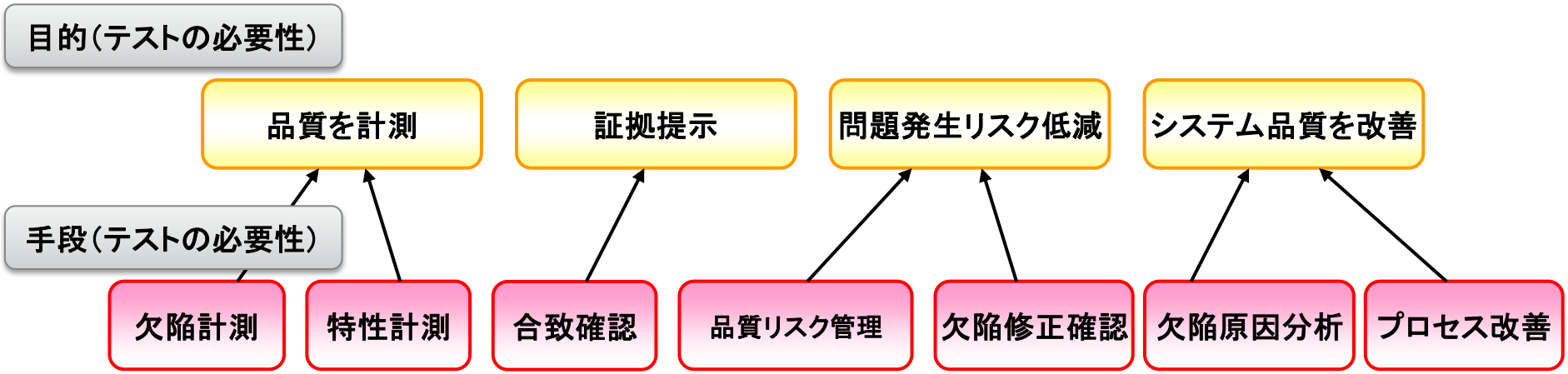
Forestall（予防）

- 早期にテスト視点で、レビューを行うことで欠陥の作りこみを防ぐ



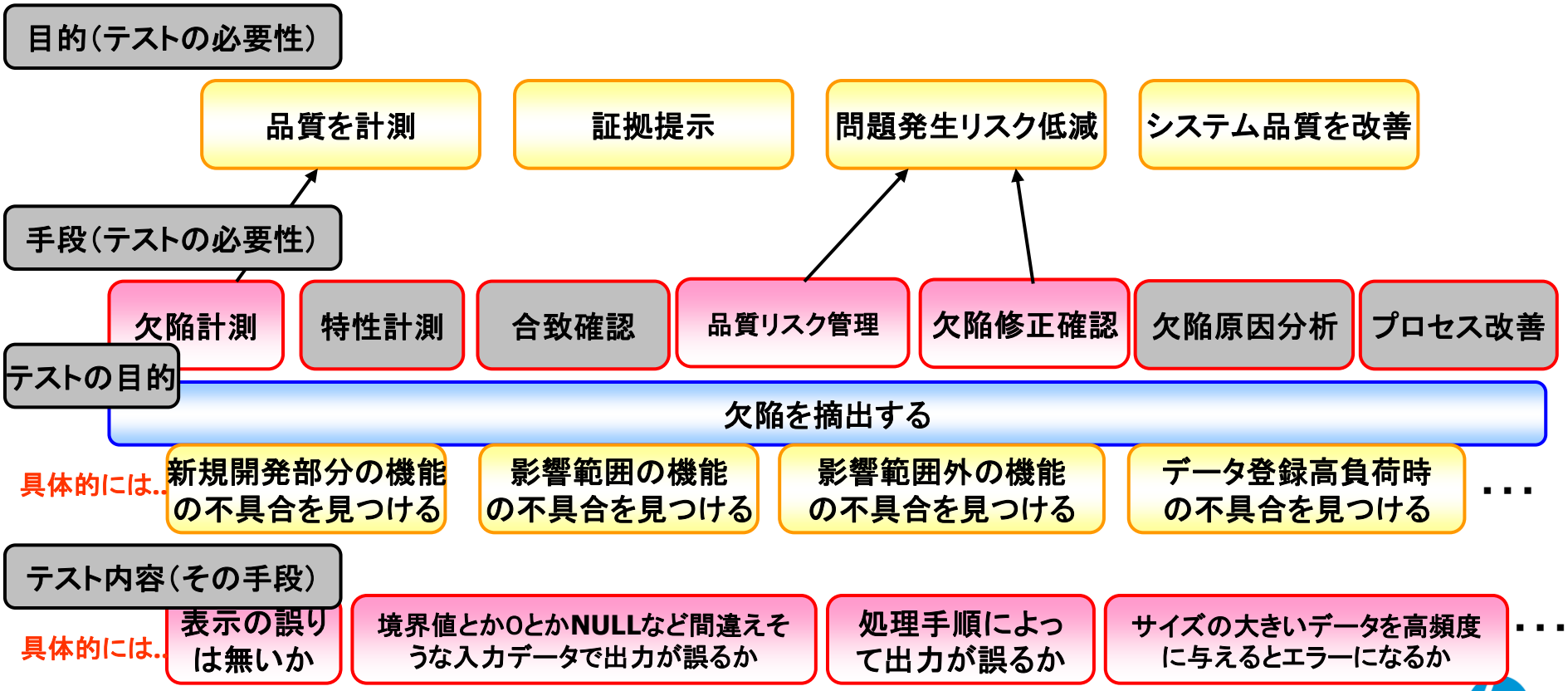
「テストの必要性」を目的と手段に分けると...

「テストの必要性」は目的と手段に分けることが出来る



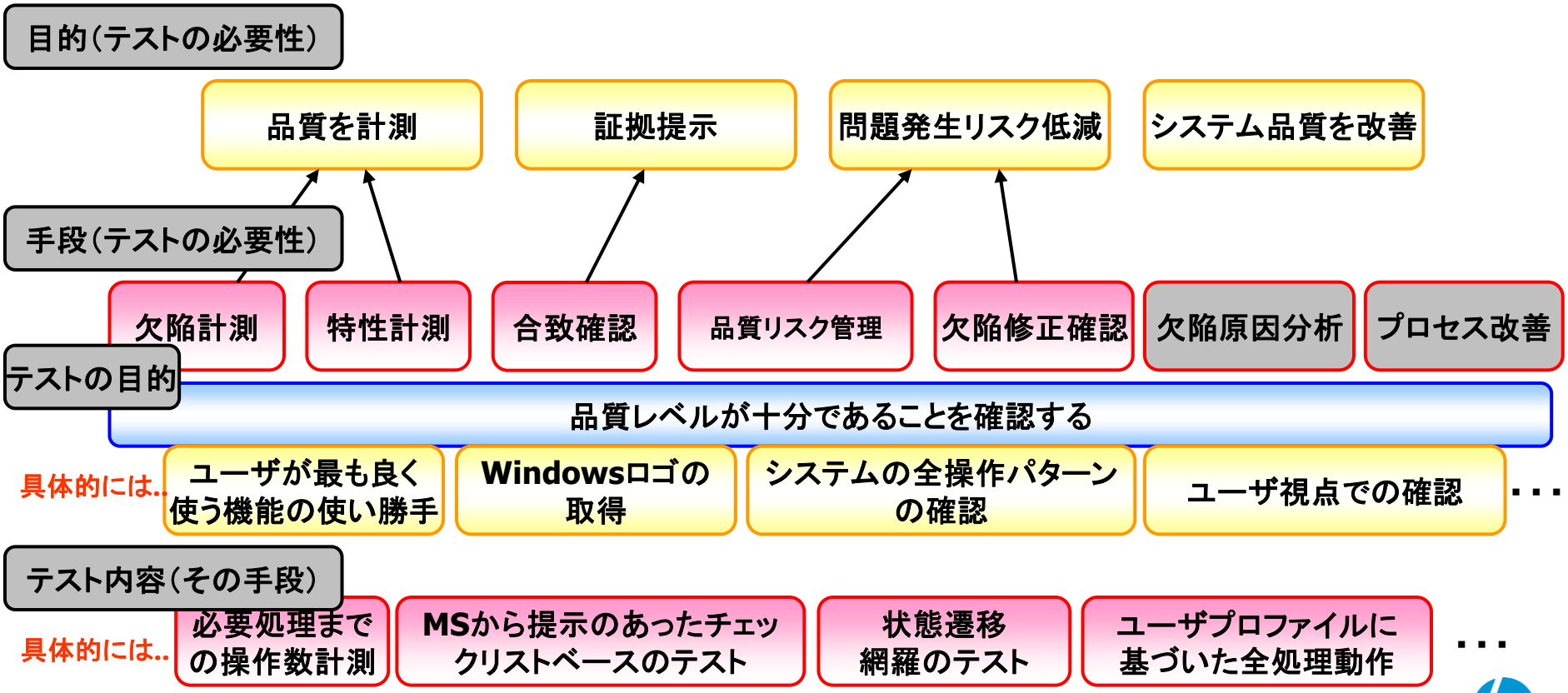
「テストの目的」を追加...欠陥摘出

テストの目的は「テストの必要性」を実現する手段



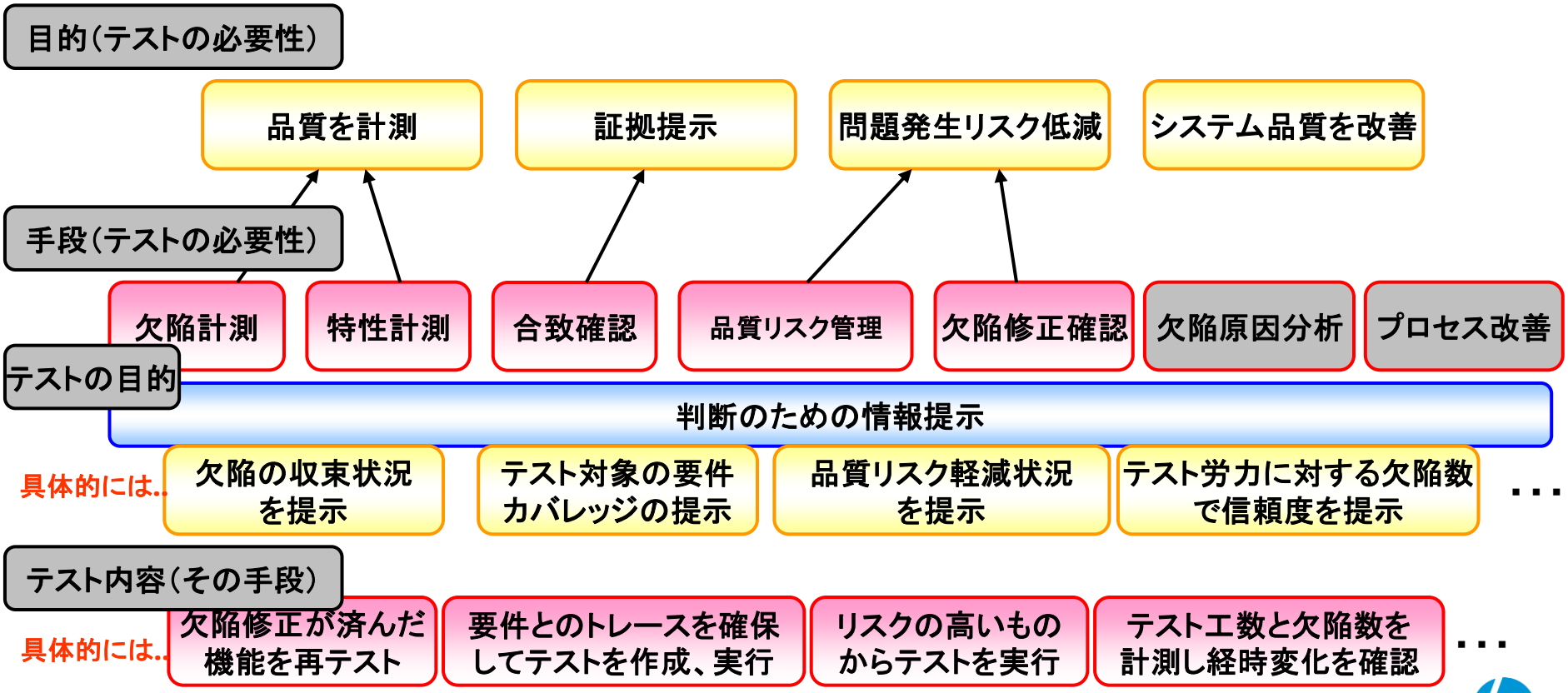
「テストの目的」を追加...品質レベルの確認

テストの目的は「テストの必要性」を実現する手段



「テストの目的」を追加...判断のための情報提示

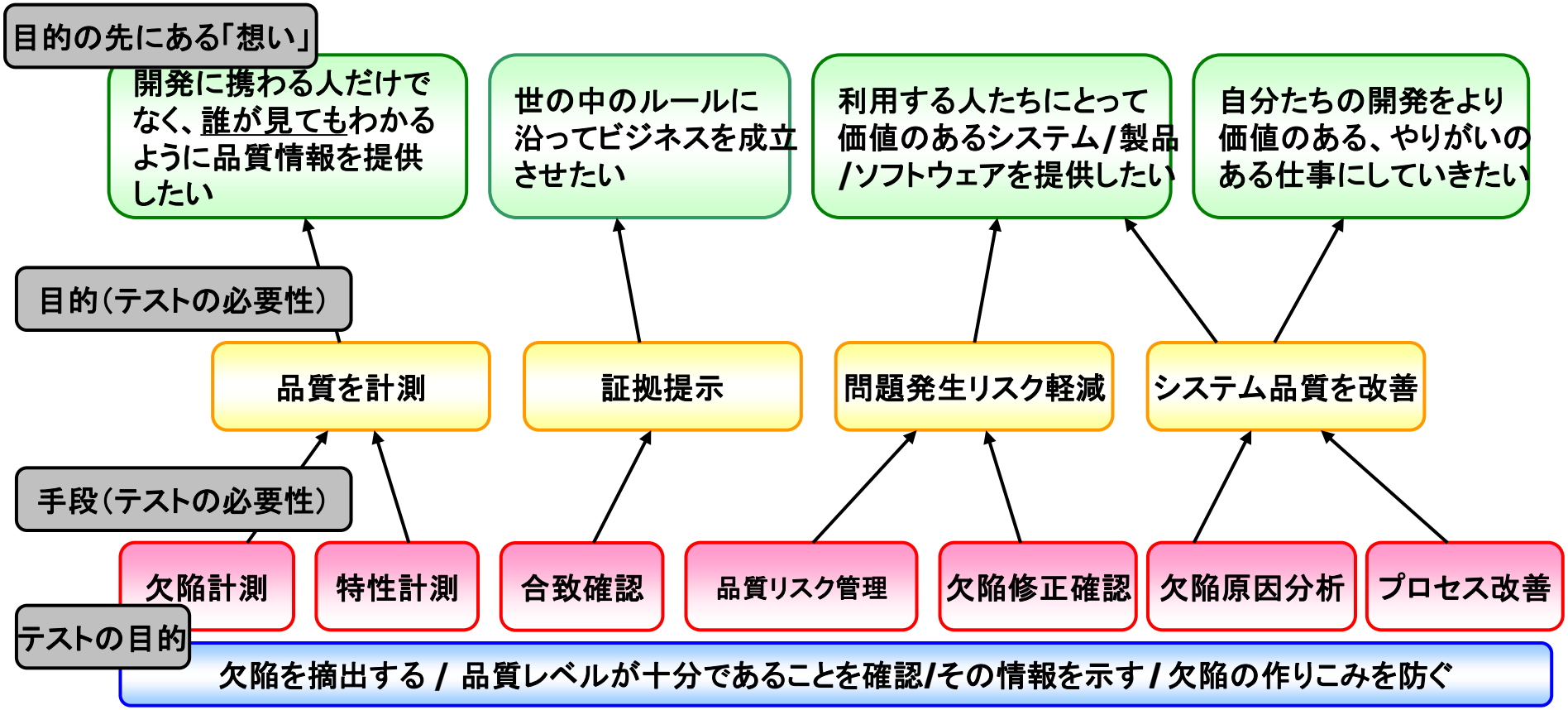
テストの目的は「テストの必要性」を実現する手段



「テストの目的」を追加...欠陥の作り込みを防ぐ

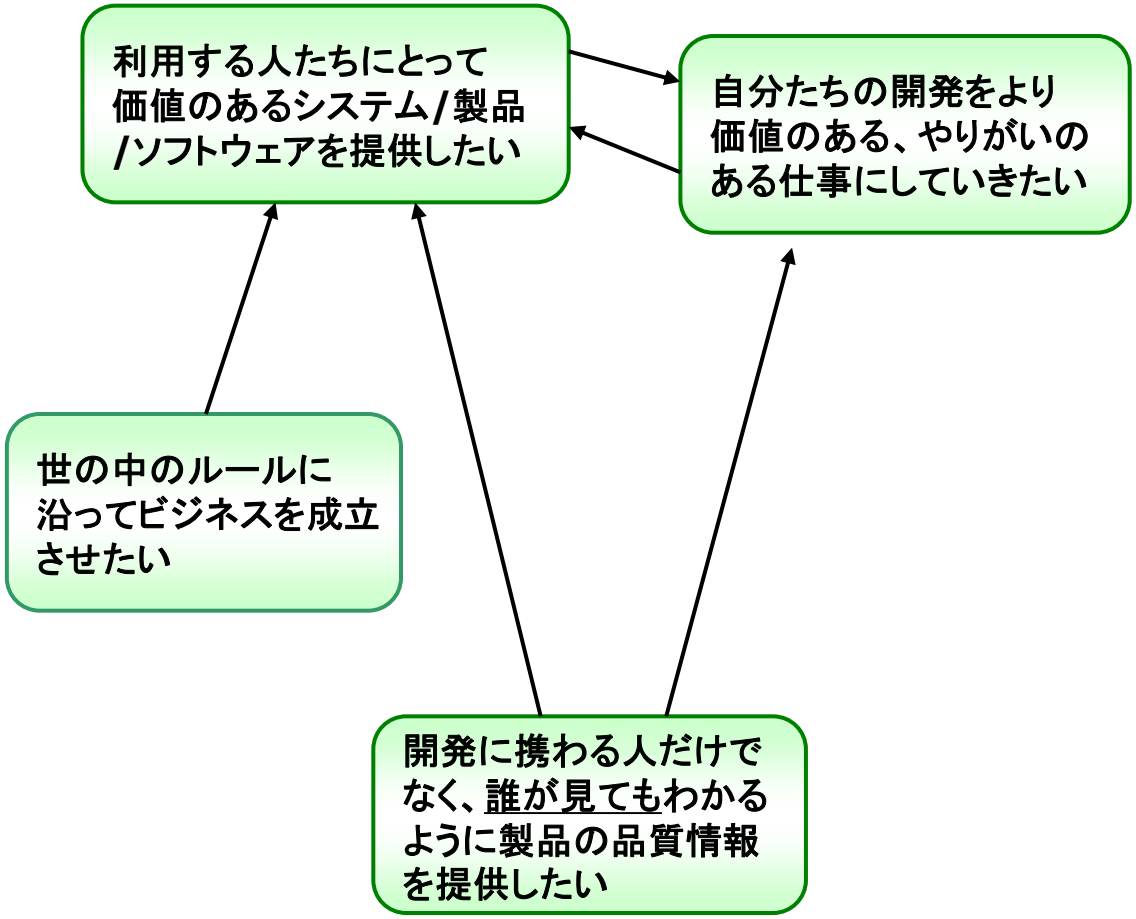
テストの目的は「テストの必要性」を実現する手段

それぞれの目的はなんで必要なんでしょう？

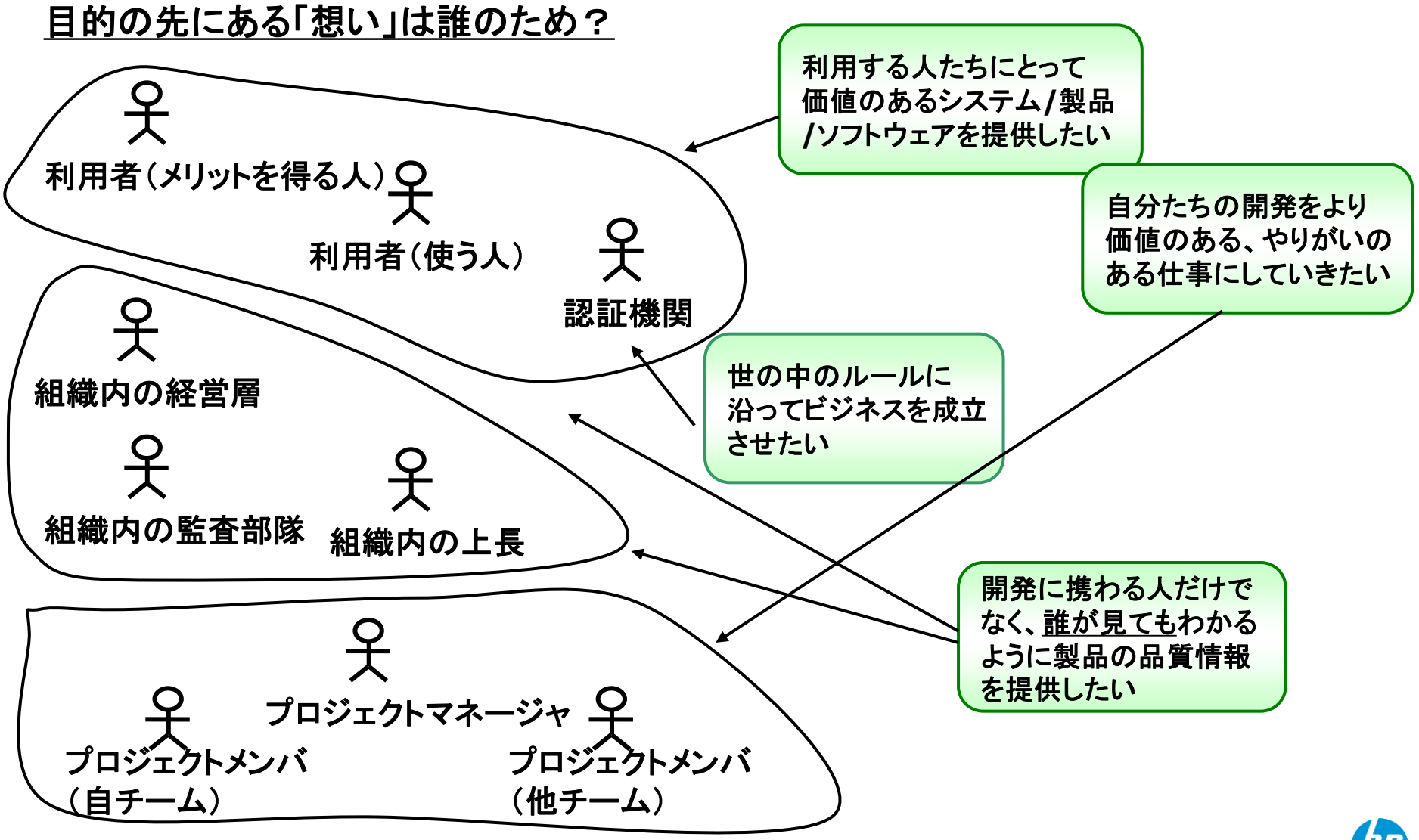


それぞれの目的はなんで必要なんでしょう？

目的の先にある「想い」にも関係がある



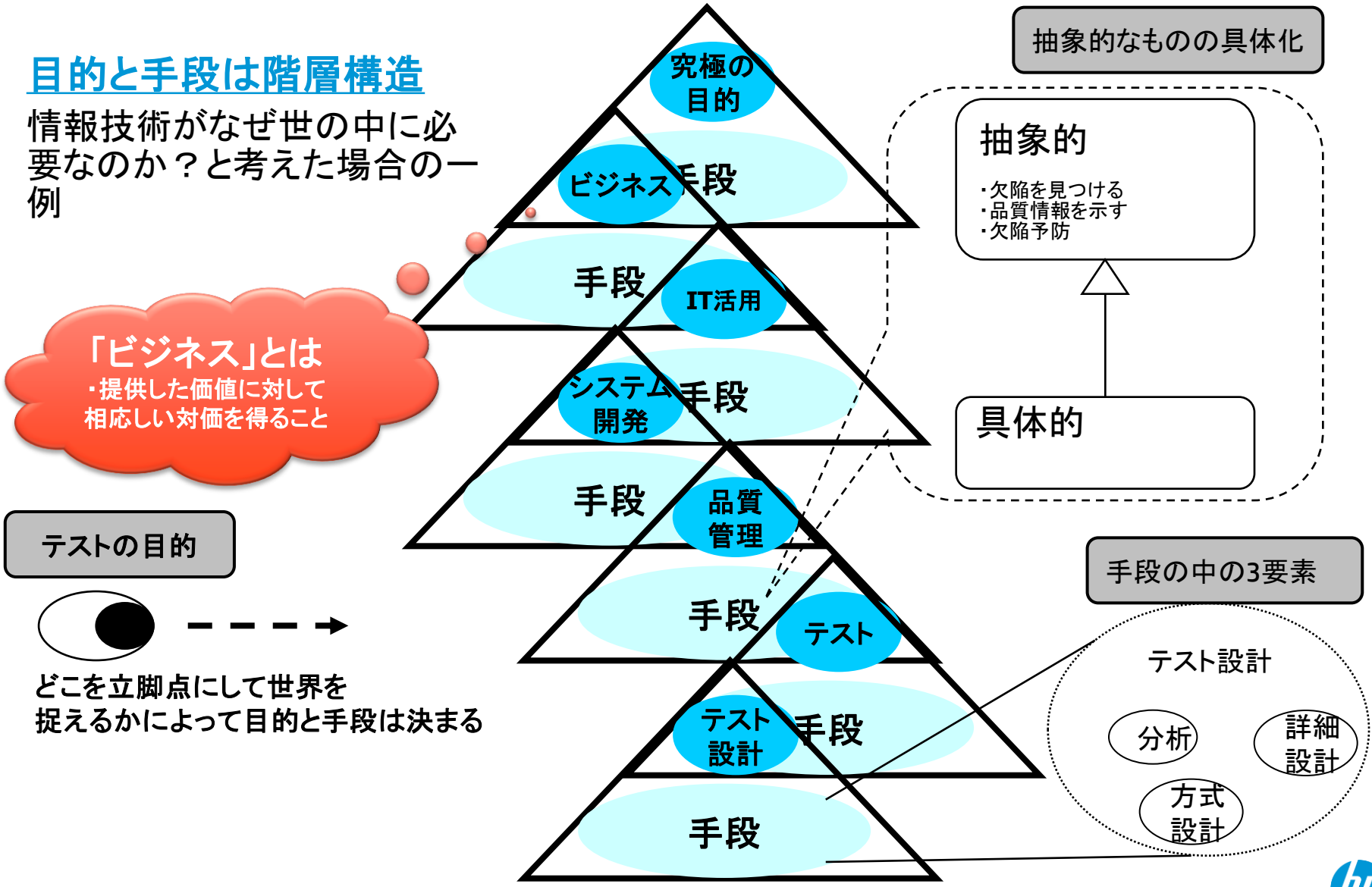
それぞれの目的はなんで必要なんでしょう？



【参考】目的の階層構造・抽象・手段の3要素

目的と手段は階層構造

情報技術がなぜ世の中に必要なのか？と考えた場合の一例



4.テストの目的とは

最適なテストケースの集合を選定する基準

そもそもテストとは

テスティングの定義

ソフトウェアテスティングとは、通常は無限に大きいと考えられる「プログラムの振る舞いの実行領域」から、最適だと考えられる有限な「テストケースの集合」を選定し、所期の通りかどうかを実際に動かして検証すること（SWEBOK 2004）

- テスティングとは単に「動かして検証する」だけの行為である。

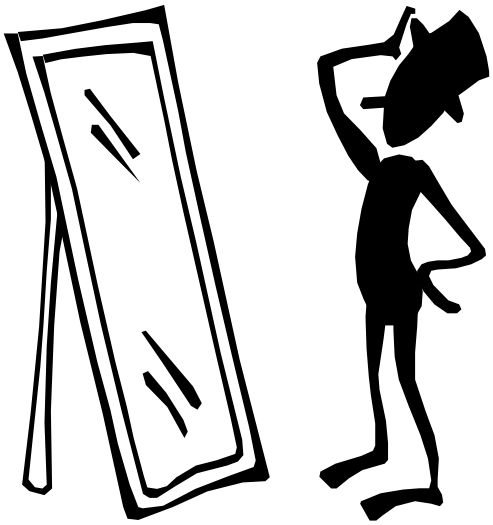
例えるなら

テストは品質を写す「鏡」、テスティングは鏡を使って知る「行為」

それなのに難しい…「何故その行為をするのか？」

- どこを見たいのか？
- 見ることで何を知りたいのか？
- 知ってどうするのか？

つまり…何をどこまでやれば良いかがわからない



ソフトウェア開発では

いつも同じものを作るわけではないし、作る必要もない

製造業はいつも同じものを作るために頑張るが、ソフトウェア開発はそうではない

- ホントに同じものはコピーできる
- 必ず何かがちょっと違うから作らないといけない

テストするのも、製造業と違い「いつも同じになっていることをテストする」わけではない

- （そういう意味で）製造業のやり方は使えない
- 品質保証に対する誤解はここからきてる

だけど、学ぶところも多いので、そこを間違えてはいけない

- 製造業が効率を上げるためにしている工夫点はソフトウェア開発への重要な示唆
  - 検証しやすく作る
  - IT化を推し進める

ソフトウェア品質の本音　第12回　[ソフトウェアテストの自動化について　小井土 亨]を参考

テストすることも毎回同じではない!!

何を見て、知りたいのか？知ったことをどうするのか？

最適とは？

テスティングの定義

ソフトウェアテスティングとは、通常は無限に大きいと考えられる「プログラムの振る舞いの実行領域」から、最適だと考えられる有限な「テストケースの集合」を選定し、所期の通りかどうかを実際に動かして検証すること（SWEBOK 2004）

どれが最適？

何を持って「最適」とするのか？

- プログラムコードを実行してエラーにならなければよい
- 仕様に書いてある通りに動けばよい
- 「いつでもどんな時でも」仕様に書いてある通りに動けばよい
- 「いつでもどんな時でも」「所定の速度で」仕様に書いてある通りに動けばよい
- 仕様に書いてある通りに動いたときに、変な動作（ex. メモリ漏えい）しなければよい
- 仕様に書いてない事も「利用シーンを想定し」上手く動いてくれればよい
- ソフトウェアを使うと当初のもくろみを達成すればよい（ex.商品売上〇%アップ）

「最適」を共有すること...「テストの目的」の共有

知らず知らずに考えている「最適」を、あらためて整理し共有する
「最適」になっていることを確認する事がテストをする目的...「テスト目的」と呼ぶ

＜テストの目的の典型例＞

欠陥の検出
- 仕様・要件に対する不具合を見つける（当たり前品質の確保）
- 意図的な欠陥識別
  - e.g. ストレス系・評価系のテスト

品質レベルが十分であることを確認
- 仕様・要件に対する合致性の確認
- 技術課題に関連して：解決策の妥当性確認・検証
- リスクに関連して：リスクの評価、対応策の妥当性確認・検証

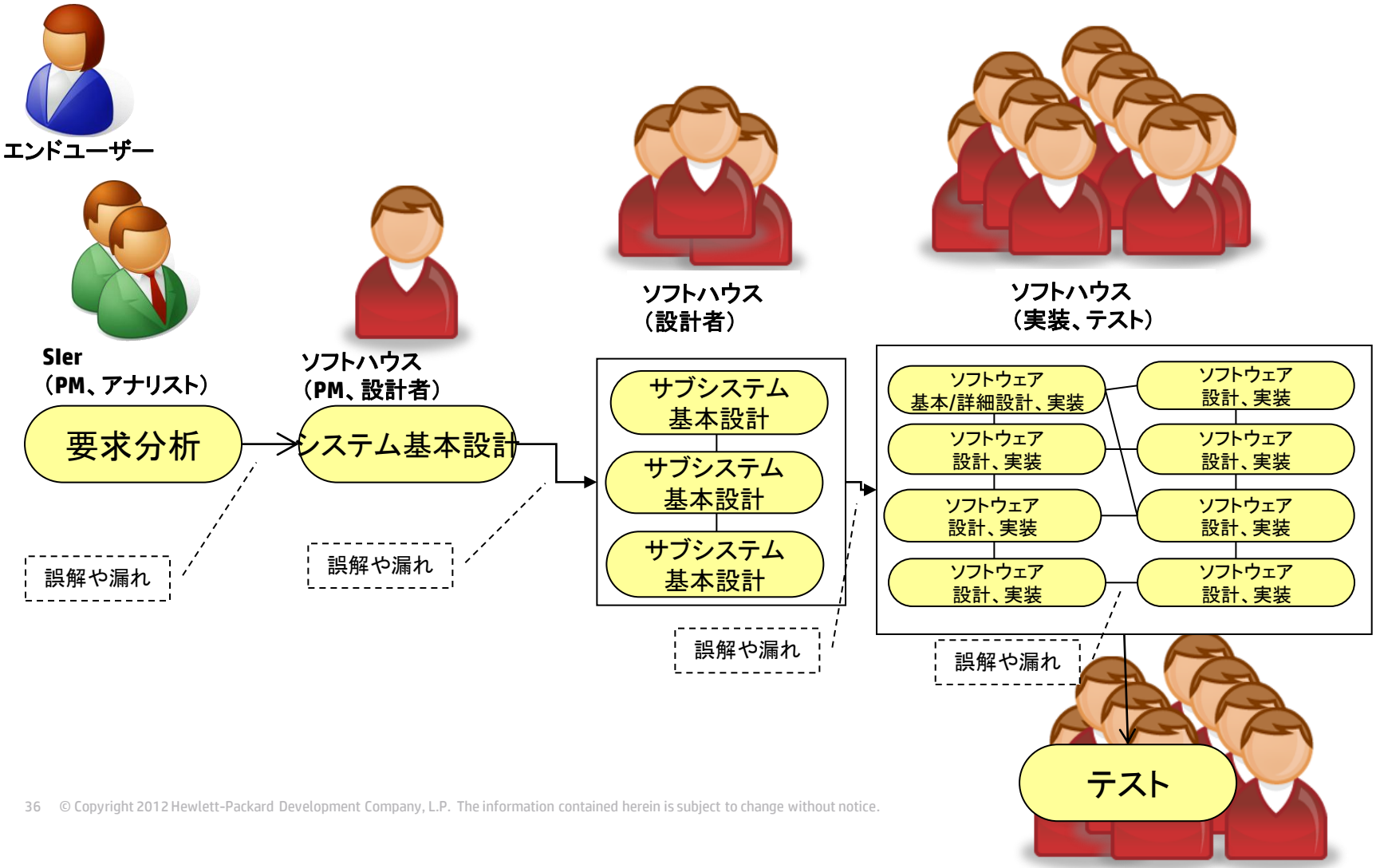
判断のための情報を提示
- 信頼度の測定
  - e.g. 運用プロフィールに沿ってランダムに生成した

早期にテスト視点で、レビューを行うことで欠陥の作りこみを防ぐ
- 要求に関連して：要求の妥当性確認、要求の抽出（の促進）
- 要件・設計に関連して：要件・設計の妥当性確認・検証

何故「共有」しないといけないか？

（テストも含め）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



現実の問題点

ソフトウェア開発の中で「共有」が出来ない

- 開発の進め方に対する考え方が違う（組織の違い/規模の違いをバックボーンにしたバラバラな考え方）
  - 外部設計、内部設計　vs 論理設計、構造設計
  - 統合テスト、システムテスト vs 結合テスト、総合テスト
- そもそも共有をしない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…目的をはっきりさせず言われたことしかしない
  - ドキュメントを作らない….他人と共有しようとしない、身近な人にしか伝えない
  - （工数がかかりすぎるため）監視とコントロールを行わない…共有したものがどうなったかは考えない

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。

- あるお客さんとの会話
  - お客さん：ウチのチェックリスト（テストケースのこと）は細かすぎるから見ていない。テストする内容は口伝えしている。
  - こちら：市場でバグが出た時のどんなテストをしていたかはどう報告するんですか？
  - お客さん：チェックリストを提出する。
  - こちら：チェックリストを使ってテストをしていないんですよね？
  - お客さん：そういうことを追求するのは本質的ではないし、調べるのも大変。時間の無駄ですよ。

共有するために大事なこと

目的を具体化する

抽象的な表現だと共有できない

目的の達成度合いをどう確認するかを決める（十分性基準）

絵に描いた餅になってしまう

全体像を提示する

意味を理解せず、ピンポイントに言われたことだけやると間違いに気づかない

目的と手段の協奏

テストの目的がその後どう使われていくかを理解しないとバラバラになってしまう

- 手段の中の3要素

マネジメント課題

- 文書化スキル
  - 文書化しないと他人と共有できない/身近な人にしか伝わらない
- テクニカルコミュニケーションスキルの教育
  - バラバラの考え方/バックボーンだと同じ言葉/文脈を別の意味に捉えてしまう
- 交渉スキルの教育
  - 合計形成の方法を知らないとまとまらない

共有するために大事なこと...具体化

テスト目的を2つの「観点」で具体的にする。

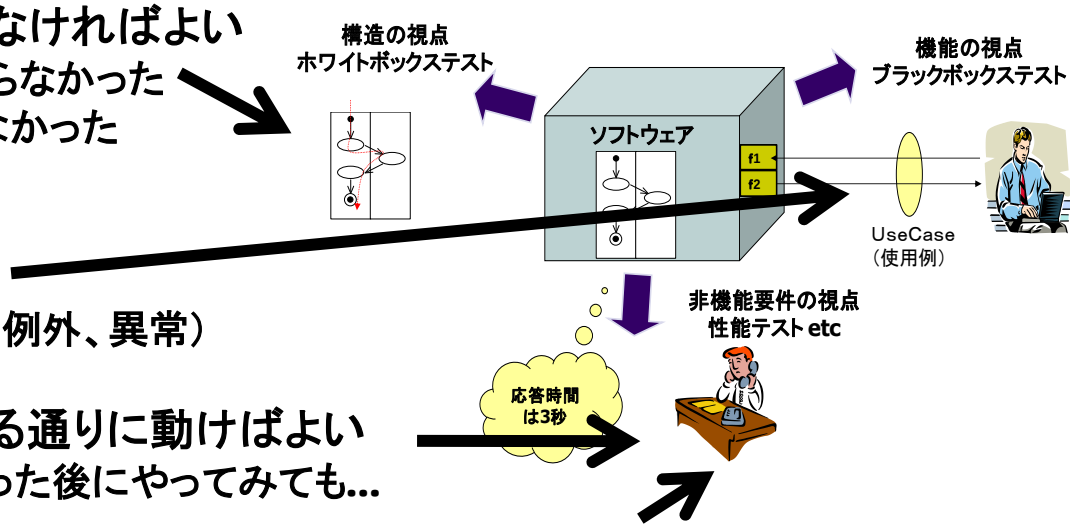
見方（網羅的、ピンポイント的）...目的のブレイクダウン
見る位置（外観、中身など）...対象のブレイクダウン

テスト目的を満たす事が「十分」であるということは...

プログラムコードを実行してエラーにならなければよい
- 全ソースコードを一回は動かしてエラーにならなかった
- 全部の分岐を一回は動かしてエラーにならなかった

仕様に書いてある通りに動けばよい
- ●●仕様書通りに動いた
- ●●仕様書に書いてないが本来書くべき事（例外、異常）

「いつでもどんな時でも」仕様に書いてある通りに動けばよい
- ●●を何回も繰り返しても、一度エラーになった後にやってみても...



「いつでもどんな時でも」「所定の速度で」仕様に書いてある通りに動けばよい
- データ量が膨大でも、、CPU負荷が高いときも...

Etc...

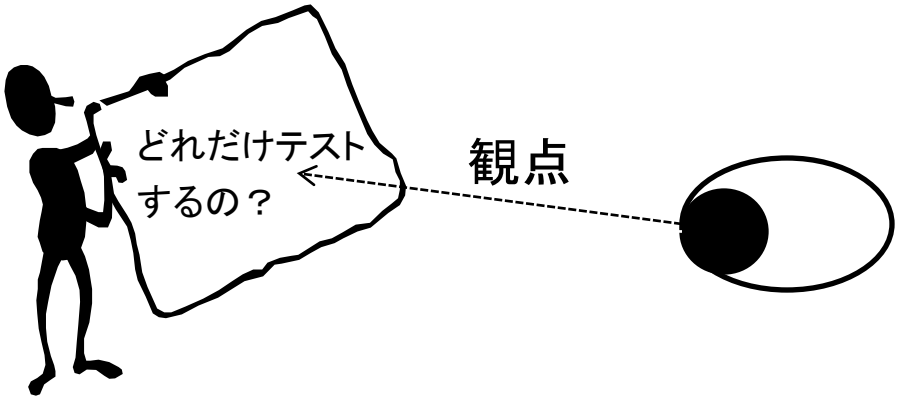
共有するために大事なこと...達成度合いの確認

達成度合い＝十分かどうか？（十分性基準）

「テストする対象」に対して「テスト目的」にそったテストをして全部合格すること

- どう測るか？はどう「目的」と「対象」を選択したかで決まる
- 目的と対象をベースに分析、設計を行うことでより具体的になる

- テスト対象
  - システム...要件定義書
  - 機能...外部仕様（画面仕様）
  - モジュール...内部仕様（プログラム仕様）



- テスト目的

＜網羅観点＞

- 構造を網羅（ホワイトボックス）
- 機能を網羅（ブラックボックス）
- 非機能を網羅

＜ピンポイント観点＞

- バグが出そう
- バグになると大問題

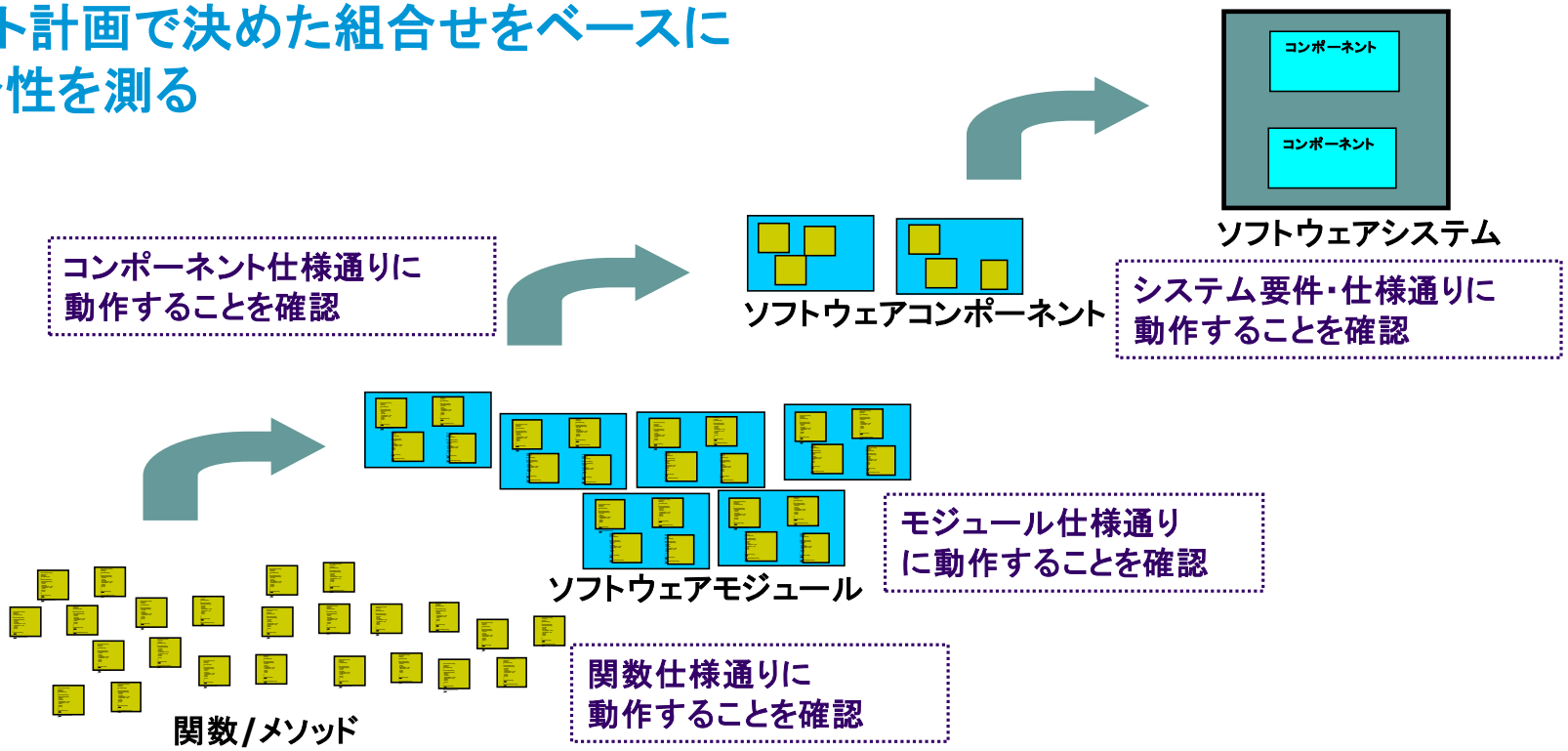
共有するために大事なこと...全体像の提示

テスト目的とテスト対象の選択と組合せを考えるのがテスト計画の根幹

複数の組合せでシステム全体がテストされたことを保証する

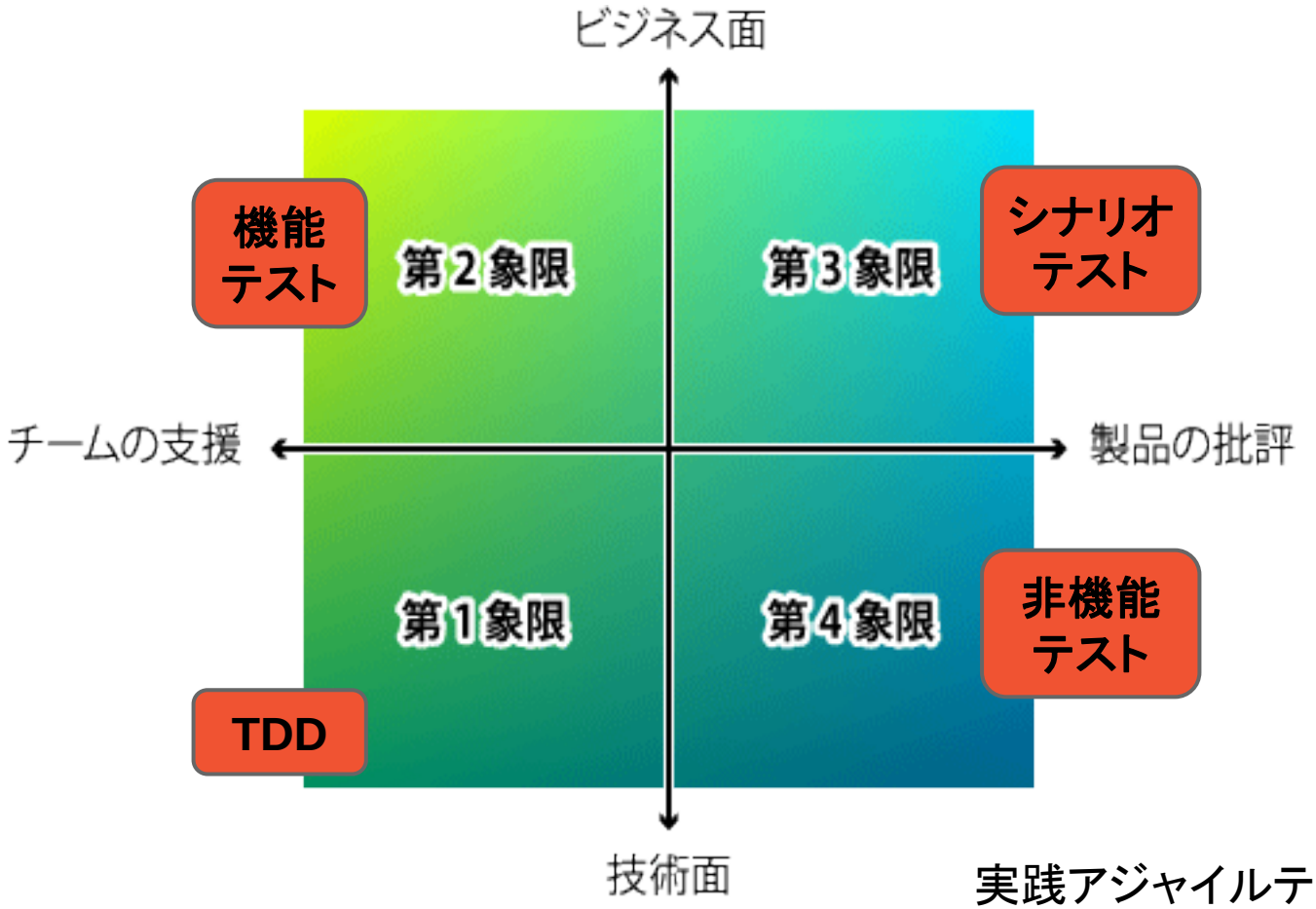
テストしきれないところはレビューも組み合わせてよい

テスト計画で決めた組合せをベースに十分性を測る



共有するために大事なこと...全体像の提示

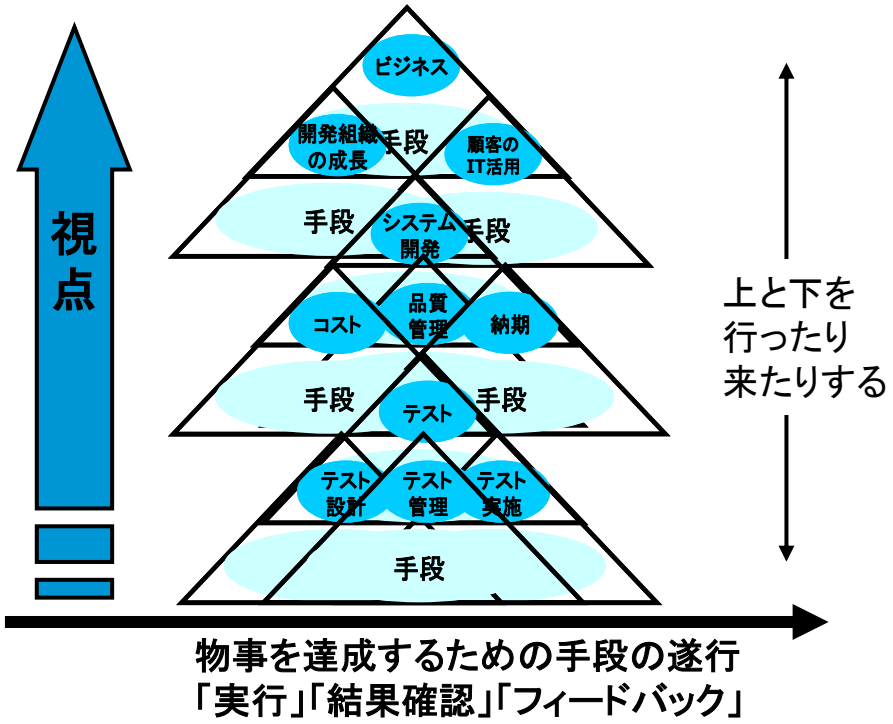
アジャイルの4象限（テスト対象と目的の組合せ）



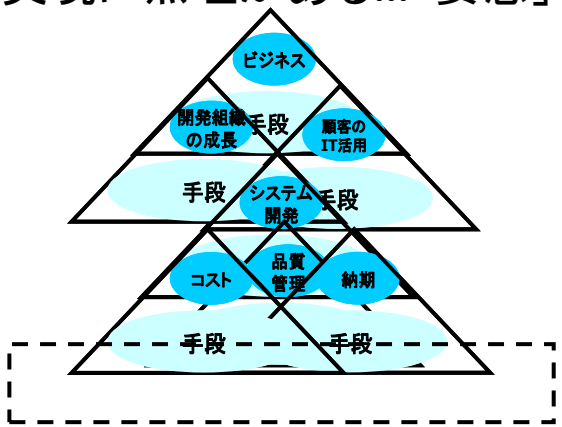
実践アジャイルテスト　より

共有するために大事なこと...目的と手段の協奏

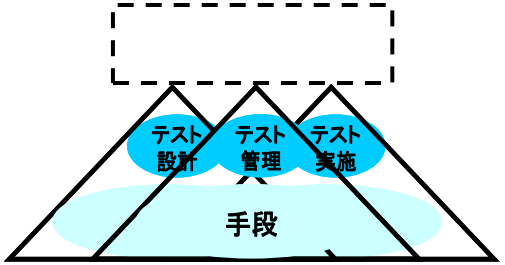
目的と手段の両方が繋がっている...「協奏」 ○



具体的な手段と繋がらないと、実現に無理がある...「妄想」 ×



向かうべき場所がないとバラバラになる...「暴走」 ×

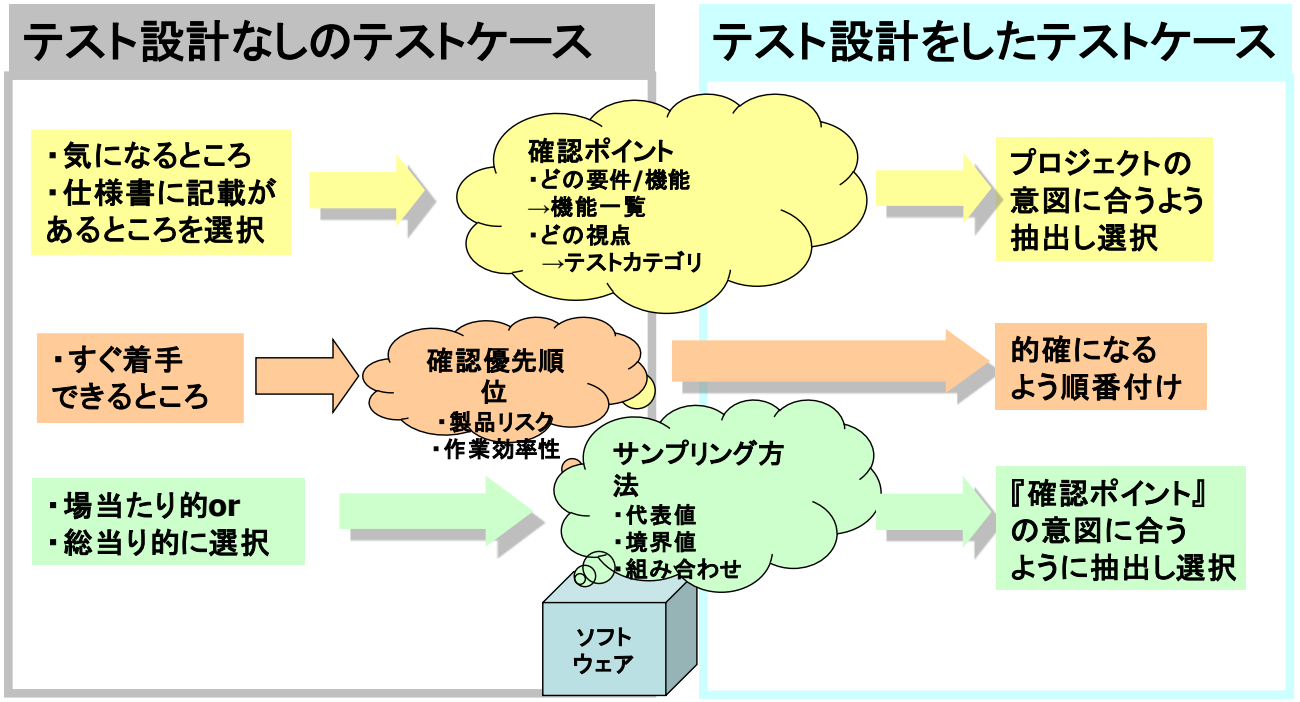


テスト設計…テスト目的を実現するための「手段」

テスト目的とテスト対象を決めたらテストが実行できるか？

- テスト対象に関することは全て仕様書に書かれているか？
- 仕様書に書かれた事全てテストしたほうがよいか？
  - 時間が足りますか？（工夫して適切な量に収めることが必要）
  - 十分ですか？（テストの目的に沿ってないといけない）

テストを設計すること＝テストの目的を満たすテストケースを造ること



テスト目的からテストケースへのつなぎ方

テスト分析・テスト設計手法の活用

JaSST'09Tokyoパネルディスカッション

「テスト技法からテストメソドロジへの進化を目指して」と題して、日本のテスト業界での第一人者が各自の考え方/実践手法を紹介

秋山 浩一（富士ゼロックス）
- HAYST法

池田 暁（日立情報通信エンジニアリング）
- マインドマップを使ったソフトウェアテスト/TAME

工藤 邦博（ベリサーブ）
- 俺流！（現場で実践している）テスト設計方法論

鈴木 三紀夫（TIS）
- マインドマップを使ったソフトウェアテスト/Tiramis

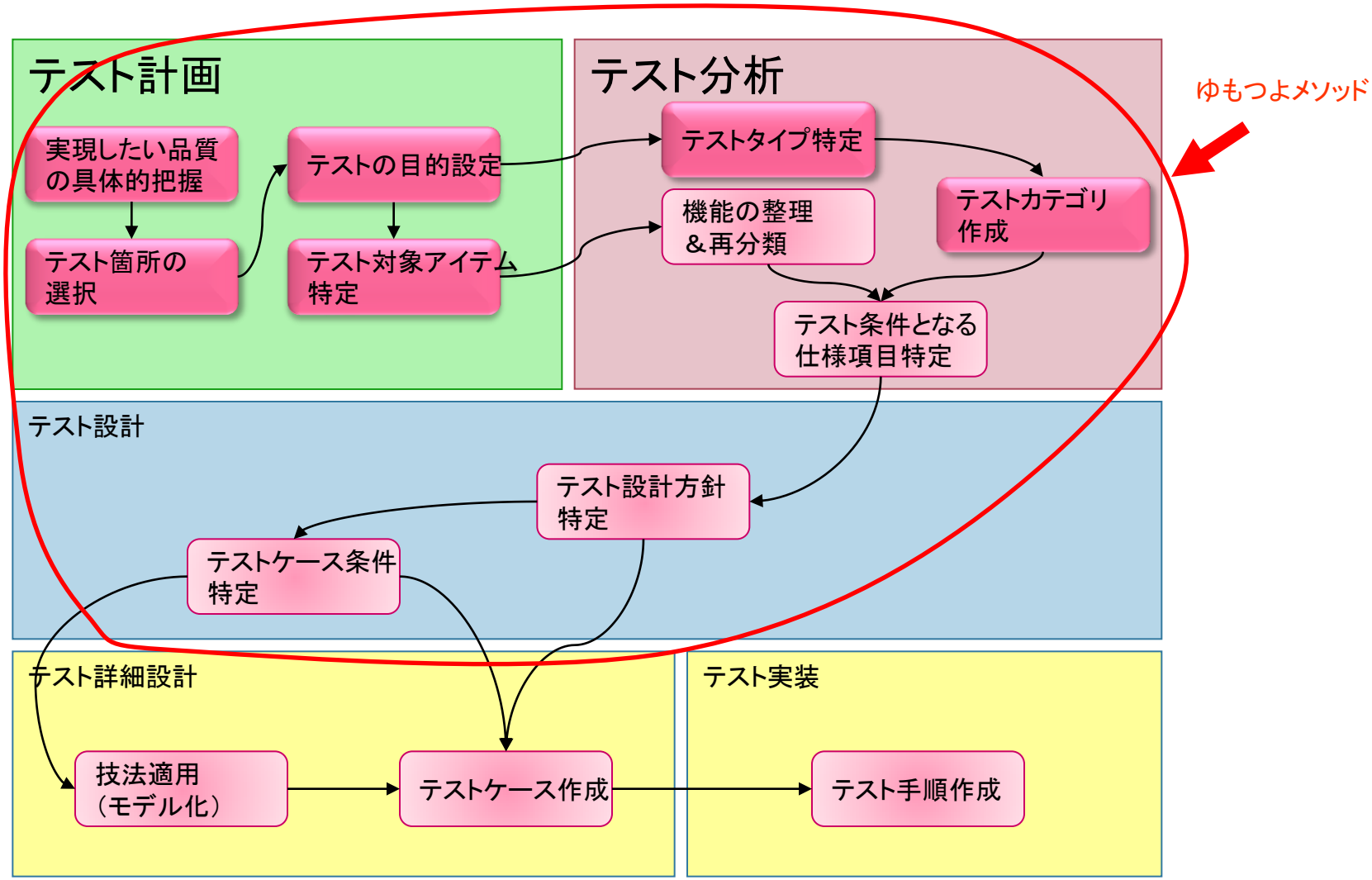
西 康晴（電気通信大学）
- VSTeP

松尾谷 徹(デバッグ工学研究所)
- CFD法（探索理論/VOP）

湯本 剛（豆蔵...現日本HP）
- 豆蔵流テスティングプロセス/ゆもつよメソッド

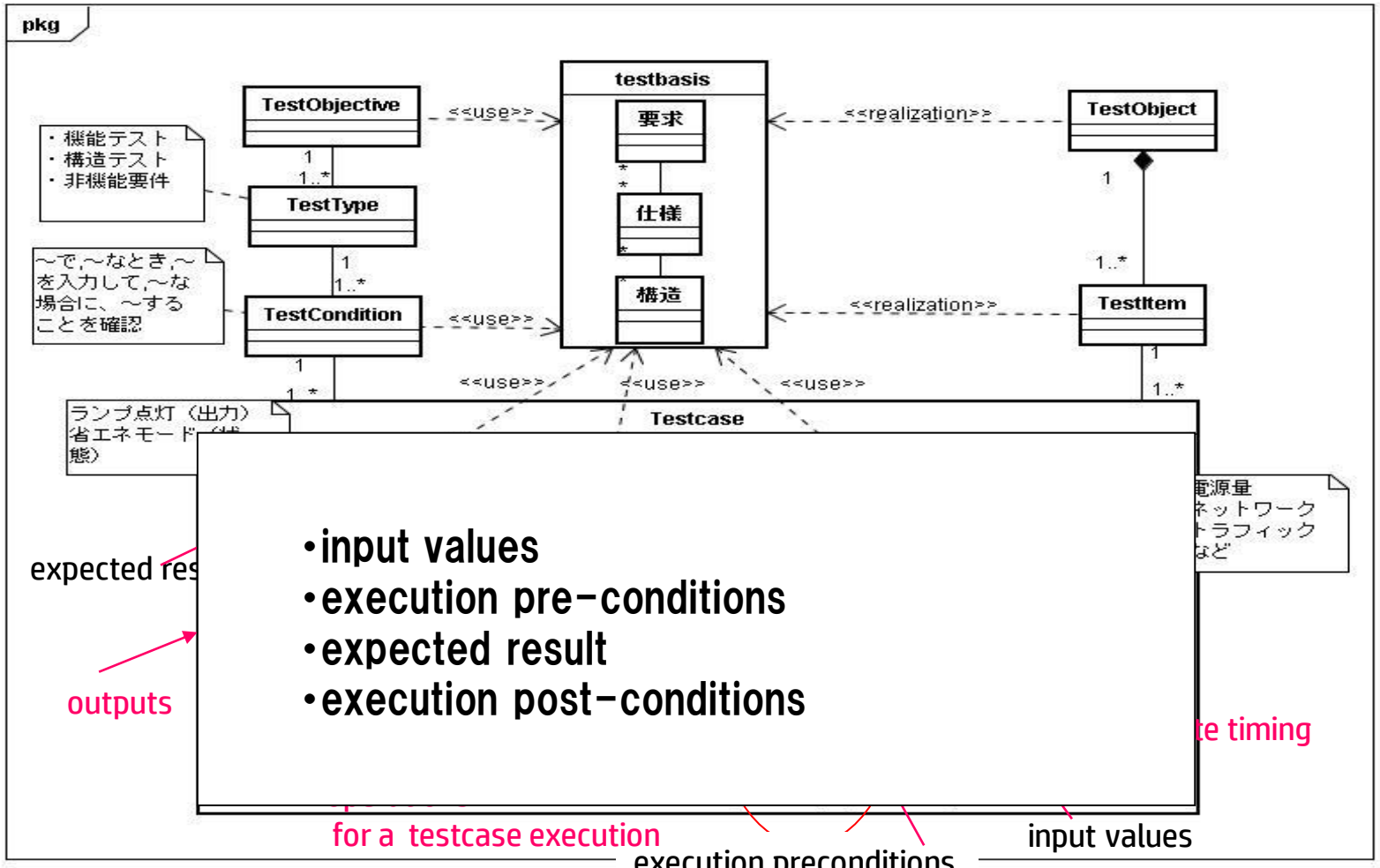
智美塾
テスト開発のメタモデルを作り
各自が独自の方法論をつくれるようにするための場
（TEFに入ると案内が流れます）

テスト開発プロセスの一例（ゆもつよメソッド）



テストの静的構造

ISTQBをベースにモデル化



まとめ

ソフトウェアテストが直面する課題

- ソフトウェアテストはソフトウェア開発のボトルネック

ソフトウェアテストの全体像

- テストに必要な活動を知る＝何をよくしていけばよいかを知る

テストは何故やるのか？

- 目的と手段の階層構造

テストの目的とは？

- 最適なテストケースの集合を選定する基準

参考文献

HP「Optimize Quality for Business Outcomes: A practical Approach to Software Testing 」
(Wiley) 2010

高橋寿一,湯本剛「現場の仕事がバリバリ進むソフトウェアテスト手法」
(技術評論社)2006

湯本剛「ソフトウェアテストPressVol.10」 特集1　今こそ聞きたいテストの上流設計
（技術評論社）2010

豆蔵 「ソフトウェアエンジニアリング講座　テスティング基本編」
（豆蔵）

Janet Gregory , Lisa Crispin「実践アジャイルテスト」
（翔泳社）2009

IEEE Computer Society「SWEBOKソフトウェアエンジニアリング知識体系」
(オーム社)2005

大村 朔平「一般システムの現象学」
(技報堂出版)2005

ISTQB「ISTQBテスト技術者資格制度Foundation Levelシラバス日本語版 Version2007.J01」
(NPO法人ASTER ,JSTQB)2007

IPA「ソフトウェア産業の実態把握に関する調査」2011年度版
（独立行政法人情報処理推進機構）2012

小井土 亨 「ソフトウェア品質の本音　第12回　ソフトウェアテストの自動化について」
(SQiP:http://www.juse-sqip.jp/wp3/honne/backnumber_012/) 2011

Thank you