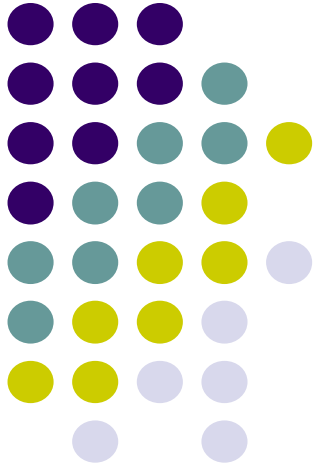
現場の力を
メキメキ引き出す
テスト戦略

JaSST'10 Hokkaido
1 October 2010
9:40 ～11:20



湯本　剛(日本HP)
tsuyoshi.yumoto@hp.com

自己紹介

湯本　剛

Consultant at HP Japan

- 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
- テストに関する書籍、Web記事や雑誌への執筆や翻訳多数

- 外部活動
  - NPO法人ASTER　理事
    - JSTQB 技術委員
    - ISTQB　CTAL WGメンバー
    - 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

- Twitter http://twitter.com/yumotsuyo

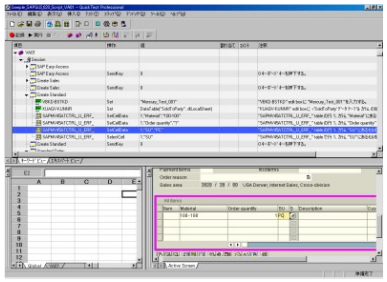
- 得意分野
  - テストマネジメント、テスト分析

HP Softwareのテスト系主要製品

キャプチャリプレイツール

QuickTestProfessional

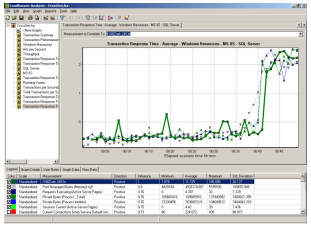
①アプリケーション操作を記録（スクリプト化）
②スクリプトを再生し、自動検証
データ駆動、キーワード駆動テスト



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

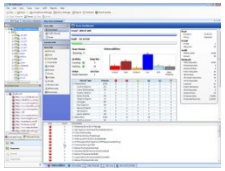


データベース

セキュリティテストツール

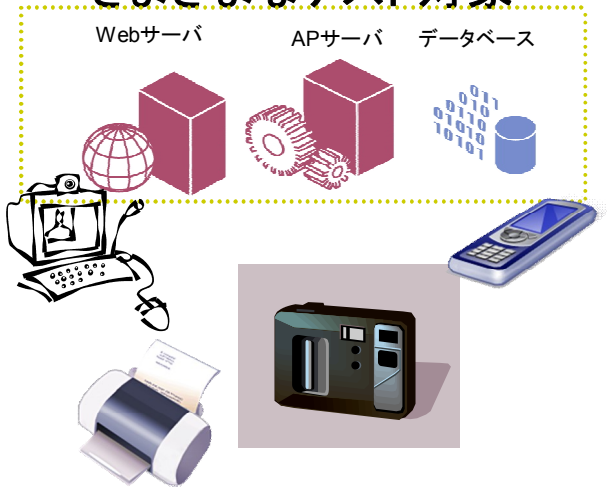
WebInspect

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



さまざまなテスト対象

Webサーバ　APサーバ　データベース

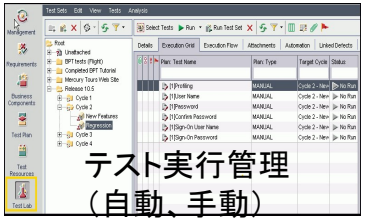


テストマネジメントツール

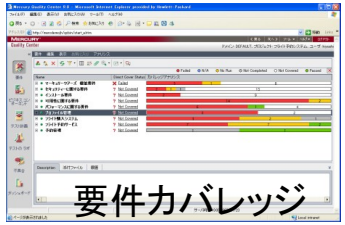
QualityCenter

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

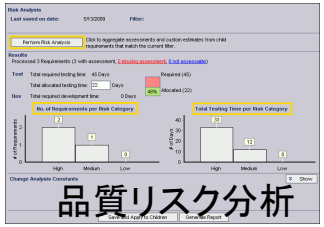
データベース



テスト実行管理
（自動、手動）



要件カバレッジ



品質リスク分析

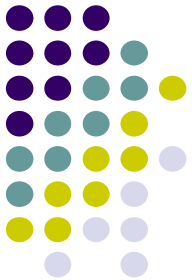
サマリ

- テスト戦略を立てるときに“人員が足りない”とか“納期が短い”ということを出発点にしていませんか？

- 自分たちの力を最大限発揮するのに大事なことは、二つ
  - 本当にやりたいこと、やるべきことをテスト戦略に反映する
    - 基本を踏まえたうえで応用する
  - テスト戦略をテスト設計へつなぐ
    - 「つなぐもの」の関係を整理し、ぶれないようにする

- 考える順番が大事
  - 最初にやるべきことをはっきりさせ、その後どう実現するか考える
    - 向かう場所があるからメキメキ引き出すことができる

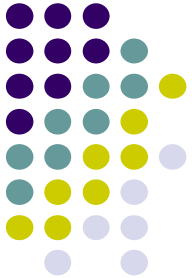
アジェンダ



- テスト戦略とは？
  - いろいろな「テスト戦略」を整理
    - プロダクト・プロジェクト・組織の分析
    - テストポリシー
    - 戦略の基本パターンとトレンド
    - 戦略オプション
    - 基本戦略
    - 個別戦略
- テスト戦略立案の進め方
  - テスト計画を進めるときの作業フロー
  - 全体テスト計画でのアプローチ作成までのステップ
  - 個別テスト計画の考え方のポイント
  - テスト設計と繋げる

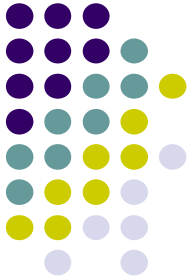
テスト戦略って何ですか？

テスト戦略って何ですか？



いろいろある

一般的に、戦略とは



- 勝つため（目的達成）のためのノウハウ
  →法則の取捨選択
  - 戦略的かどうかは、目的達成のため、意図的な取捨選択をしているかどうか
  - 「テスト戦略」は、テストをどう行なえば最も効率的で効果が高いテスト
    が出来るかを示すもの

e.g.ランチェスター理論の二つの法則
強者の戦略→空爆のような「確率戦」
弱者の戦略→「一騎打ち」

ソフトウェアテスト293の鉄則では…

「テスト計画とは、

『戦略』『段取り』『成果物』の三位一体と心得よ」

- 戦略：重要な問題を早く発見するために，製品のどの範囲をテストするのか。どこを重点的にテストするのか。どのテスト技法を用いるのか。バグが出たことをどうやって認識するのか。どのようにテストの目的を達成するのか。
- 段取り：テスト戦略を実現するために，どのようにリソースを適用するのか。誰がテストするのか。いつテストするのか。成功させるためには何が必要なのか。
- 成果物：テストの成果を，どのようにして顧客に見せるのか。どのようにバグを管理　するのか。どんなドキュメントを作成するのか。どんな報告書を作成するのか。

最良の方法はどれか

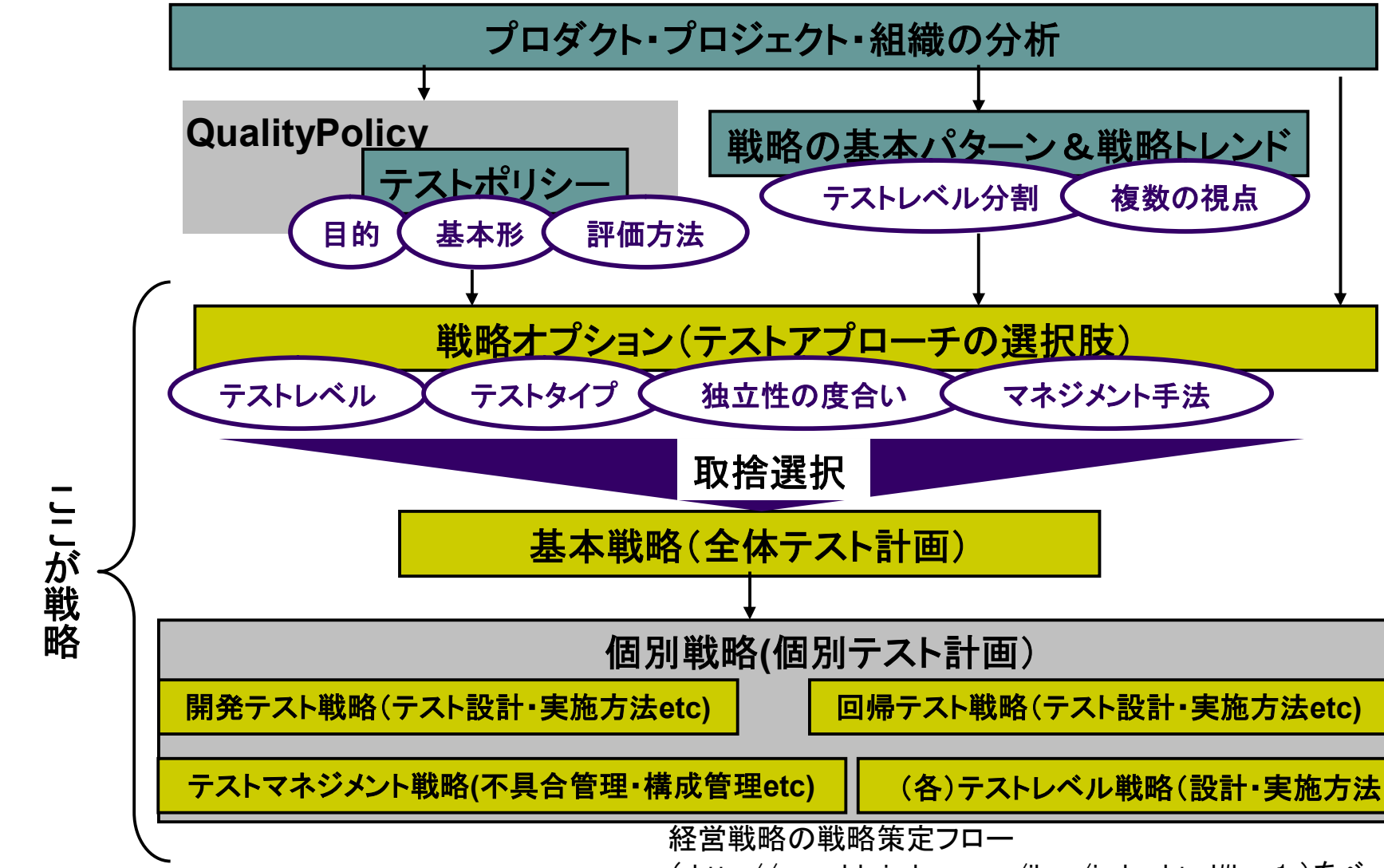
どう実現するか

実現結果をどう伝えるか

ISTQB(Advanced Level)では…

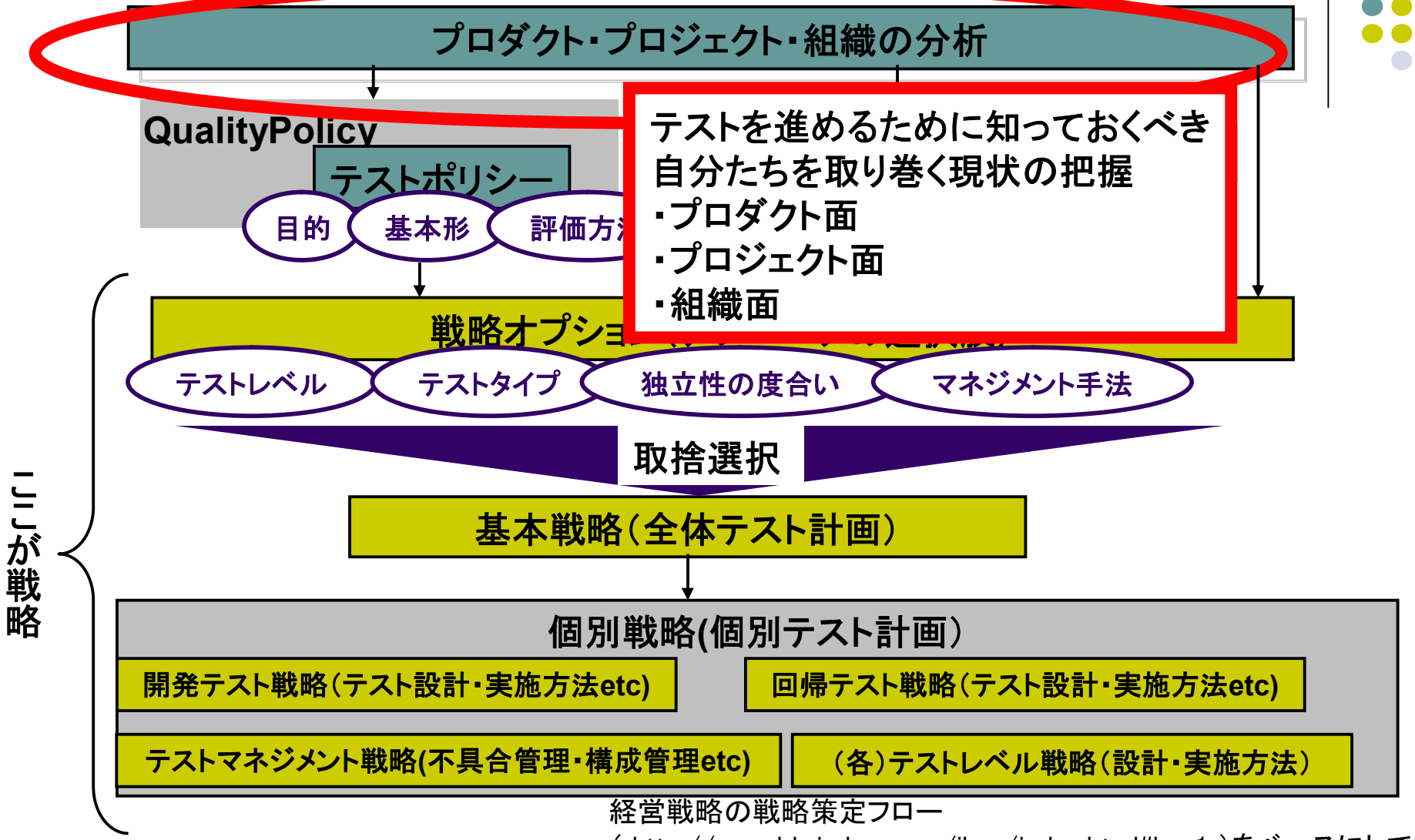
- テストマネジメント関連ドキュメントは以下の4種類
  - テストポリシー
    - 組織やプロジェクトにとってのテスト/品質保証の理念を示す
    - テストポリシーは品質ポリシーのコンポーネント、または補完的なものとなる
  - テスト戦略
    - 組織にとってのテスト方法論を示す
  - 全体テスト計画
    - プロジェクトに適用したテスト戦略を示す
  - (各)レベルテスト計画
    - 各テストレベルで実行する活動
    - 全体テスト計画内容を各テストレベルに落とし込んだもの

いろいろなテスト戦略を整理すると



経営戦略の戦略策定フロー
（ http://www.blwisdom.com/ikey/index.html#key1 ）をベースにしています

プロダクト・プロジェクト・組織の分析

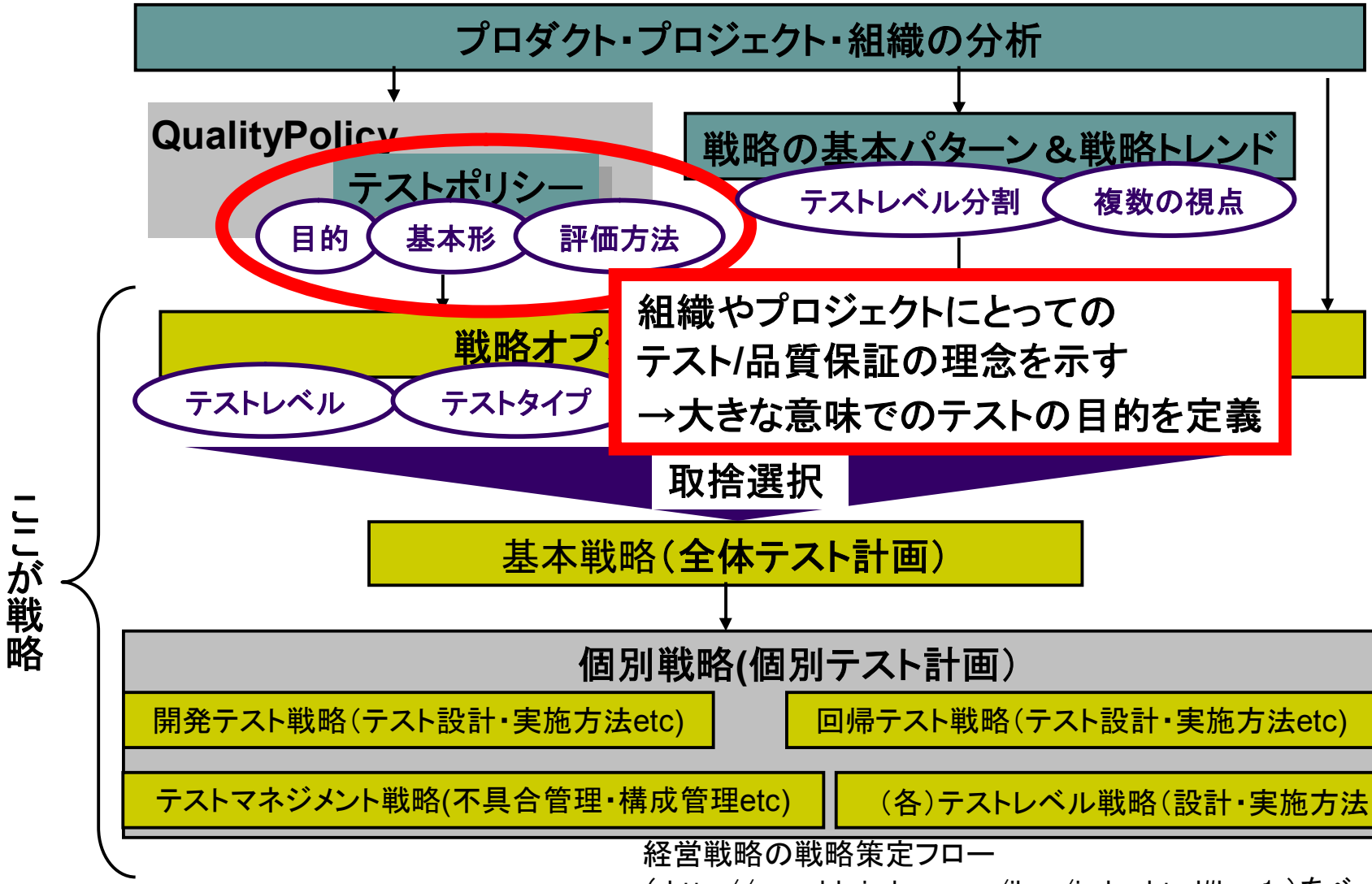


経営戦略の戦略策定フロー
（ http://www.blwisdom.com/ikey/index.html#key1 ）をベースにしています

プロダクト・プロジェクト・組織の分析

- プロダクトについての分析
  - 開発目的
    - 開発組織自身のねらい
  - 顧客の要求とシステム化範囲（システム要件）
    - 製品の「売り」/製品　or　システムが貢献できること
  - ターゲットユーザ/業務/製品ドメイン
  - プロダクト完成後の計画
    - 出荷日程/運用開始日や販売価格など
  - 技術課題
- プロジェクトについての分析（強み・弱み）
  - 選択した開発プロセス
    - 開発プロセスの中のテストの位置づけ（テストポリシーに対する考慮）
  - 選択した技術（言語やアーキテクチャパターン）
  - 開発プロセス、技術に対する経験　（テストフェーズ含む）
  - メンバー構成（自社のみ、協力会社含む、オフショアetc)
- 組織目標に対する分析
  - 組織的な開発方針・目標（e.g.開発コストを前年より削減させる)

テストポリシー

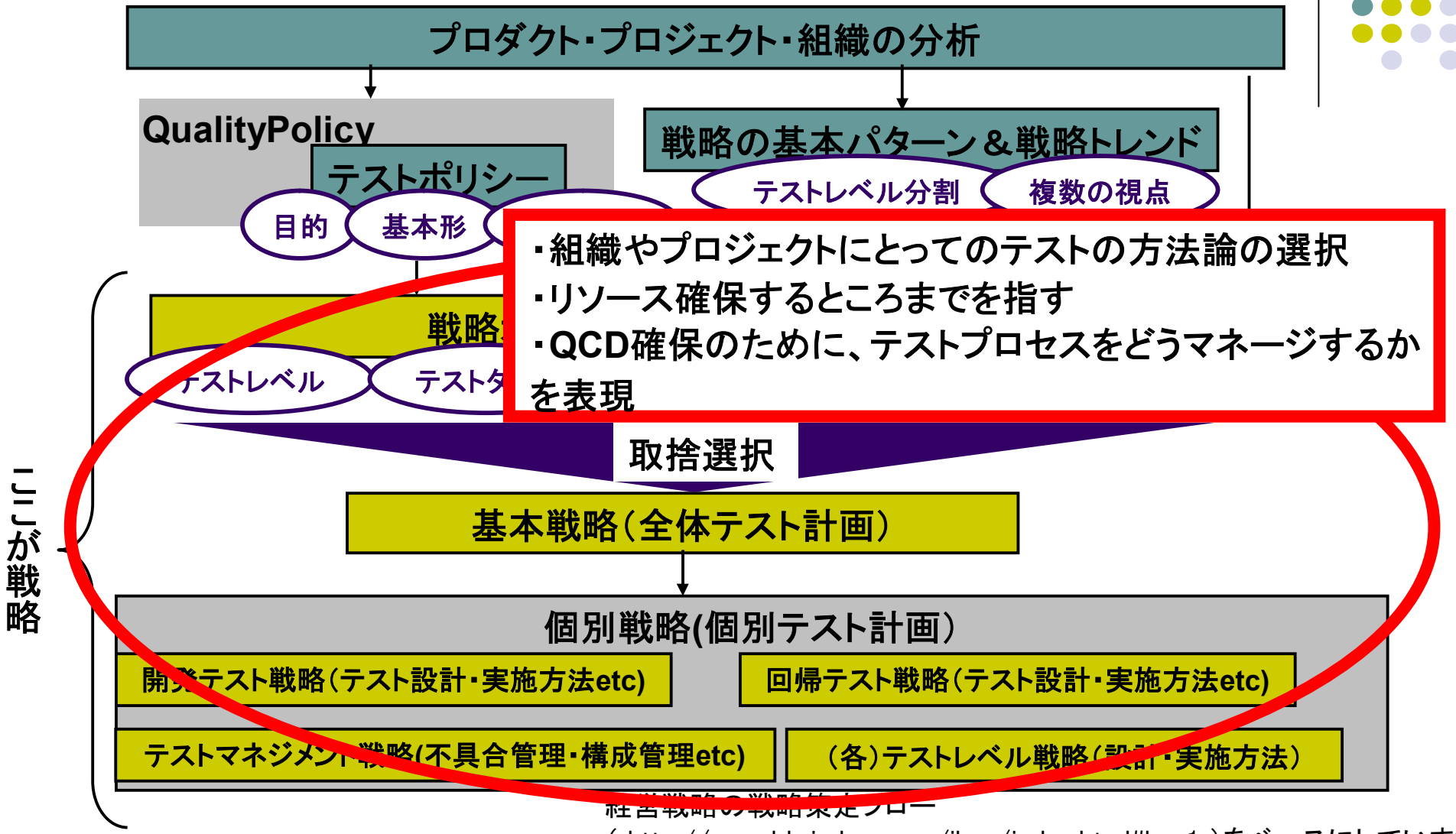


経営戦略の戦略策定フロー
（ http://www.blwisdom.com/ikey/index.html#key1 ）をベースにしています

テストポリシー（from ISTQB AL）

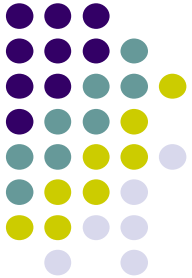
- テストポリシーは品質ポリシーのコンポーネント、または補完的なもの
- テストポリシーに含まれる内容
  - テストの定義を与える　ここが大きな意味でのテストの目的！
    - e.g.システムが目的どおり動く自信を持つ、（より低コストで早期に重大な）欠陥を見つける
  - 基礎となるテストプロセスをレイアウトする
    - e.g. 計画、分析、設計、実装、実行、報告、終了作業
  - テストの有効性と効果の評価方法を示す
    - e.g. プロジェクトコストに対するメリット、市場での失敗コスト、納期短縮への貢献などの計測方法
  - 品質を計る方法を示す
    - e.g.市場リリース後の故障率やソースコードに対する欠陥密度
  - テストプロセス改善の活動を示す
    - e.g.改善モデル（TPIやTMM）、「振り返り」からのフィードバックによる改善

テスト戦略



（ http://www.blwisdom.com/ikey/index.html#key1 ）をベースにしています

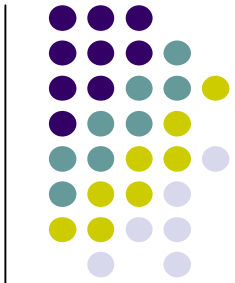
テスト戦略（from ISTQB AL）



- 組織のニーズやポリシーをもとに戦略（基本パターン）を選択、もしくは組み合わせたものを示す
  - （基本パターンは）テスト設計の開始時期という視点で分類可能
    - 予防的（段階的に実施） or 対処的（最終段階で実施）
  - 典型的な戦略（基本パターン）の例
    - 分析的、モデルベース、方法論的、スタンダード準拠型、経験則的、ユーザ先導型
- 実行するテストレベルの定義を示す
  - テストレベルの開始・終了基準
  - テストレベル間の関連や調整
- その他、以下に関するアプローチも示される
  - 統合手順（Integration procedures）
  - テスト設計技法
  - レベル毎の独立性
  - 標準（必須のものと任意のもの）
  - テスト環境
  - テスト自動化
  - 成果物の再利用性
  - 確認テストとリグレッションテスト
  - テストコントロールと報告
  - テスト関連メトリクス
  - インシデントマネジメント
  - テストプロセス改善活動

・ISTQBでは戦略≒アプローチと定義している
・ここでは、本テキスト全体の流れを考慮し、（基本パターン）と付け加えている

[補足]アプローチとは？

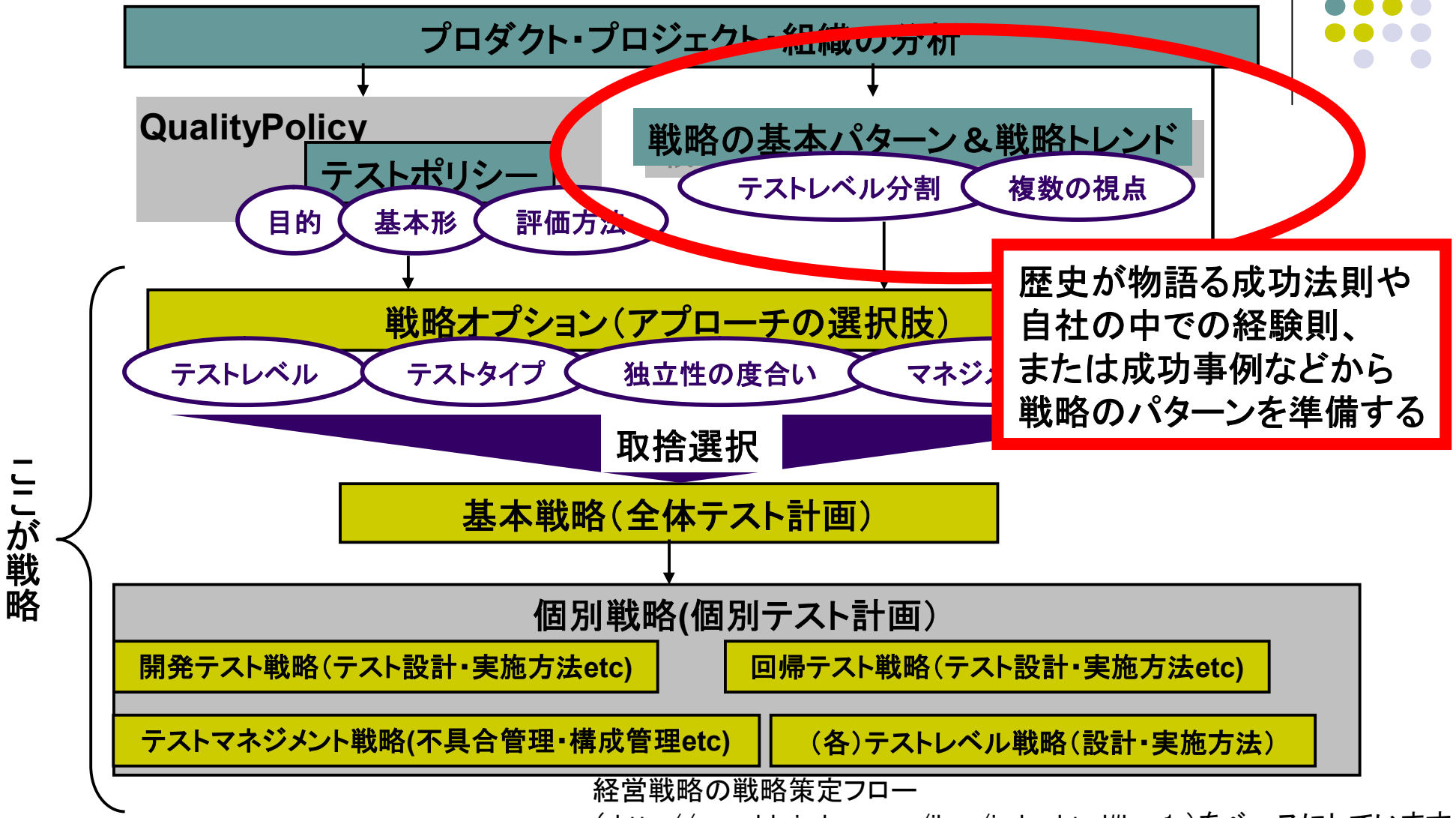


- Approach[名]
  - やり方、扱い方、取り上げ方、姿勢、取り組み、手引、学習法、研究方法、取り掛かる方法、入門（法）

    （アルク　英辞郎　http://eow.alc.co.jp/）
  - アプローチとは、単に「やり方」と表す一般用語

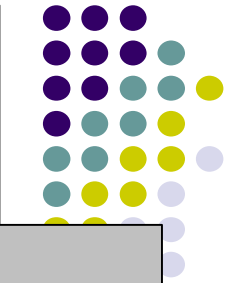
- IEEE829では、「アプローチ」と書かれている章に戦略を具現化した「やり方」が書かれる
  - 戦略＝パターン、アプローチ＝パターンをプロジェクトに実装した時の「やり方」

戦略の基本パターン＆戦略トレンド



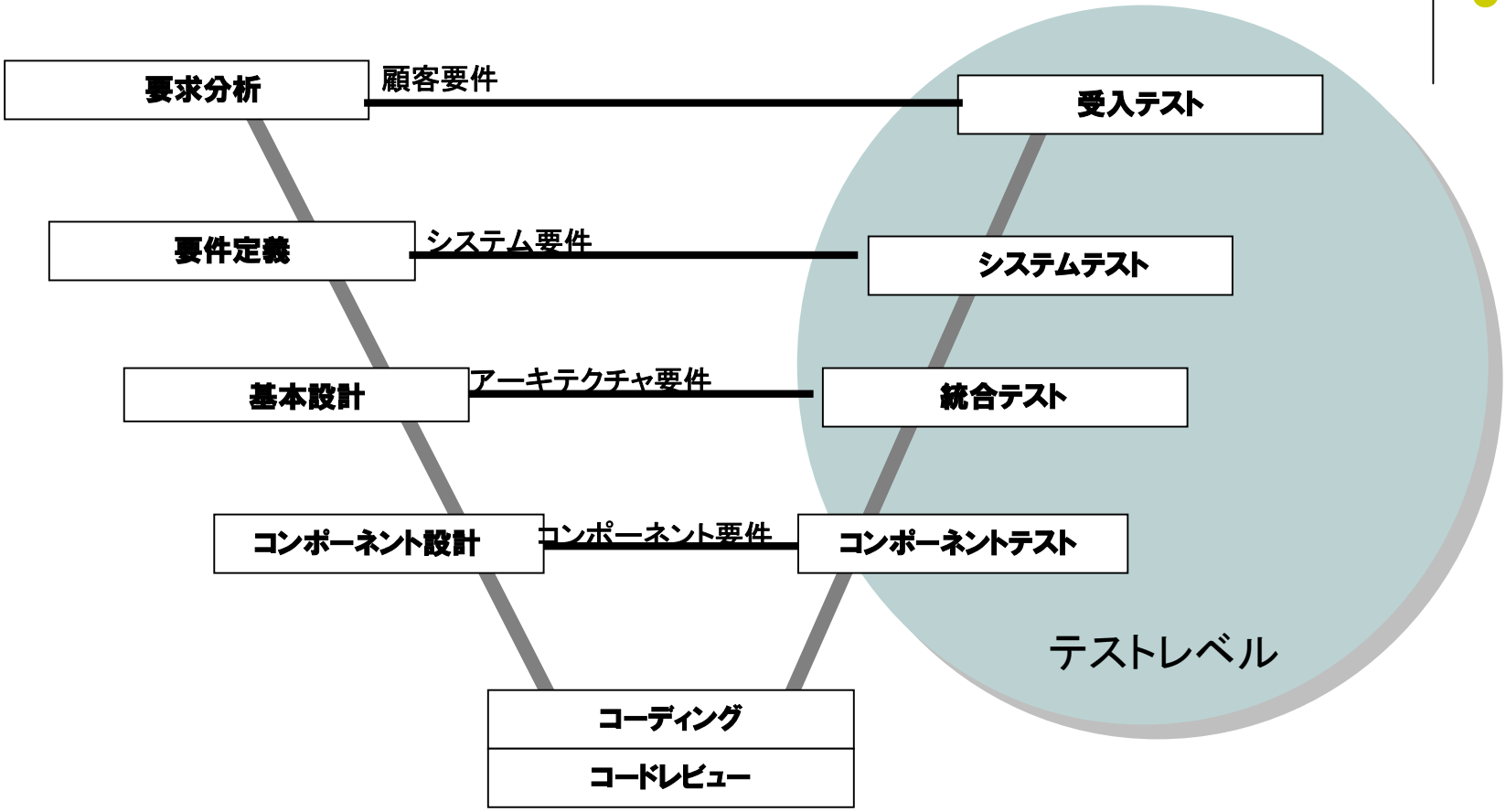
経営戦略の戦略策定フロー
（ http://www.blwisdom.com/ikey/index.html#key1 ）をベースにしています

テスト戦略の基本パターン



1. テストレベルを分けて段階的にテストを行う
2. 複数の視点を使い分ける(テストタイプ/テスト技法)
3. (テスト前に)レビューを実施する
4. 開発した本人以外がテストを行う

1. テストレベルを分けて段階的にテストをする



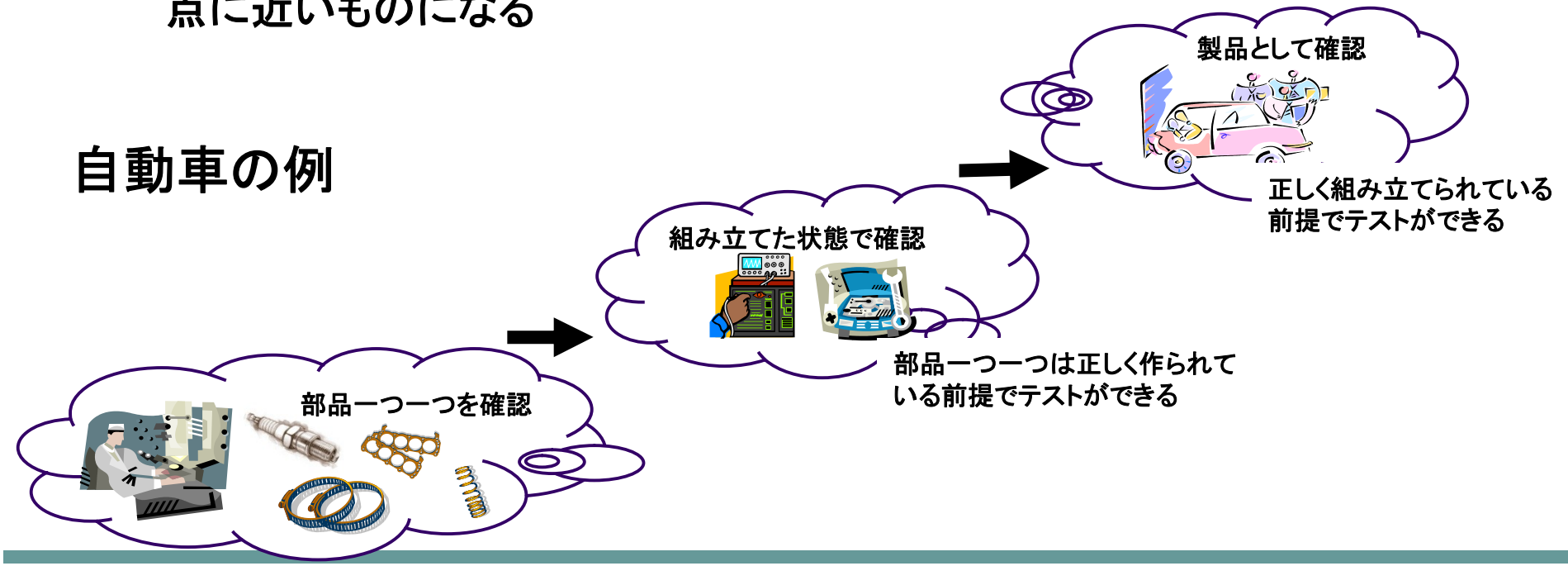
V字モデル

（ウオータフォール開発でのテスト配置のモデル）

[補足]テストレベルとは(1)

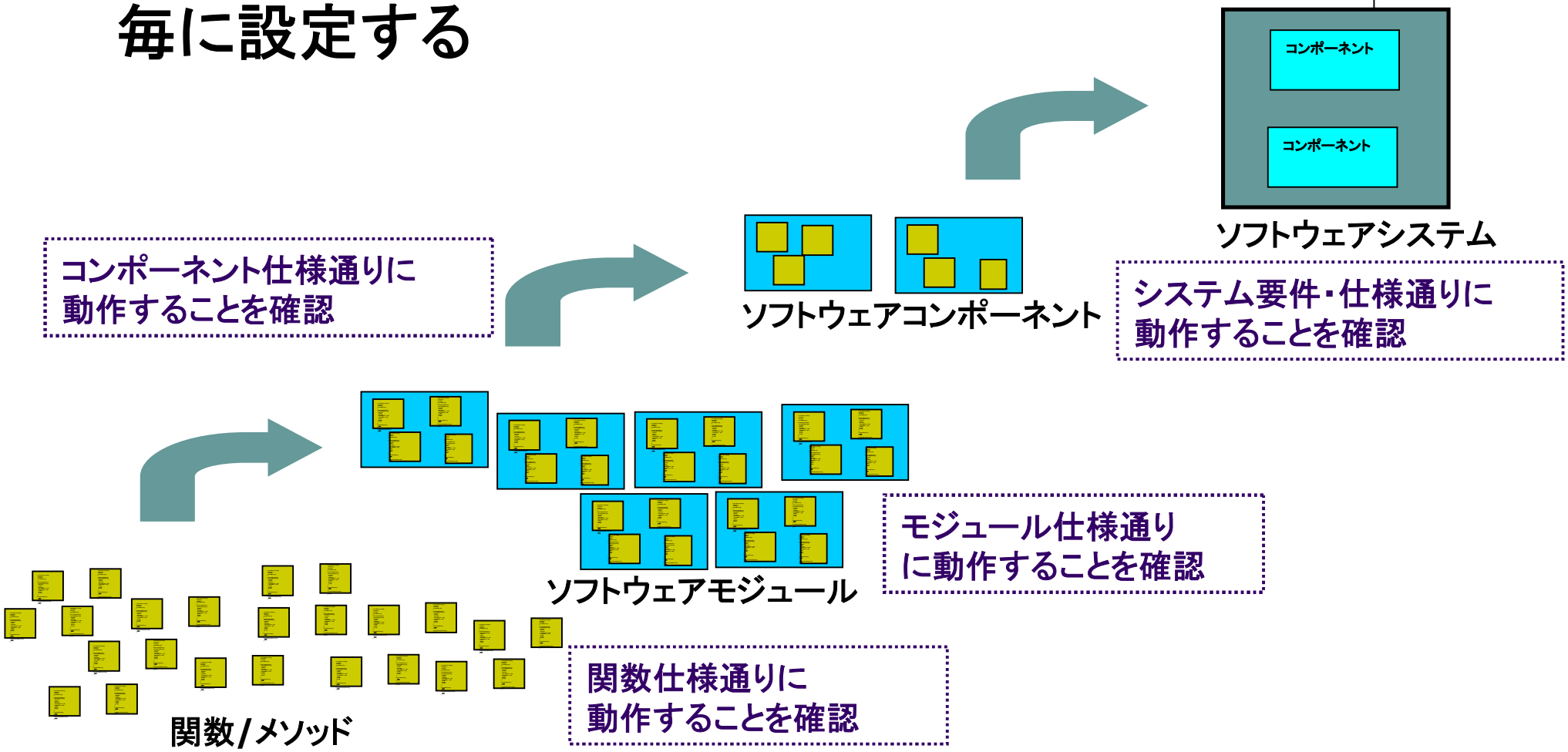
- テスト対象になるソフトウェアの詳細度合いに応じたテストの分類のこと
- テストレベルの設定
  - 複雑な対象をテストする場合、部品(構成要素)から確認し、その後、徐々に部品を組み合わせたものを確認し、最終的にシステム・製品全体を確認していく方法を取る
  - 確認内容も、詳細な部品レベルを確認する視点から、よりシステム利用者の視点に近いものになる

自動車の例



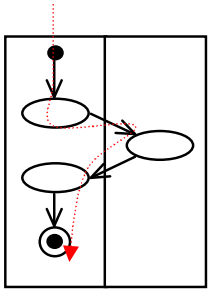
[補足]テストレベルとは(2)

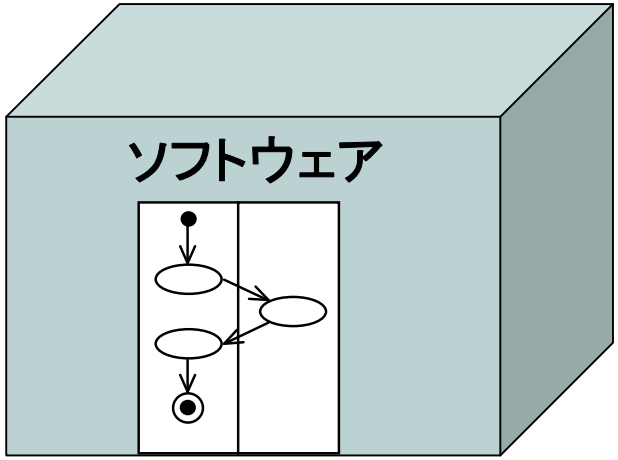
- ソフトウェア開発の場合、テストレベルは各要件・仕様毎に設定する



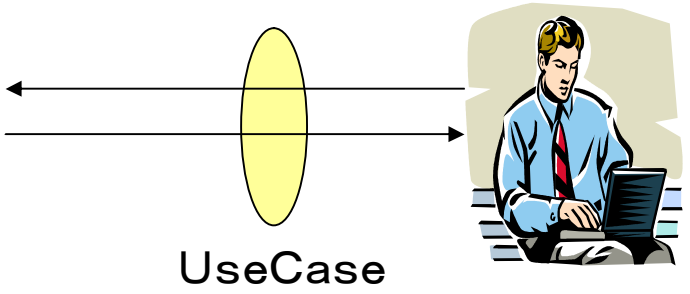
2. 複数の視点を使い分ける(テストタイプ選択)

構造の視点
ホワイトボックステスト





機能の視点
ブラックボックステスト

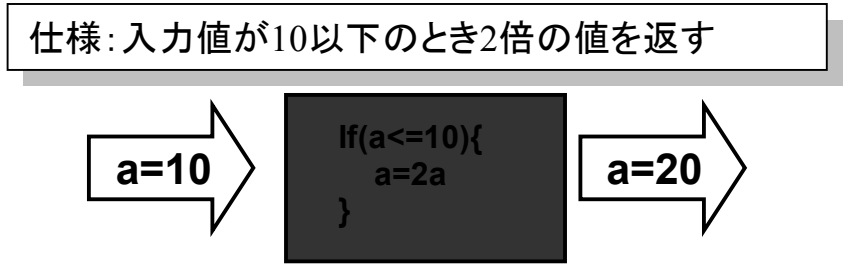


非機能要件の視点
性能テスト etc

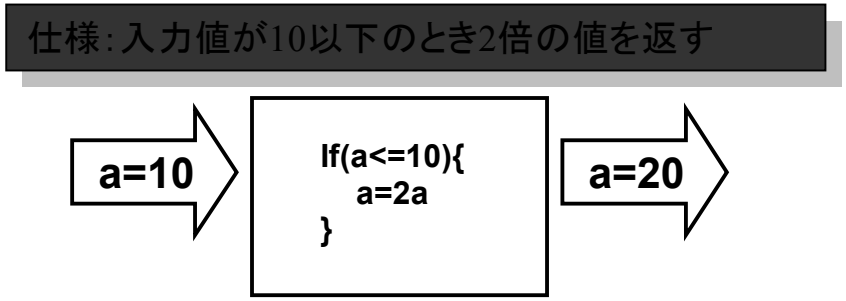
[補足]ホワイトボックスとブラックボックスの違い

- テスト設計方法が異なるため、テスト実施の動作結果の確認方法が異なる
  - ブラックボックスは、仕様で定義した振る舞いどおりに動作しているかを確認
  - ホワイトボックスは、ソフトウエアの構造を理解し、狙った処理経路を通った結果が正しく返ってくるかを確認

ブラックボックス



ホワイトボックス



3. テスト前にレビューを実施する

- ISTQBではレビューのことを静的技法（または静的テスト）と呼んでいる
- レビューのメリット・デメリット
  - メリット
    - ソフトウェアを動作させずに欠陥を見つけることが出来る
    - ソフトウェアに欠陥を埋め込む前に、その種を取り除くことが出来る
    - （動的なテストと比較して）低コストで実施できる
  - デメリット
    - ドキュメントだけでは実際に動いたものをイメージしづらい
      - 動かさないとわからない欠陥は見つけにくい
- レビューの種類
  - 人間が手動で実施
    - インスペクション
    - ウオークスルー
    - ピアレビュー
  - ツールで実施
    - 静的解析

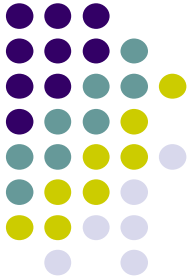
4. 開発した本人以外がテストを行う

- ISTQBでは開発した本人以外のテストを「独立したテスト」と呼んでいる
  - 「独立したテスト」のレベル（from ISTQB FL）
    - レベル1：ソフトウェアを作成した本人がテスト設計する
    - レベル2：開発者とは別の人（開発チームの人）がテスト設計をする
    - レベル3：開発者とは別の部署の人がテスト設計をする
    - レベル4：開発者とは別の会社の人がテスト設計をする
- 独立テストのメリット・開発者によるテストのメリット
  - 開発者のバイアスを排除できる（作ったものを壊す心構えを持つのは難しいうえ、仕様に対して持った誤解は、プログラミングのときも続く）
  - 「時間」「コスト」という定量的に捕らえやすい目標に気持ちがシフトしてしまうのを防ぐ効能がある
- 開発者によるテストのメリット
  - 独立テストより開発者自身が作ったコードのバグを見つけることが容易
  - テストで見つけたバグをデバッグするのが短時間で可能

テスト戦略のトレンド

1. 優先度の高い事柄からテストする
   - 分析的＝＞　e.g. 品質リスク分析
   - 方法論的　＝＞e.g. 品質特性をベースに優先度の高い品質を分析
   - コンサル型　＝＞ユーザ/開発者など外部の助言から優先度を判断
   - 動的かつ経験則的　＝＞e.g. バグ分析をベースに優先度を判断
2. 回帰テスト戦略を立てる
   - e.g.テスト自動化、影響度分析
3. 複数のテストを最適化する
   - e.g.テストベースレビュー
   - e.g.ライフサイクル全体のテスト（開発者が個人で行うテストとプロジェクトとして行うテスト）をコーディネイト

1. 優先度の高い事柄からテストをする



- 品質リスク分析の例

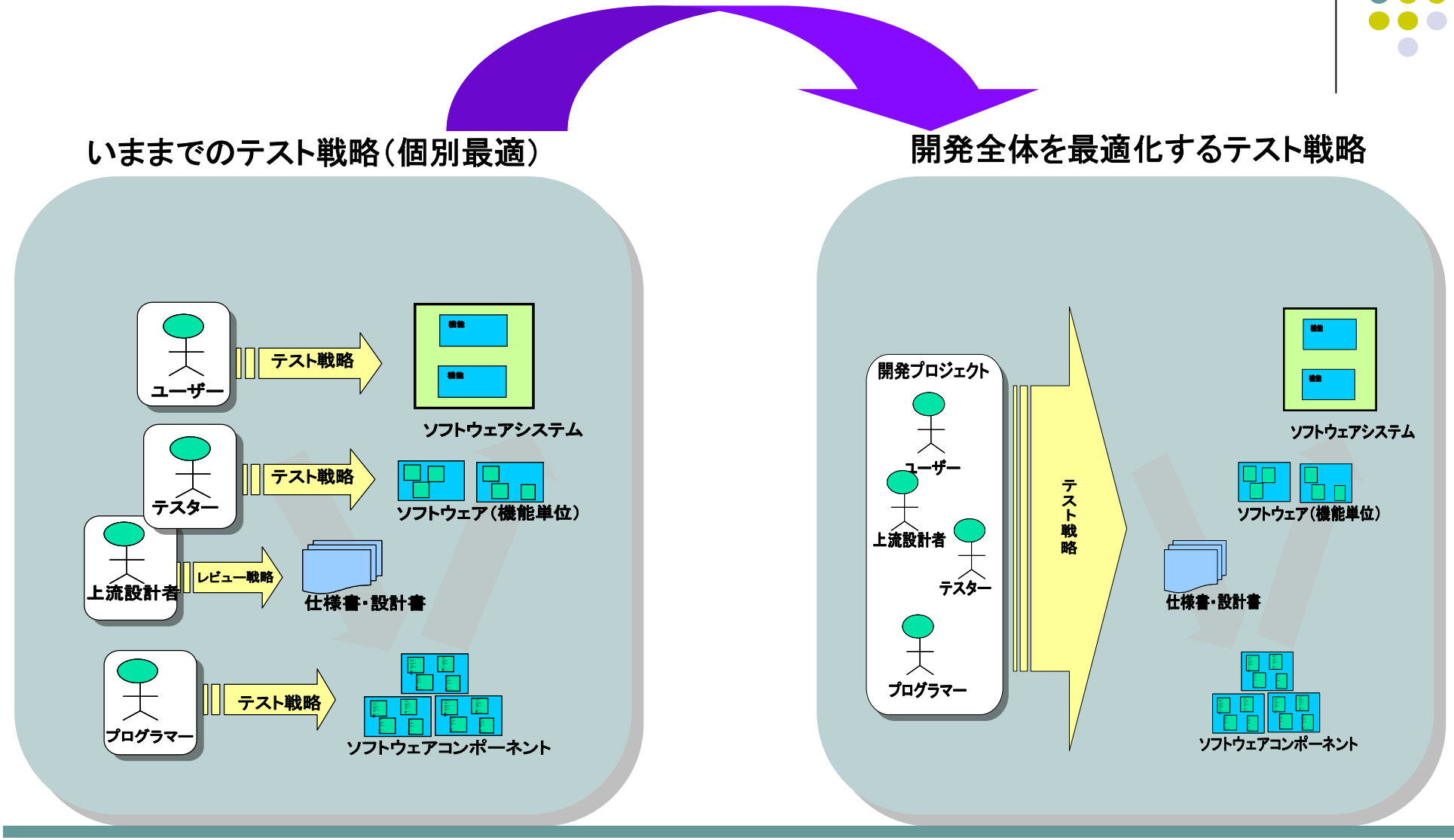
	カテゴリ	リスク	ビジネス的影響	不具合発生可能性	指数（優先度）	テスト方法
機能	印刷キュー	キューに残った印刷データの取り消しができない	4	3	12	基本機能
機能	印刷設定	ドライバの印字設定どおりに印刷できない	1	4	4	組み合わせ
構成	アプリケーション	MSエクセルに貼り付けた画像データを印刷するとノイズが入る	4	2	8	機能テスト

1~5　　組み合わせテスト
6～10　機能テスト
11～15　基本機能テスト
16～25　動作確認

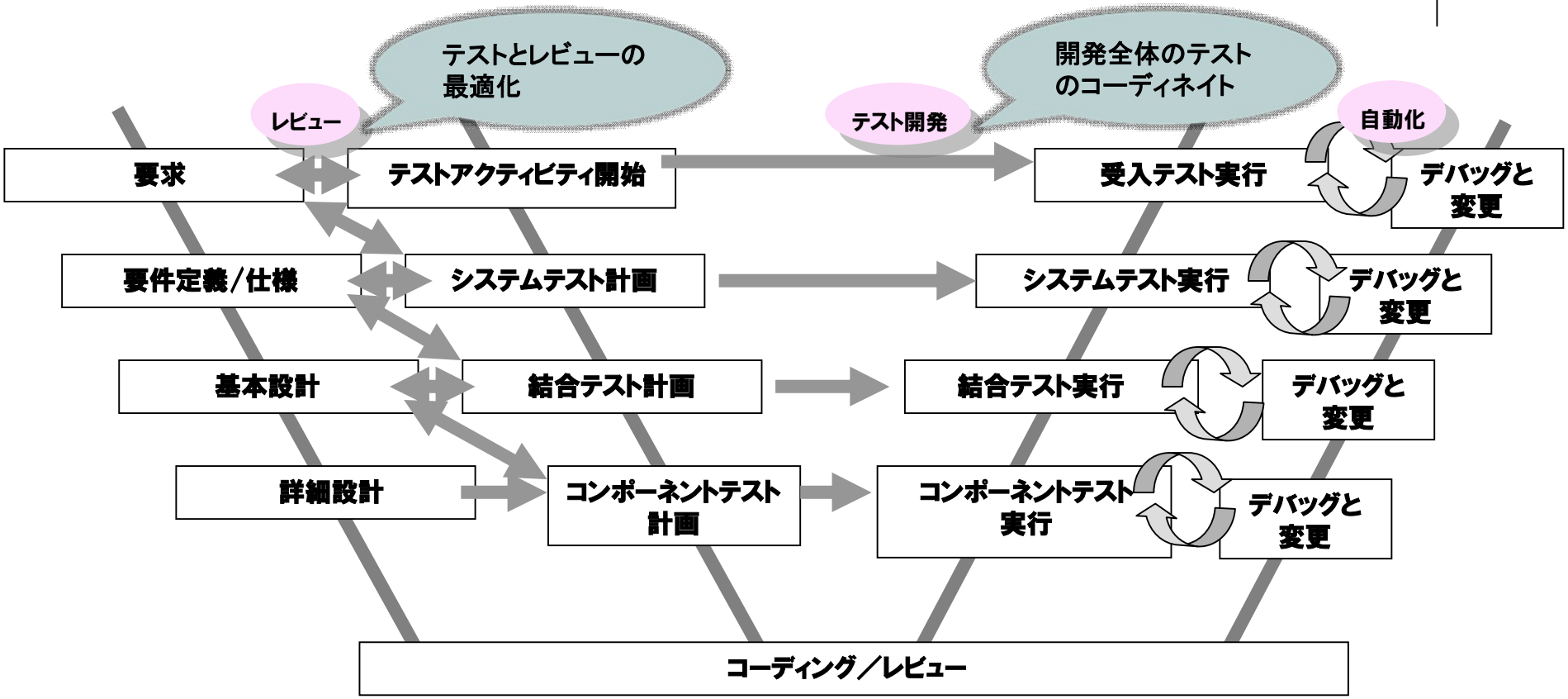
2. 回帰テスト戦略を立てる

- 回帰テストの戦略で考慮すること
  - 回帰テストの目的
    - 例えば以下のような目的が典型的
      - ディグレードを見つける
      - 一度正常に動作した箇所が常に正常に動作する
  - 開始時期
    - ベースラインにするビルドはどこからにするか
  - 対象のテストレベル
    - モジュール要件レベルの回帰テスト→xUnitを使って自動化
    - 業務要件レベルの回帰テスト→キャプチャリプレイツールで自動化
  - 実施方法
    - 何回も繰り返し同じテストをする場合は自動化の効果が高い
    - デイリービルド＆テストする場合は更に仕組み/教育から準備が必要
    - 人間が手動テストで実施する場合の要員確保/その段取りが必要
  - テストケース準備方法
    - 既存のテストケースから抽出するか否か？抽出基準は？
    - 回帰テスト用のテストケース（自動テストで一気に流せるテストスイート）
    - 「殺虫剤のパラドックス」を考慮するか？
      - 一度正常に動作した箇所が常に正常に動作することが目的ならば考慮しなくて良い

3. 複数のテストを最適化する



3.1 複数のテストを最適化するプロセス [W-Model]

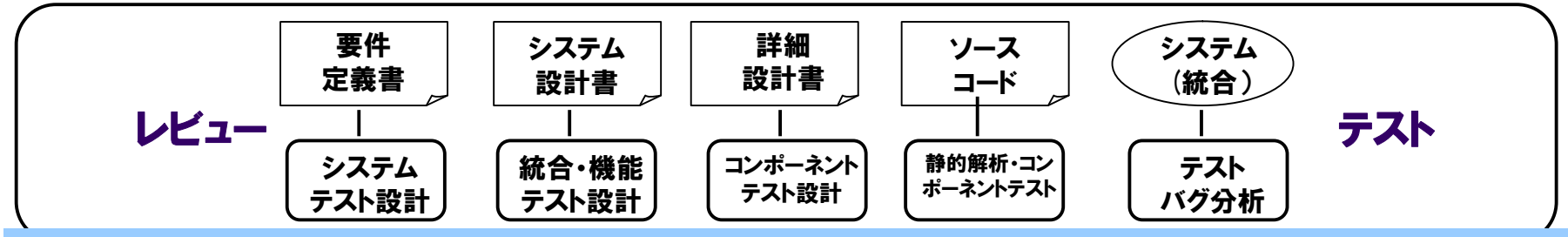


http://www.stickyminds.com/sitewide.asp?ObjectId=3572&Function=DETAILBROWSE&ObjectType=ARTを元に加筆

テストの関与を早め、テストフェーズの課題を取り除く

3.1.1 テストとレビューを最適化する

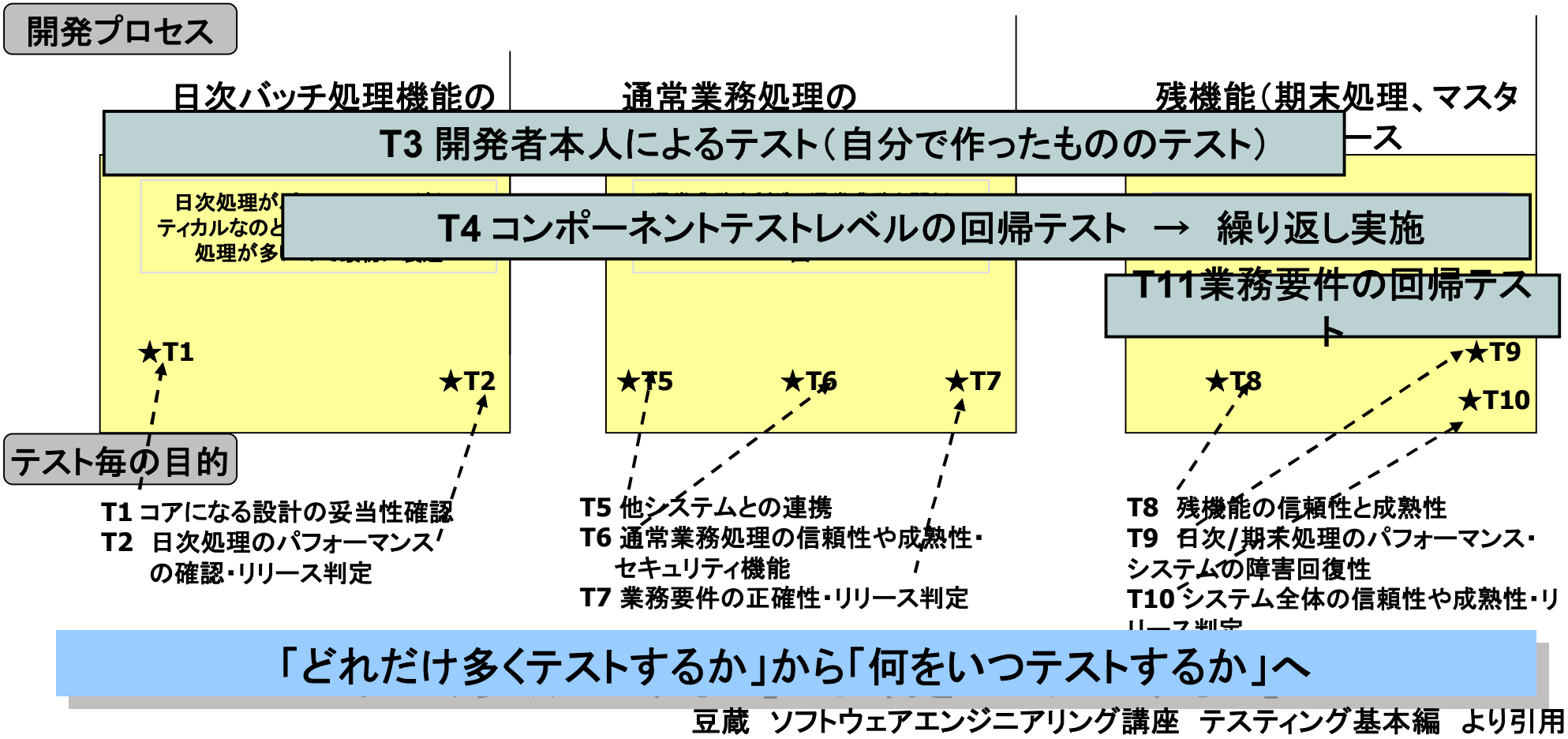
- テスト設計前にテストベースレビューを実施する
  - テスト設計の効率をよくする
    - テスト設計のため、元になる設計情報にあいまいな部分がなく、一貫性を持っていることをレビュー
    - テスト実施を想定しながら資料を熟読することにより、設計時に気づかなかった矛盾や不足していることを指摘
  - 欠陥を予防する
    - テスト設計の効率を良くする作業を、コーディングの前に対処することでソフトウェアに欠陥混入することを予防
      - e.g. 記載内容が不十分な仕様書からディシジョンテーブルや境界値分析を使ってテストケースを設計し、これを仕様書のレビューと言う形で、設計段階に開発にフィードバックする
  - テスト容易性(テスタビリティ)をあげる
    - テストが容易になるような機能や設計になっているかをレビュー
      - テスト実施を容易にする(テストした結果判断のためのログやメッセージが機能)
      - テストケースを削減する(DB接続処理などを共通化)



トータルにテスト工数を削減するための方策としてレビューをする

3.1.2 開発全体の(動的)テストをコーディネイト

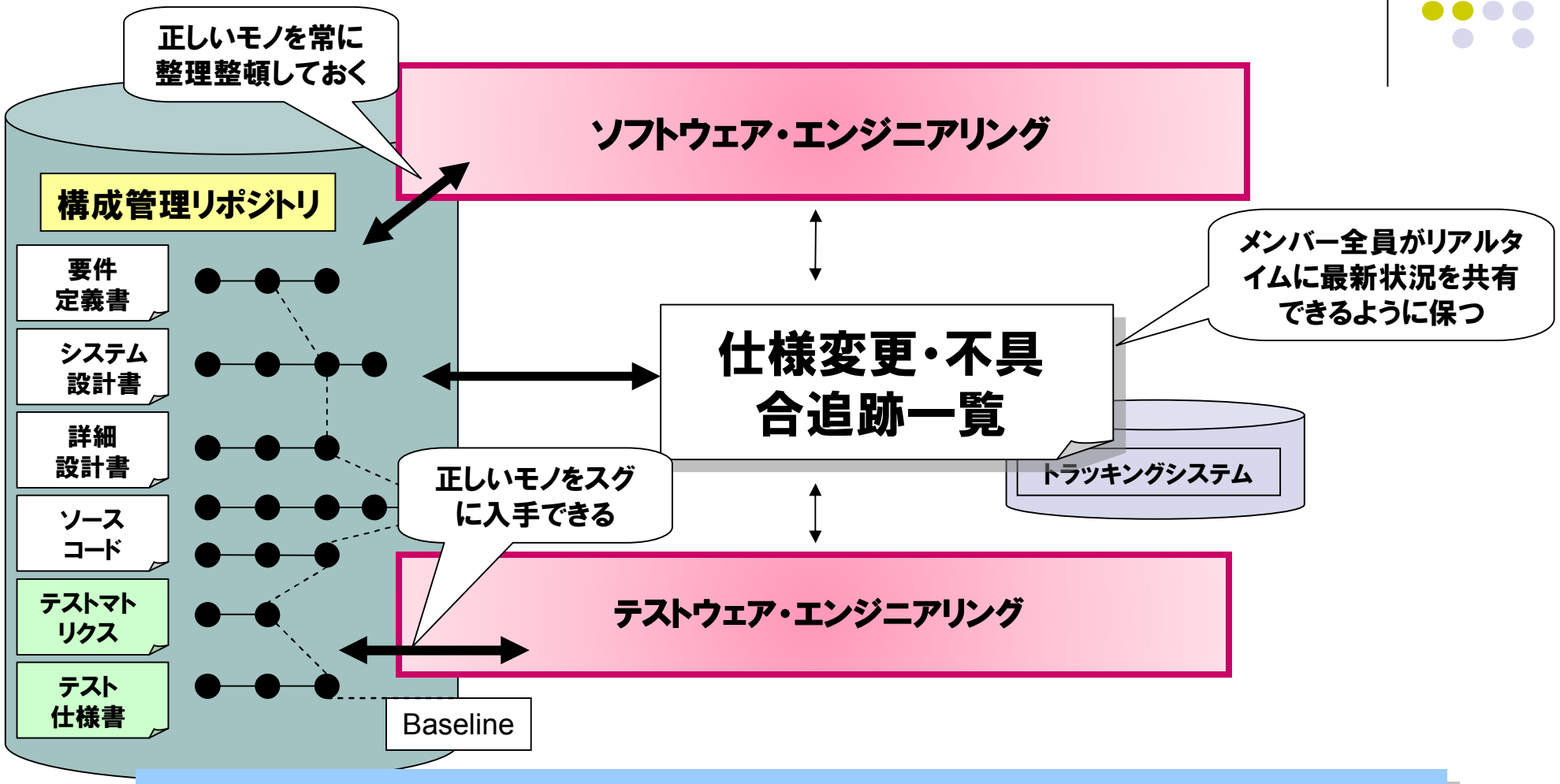
- 企業向けの売り上げ管理システムの開発を例に
  - リリースを3回に分けて行う計画を想定



「どれだけ多くテストするか」から「何をいつテストするか」へ

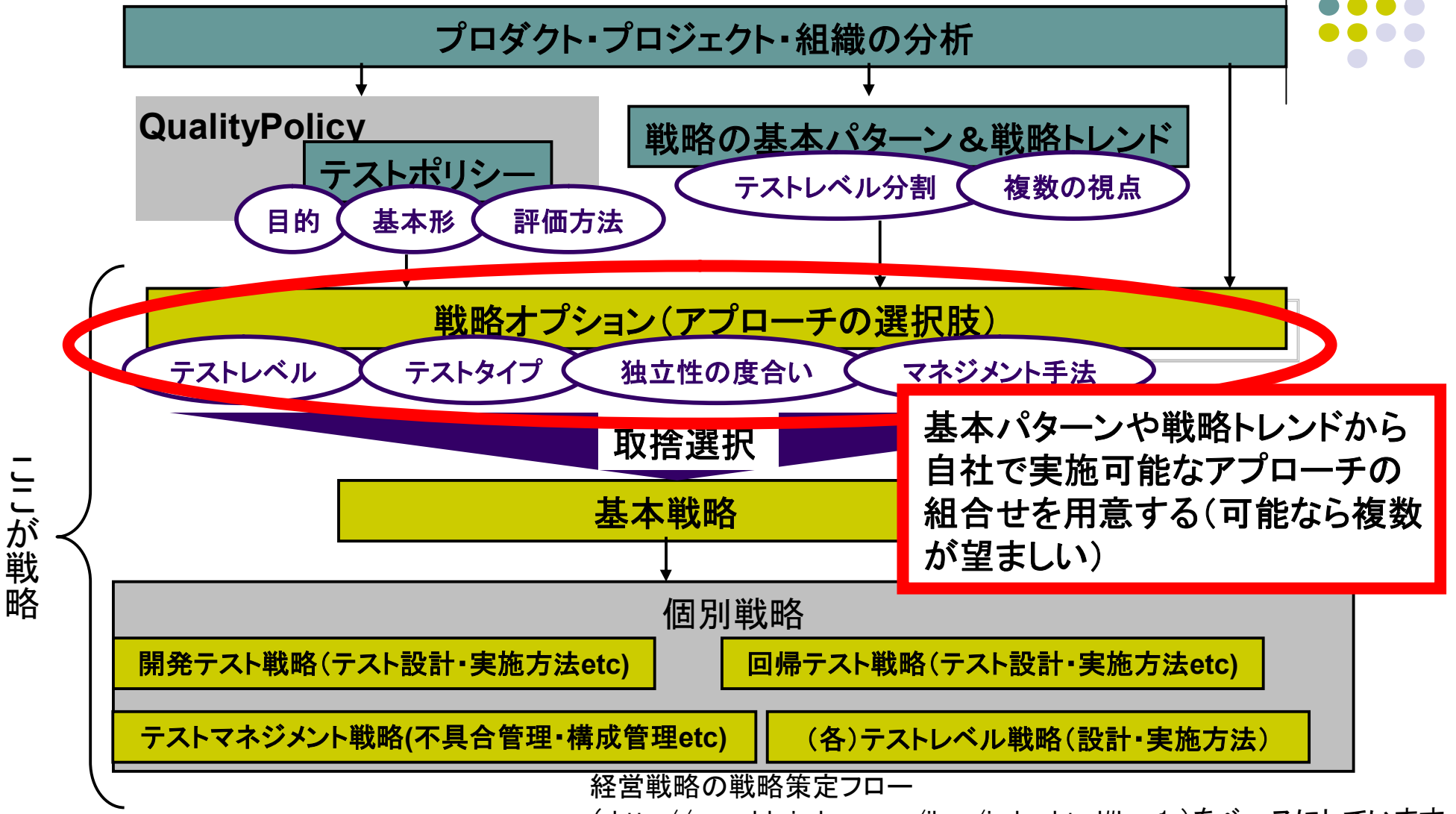
豆蔵 ソフトウェアエンジニアリング講座 テスティング基本編 より引用

[補足] テストと開発が並走できるインフラ



コミュニケーションを円滑にする仕組み＝並走のムダを防ぐ

戦略オプション



経営戦略の戦略策定フロー
（ http://www.blwisdom.com/ikey/index.html#key1 ）をベースにしています

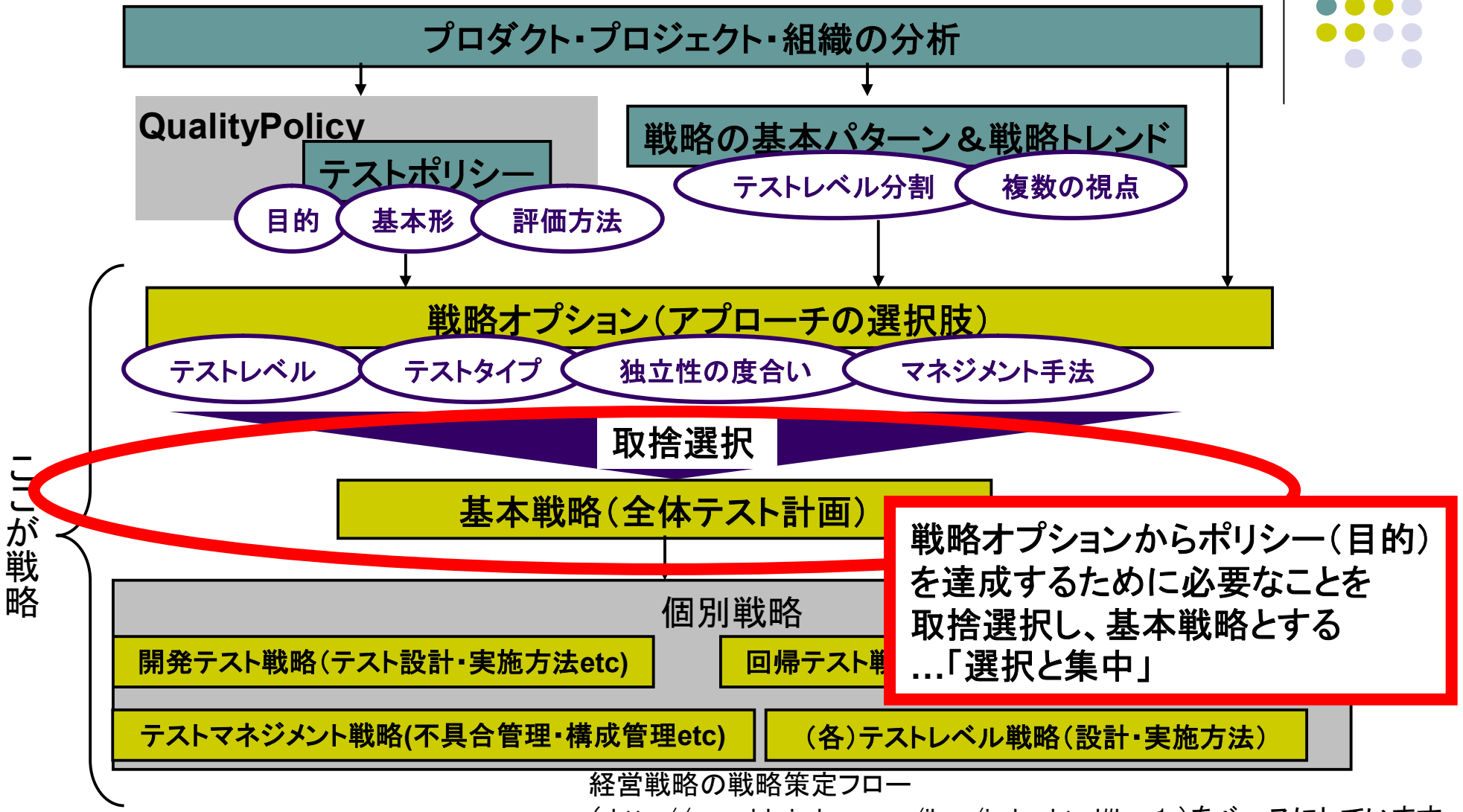
戦略オプションの例

- 優先度付けアプローチ、レビューアプローチ、テストレベル毎の役割をそれぞれ選択してまとめあげる
  - 複数のオプションを用意するのが望ましい

テストレベルの分割にも複数の選択肢があると更に望ましい

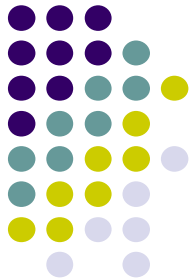
優先度	分析時→品質リスク分析による優先度付け テスト実行中→バグ分析による優先度付け見直し		
レビュー	テストレベルに対応する開発レベルの仕様書が出来た時点でテストベースレビューを実施		
レベル	**コンポーネントテスト**	**統合テスト**	**システムテスト**
目的	技術検証、欠陥除去	欠陥除去	信頼性評価
誰が	プログラマ	開発チームのテスト担当	テストチーム
どこで	開発用パソコン	開発用テスト環境	テスト用疑似本番環境
何を	メソッド，クラス	コンポーネント	全コンポーネント
実施するテストタイプ	・機能テスト（ロジック） ・構造テスト	・機能テスト （画面単位のUI、機能） （データベース接続）	・機能テスト（業務シナリオ） ・負荷テスト ・構成テスト
ツール	・コード・カバレッジ ・テスティングフレームワーク（xUnit）	・テスティングフレームワーク（xUnit） ・キャプチャ・リプレイ・ツール	・負荷テスト・ツール ・OSイメージのバックアップ・リストア・ツール

基本戦略(全体テスト計画)



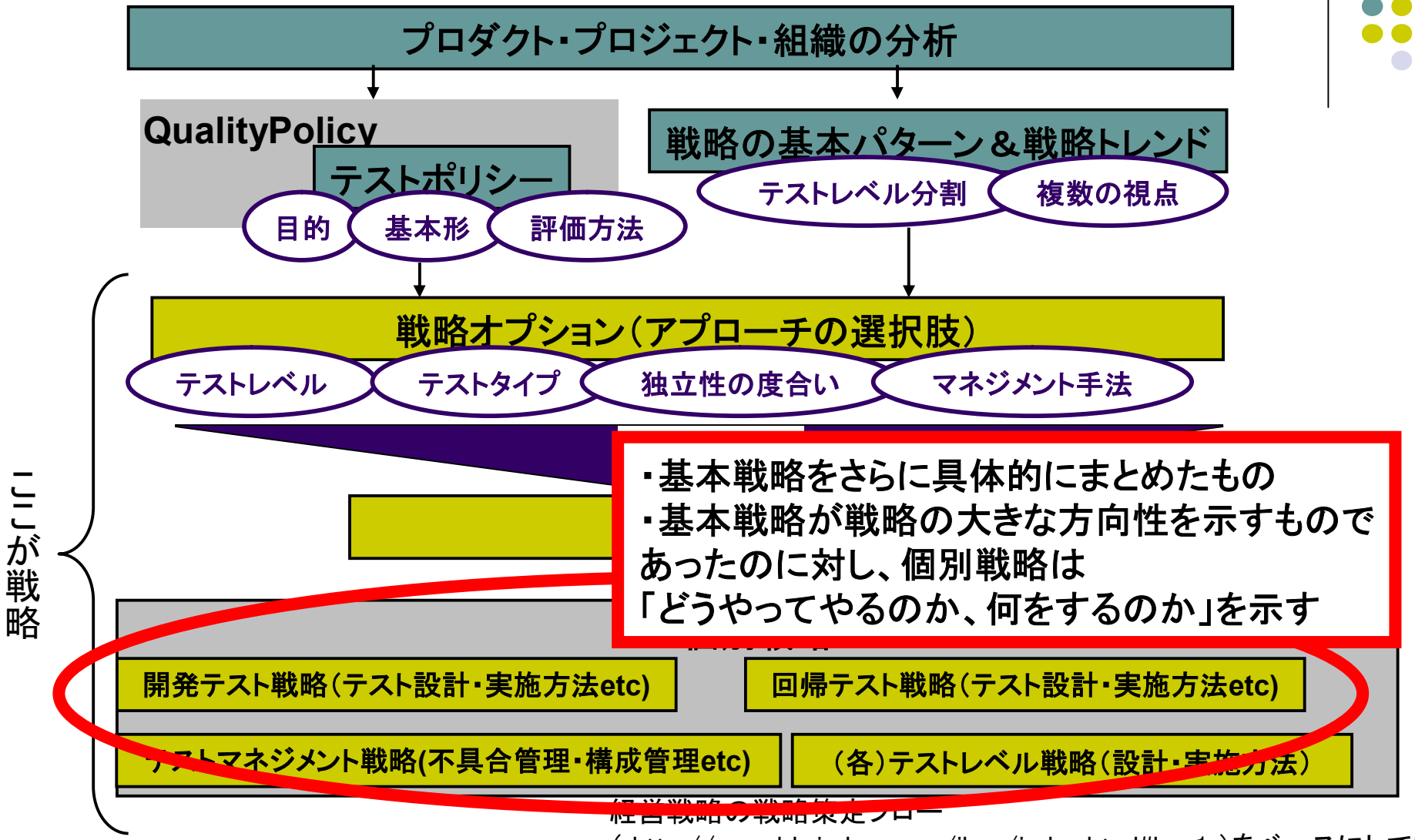
経営戦略の戦略策定フロー
(http://www.blwisdom.com/ikey/index.html#key1)をベースにしています

基本戦略のポイント・ドキュメント内容



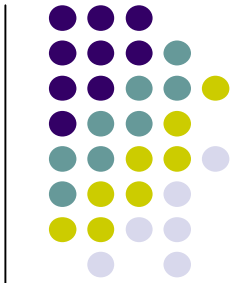
- 基本戦略（全体テスト計画）のポイント
  - ポリシー（目的）との一貫性を保つ
  - 開発全体の課題を解決できるアプローチの集合体であること
  - 個別戦略への橋渡しが出来るものであること（以降のリソース配分の根拠となる）
- 全体テスト計画テスト計画に戦略面として記載すること
  - 戦略の概要（一言で言うとどんな戦略か）と背景（選択した理由）
    - 概要例：「リスク分析し、リスクの高い機能に対して早期にリスクを軽減する戦略。本戦略では網羅性が弱くなるため、最終フェーズでのシステム全体に対する機能テストを実施する。」
    - 背景例：「技術的にはじめての挑戦になることが多く、通常の最終フェーズでのシステム全体に対する機能テストだけでは、不具合の修正が間に合わなくなる。」
  - 決定した戦略でのアプローチ
    - テストの配置（実施時期）と各テストの目的設定
      - 欠陥を見つけるため
      - 信頼性を評価するため
      - 技術検証
      - 要件の妥当性確認
    - テストアイテム/テストベース
    - テストレベル
    - テストタイプ
    - 担当
    - 環境
  - 戦略に必要な（確保しなければならない）リソース

個別戦略

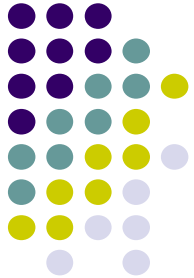


（ http://www.blwisdom.com/ikey/index.html#key1 ）をベースにしています

個別戦略のポイント

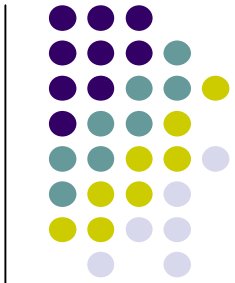


- 個別戦略のポイント
  - 基本戦略との一貫性があること
  - テスト設計、テスト実施、準備などの意向の実際の活動に対する指針になるよう具体的であること
  - 他の個別戦略とのズレがないようにすること
- 個別戦略として必要なもの
  - 各テストレベル毎の戦略
    - テストタイプ毎のテスト分析、設計手法（HAYST法、VSTeP、ゆもつよ　etc. ）
    - テスト対象範囲（他のレベルとの整合性や分担）
    - テスト実施方法
      （バックツーバックテストやデータ、ツール、環境などを使う方法をより具体的に提示）
  - 回帰テスト戦略
  - マネジメント戦略（以下に対するやり方（アプローチ）の集合体）
    - 不具合管理
    - 構成管理
    - メトリクス関連（データ収集、分析）
    - テストケース管理
    - スケジュール管理
  - 戦略に必要な（確保しなければならない）リソース



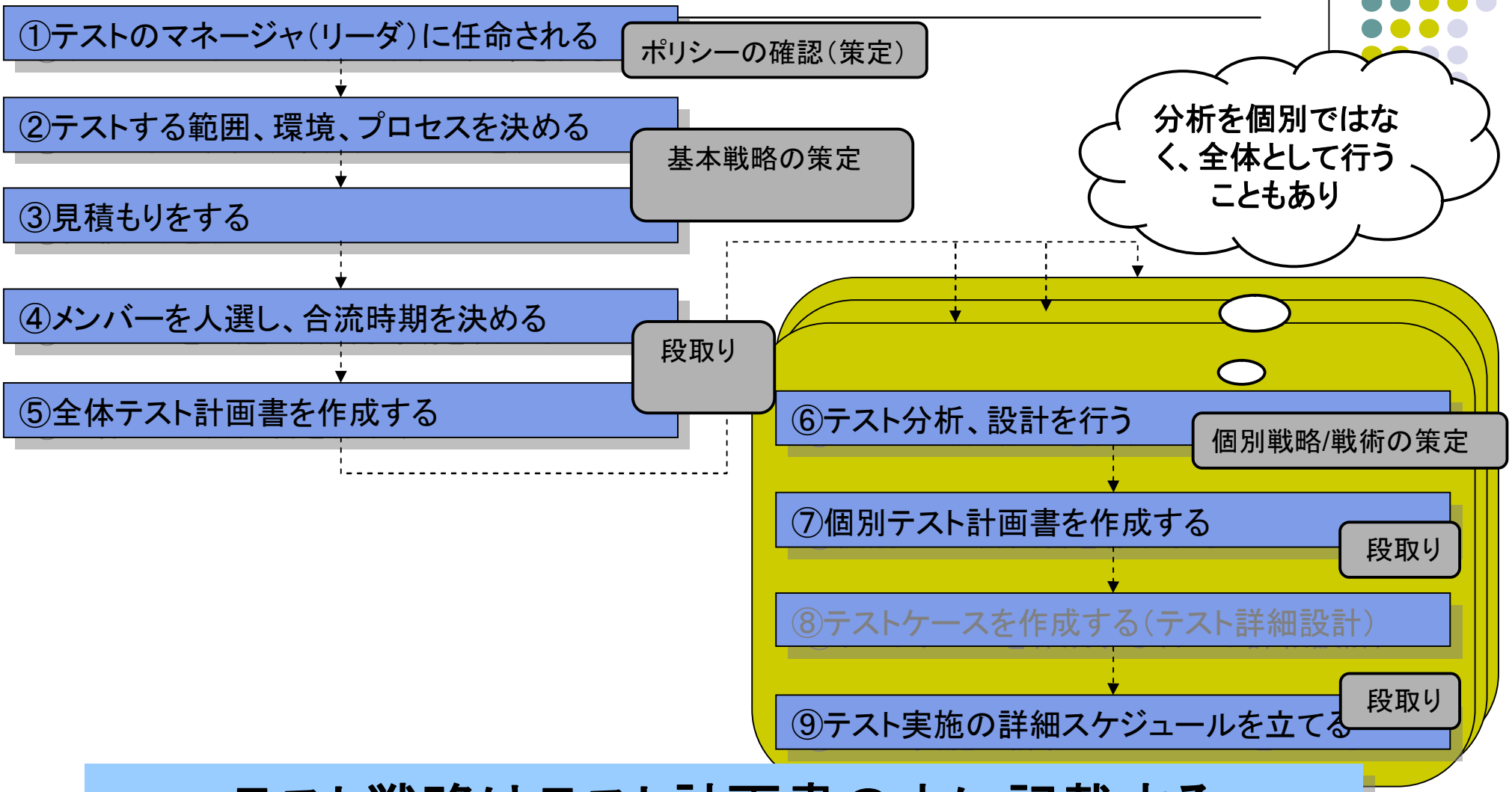
ここまでが

基本のおさらい



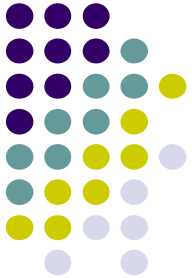
テスト戦略立案の進め方

テストを計画していくときの作業フロー



テスト戦略はテスト計画書の中に記載する

[参考]計画書テンプレートの例



IEEE 829 テストプランテンプレート

- ■テスト計画書文書番号（Test plan identifier）
- ■レファレンス（References）
- ■はじめに（Introduction）
- ■テストアイテム（Test items）
- ■テストする機能（Features to be tested）
- ■テストする必要のない機能（Features not to be tested）
- ■アプローチ（Approach）
- ■テストアイテムの合否判定基準（Item pass/fail criteria）
- ■中止基準と再開要件（Suspension criteria and resumption requirements）
- ■テスト成果物（Test deliverables）
- ■テストティングタスク（Testing tasks）
- ■実行環境（Environmental needs）
- ■責任範囲（Responsibilities）
- ■人員計画、トレーニングプラン（Staffing and training needs）
- ■スケジュール（Schedule）
- ■リスクとその対策（Risks and contingencies）
- ■承認（Approvals）

ここに戦略を具現化した
やり方（アプローチ）を記載

業界の標準テンプレートであり、
これだけ多くのことを考慮しなければ
ならない…ということ

↓

ただし、このままドキュメントにするのは
大変なので工夫が必要

ホットスポットとフローズンスポット

※　各項目の意味、記載する内容に関しては、
「現場の仕事がバリバリ進むソフトウェアテスト手法」を参照してください

[参考]計画のホットスポットとフローズンスポット

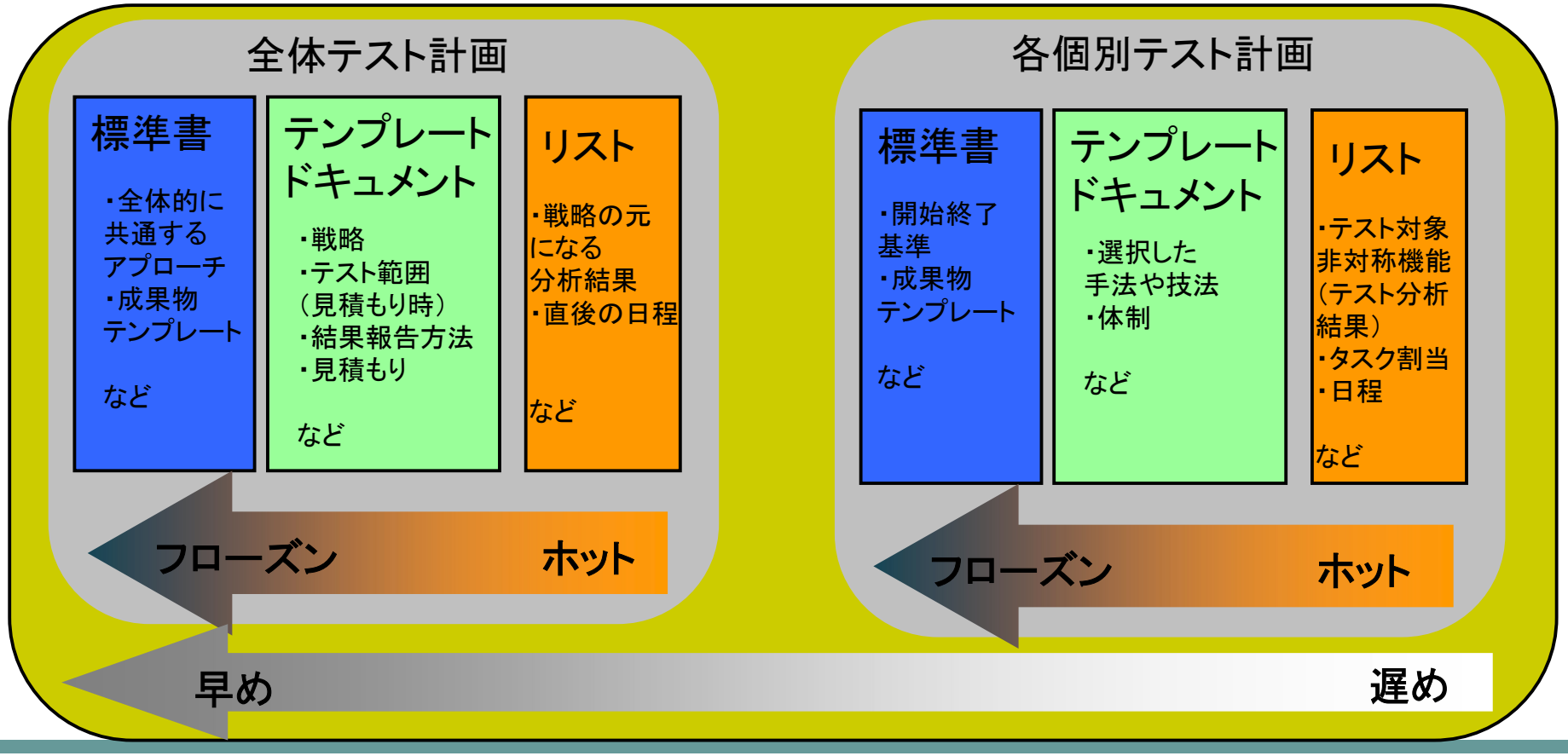
- ホットスポットとフローズンスポット
  - フローズンスポット
    - 分析の結果、相互作用や変化がなく固定化しており、今後の変更が起こらないと思われる部分「固定部分」
      - メンバー間、顧客、上層部で共有すべき原則や組織で統一した基本的作業
      - 標準書のような形式でドキュメント化する。内容の教育を行ったりガイドを作成し、（改まった説明などなしに）共通の認識を持てるようにする
  - ホットスポット
    - 分析の結果、新たな要求の発生が予想され、その変更に対し柔軟に対処できるようにした部分「可変部分」
      - プロジェクトが進むにつれ見えてくるものorメンバーだけで共有できればよいもの
      - プロジェクト毎に内容が異なるが、テンプレート化したり、リスト形式やデータベースで管理し、変更を容易にする

[参考]ホット・フローズンスポットを見つける

- フローズンスポット
  - 過去のプロジェクトから流用できること
  - 組織共通の認識として認知されていること
    e.g.標準的なテストプロセス、テストケース管理、欠陥追跡、構成管理
    見積もりに使う基礎データ、用語
    - 認知されているためには、日ごろの教育などが重要
- ホットスポット
  - プロジェクト毎に作るもの→必要な時期に必要な部分を作る
    - 比較的早期に作成したほうがよいこと
      - 早期のタイミングで、メンバ・顧客・経営者と合意しておくべきこと
        - e.g.テストの目的(定義)、基本戦略、見積もり
      - 次の作業に直結すること
        - e.g. テスト分析作業に必要なタスクやスケジュール
    - ギリギリまで変更が入ること前提にしておくこと(後でもかまわない)
      - 計画をたてる段階で明確に出来ないこと
        - e.g.テスト対象範囲
      - 仕様変更・メンバー移動などによる変更が入ると思われること
        - e.g.テスト実施スケジュール

[参考]テスト計画のホット・フローズンスポット例

全体テスト計画→早めに決めてメンバー間で合意すべきこと
（戦略、全体的に共通する個別戦略、テスト範囲（見積もり時）、見積もり　など）
各個別テスト計画→いろいろなことがわかってから決めていくこと
（テスト対象・非対象機能、開始・終了基準、技法や手法、個別日程）



全体テスト計画でのアプローチ作成までのステップ

- テスト方針・目的設定
  - 開発ライフサイクル全体にわたりどうテストを行っていくか（全体方針）を決め、どのようなテストが必要なのかを特定する
- テストの配置
  - ライフサイクルの中でどの目的のテストを「いつ」するかを決める。
    - 「テスト方針・目的設定」の目的をブレイクダウンし、具体化する
- テスト対象範囲特定（スコープ）
  - テスト対象アイテムをシステム要件書やシステム設計書から明らかにする。
- テストレベルとテストタイプの選定
  - どの程度の深さ、抽象レベルで、何に着目してテストするかを明らかにする。
- テスト実施の役割分担
  - テストごとに担当組織とその担当/責任範囲を定義する。

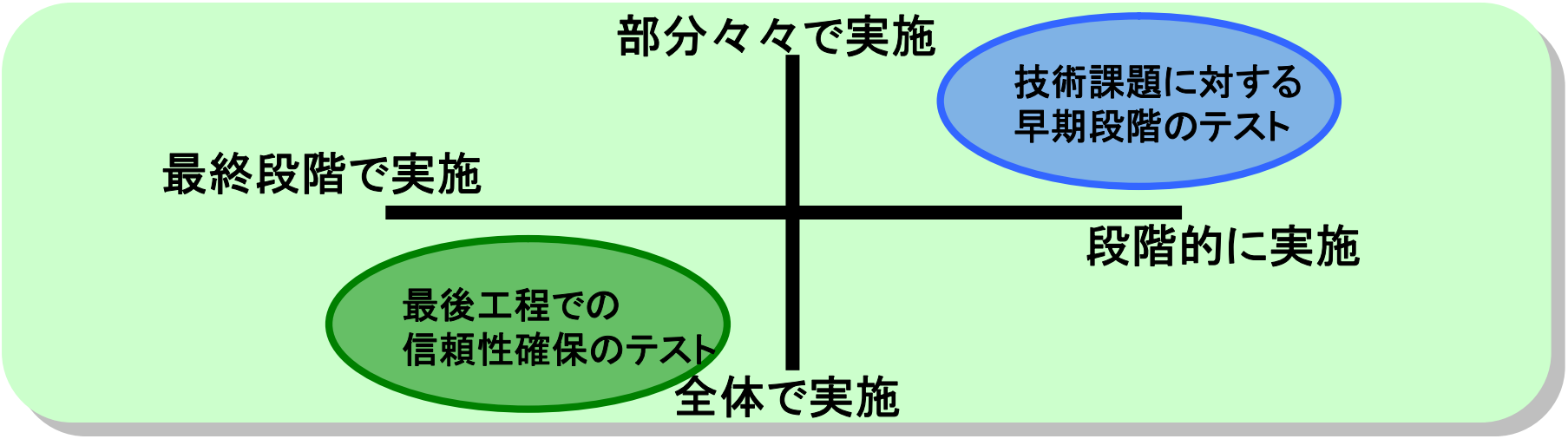
テストの方針・目的設定

- 必要なテストの特定の観点 or テストの目的の典型例
  - 開発を進めるための重要な項目の評価・確認のためのテスティング
    - 技術課題に関連して: 解決策の妥当性確認・検証
    - 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
    - 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
  - 仕様・要件への充足性を確認するためのテスティング
    - 仕様・要件に対する合致性の確認
      - 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)
  - 信頼度の向上及び評価のためのテスティング
    - 意図的な欠陥識別
      - e.g. ストレス系・評価系のテスト
  - 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テストケースを使用しての統計的な信頼度の測定(信頼度成長曲線)

[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　より引用]

典型的な全体方針

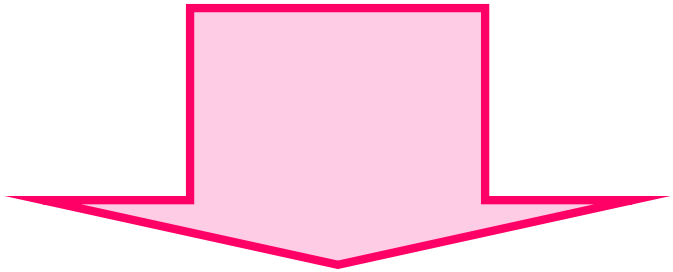
- 基本は以下の組合せの集合
  - {最終段階で実施, 段階的に実施}×{全体で実施, 部分々々で実施}



- テスティングを開発プロジェクトにどう活用したいのか…その考え方、というか理念を基に方針を決める

テストの配置

- 方針に沿って、テスト結果からのフィードバックがもっとも適切な時期になるようにテストを設定する
  - 「バインディングタイム」



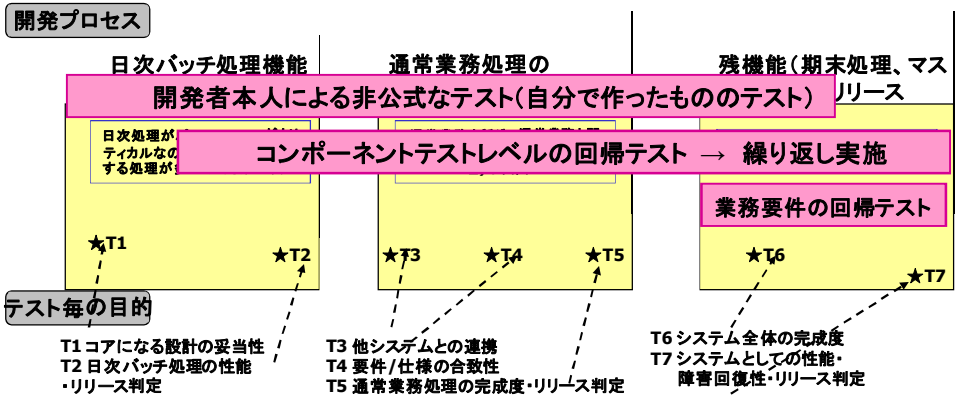
- 特定or配置したテストに対する「目的」をプロジェクトの状況に合わせて具体的にする
  - 目的を明らかにするためには以下に対する充分な分析が必要
    - プロダクト（開発範囲）/開発プロセス/開発におけるリスク（品質リスク、計画リスク）

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　より引用

配置したテスト毎に属性を決める

- 全体テスト計画書に掲載する、テストアプローチ（戦略）の内容は以下の通りとなる
  - テストの配置
    - ライフサイクルの中でどの目的のテストをいつするかを決める。
  - テスト対象範囲特定
    - テスト対象項目をシステム要件書やシステム設計書から明らかにする。
  - テストレベルとテストタイプの選定
    - どの程度の深さ、抽象レベルで、何に着目してテストするかを明らかにする。
  - テスト実施の役割分担
    - テストごとに担当組織とその担当/責任範囲を定義する。

- 立案結果の例



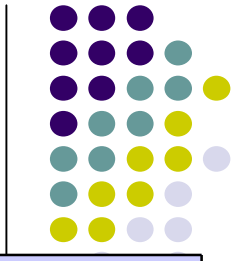
	テストの目的	テスト対象項目	テストレベル	テストタイプ	担当チーム
	T1 ・コアになる設計の妥当性確保	・基盤アーキテクチャ ・セキュリティメカニズム ・XMLデータ通信	・単体 ・統合	・構造テスト ・機能テスト ・セキュリティテスト	・開発
	T2 ・日次バッチの性能 ・リリース判定	・機能 ・サーバ間のデータ疎通 ・処理時間	・統合 ・システム	・機能テスト ・パフォーマンステスト ・ボリュームテスト ・ストレステスト	・統合→開発 ・システム→QA
	…	…	…	…	…

全体テスト計画でのアプローチの例...その1

	テストの目的		テスト対象アイテム	テストレベル	テストタイプ	役割
	T1	・コアになる設計の妥当性確保	・基盤アーキテクチャ ・セキュリティメカニズム ・XMLデータ通信	・単体 ・統合	・構造テスト ・機能テスト ・セキュリティテスト	・開発
	T2	・日次バッチの性能 ・リリース判定	・機能 ・サーバ間のデータ疎通 ・処理時間	・統合 ・システム	・機能テスト ・パフォーマンステスト ・ボリュームテスト ・ストレステスト	・統合→開発 ・システム→QA
	T3	・他システムとの連携	・連携機能（実データ使用）	・統合	・ボリュームテスト ・構造テスト	・開発・ユーザ（現地テスト）
	T4	・要件/仕様の合致性	・業務フローから見た合致性	・システム	・機能テスト	・QA
	T5	・通常業務処理機能の完成度・リリース判定	・機能 ・データアクセス処理での漏洩、改ざん	・統合 ・システム	・機能テスト ・セキュリティテスト	・統合→開発 ・システム→QA
	T6	・システム全体完成度	・機能	・統合 ・システム	・機能テスト	・統合→開発 ・システム→QA
	T7	・システムの性能 ・障害回復性 ・リリース判定	・実環境での応答時間 ・実環境でのシステム復旧処理	・システム	・パフォーマンステスト ・ストレステスト ・ 障害回復テスト	QA・ユーザ（現地テスト）

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　より引用

全体テスト計画でのアプローチの例..その2



	目的	テスト対象アイテム	テストレベル	テストタイプ	役割
MT1	・自身で書いたコードが想定どおり動くことを確認	モジュール	単体テスト	機能テスト 構造テスト	プログラマ
MT2	・既存モジュール含め、ブロック単位でビルドし機能が正しく動くことを確認	ブロック	ブロックテスト	機能テスト	開発チーム
MT3	・ブロック間シーケンスの確認 ・S/Wシステムテストへのリリース判定	複数ブロック （評価用試作機）	ブロック統合テスト	機能テスト 構造テスト	システム検証チーム
MT4	・ソフトウェア全体の機能要件・非機能要件の確認 ・システムテストへのリリース判定	S/Wシステム （量産試作機）	S/Wシステムテスト	機能テスト 非機能テスト	商品検証チーム
MT5	・H/W含めた製品としての機能要件・非機能要件の確認 ・出荷判定	システム	システムテスト	機能テスト 非機能テスト	QAチーム

アプローチの実現可能性を検討する

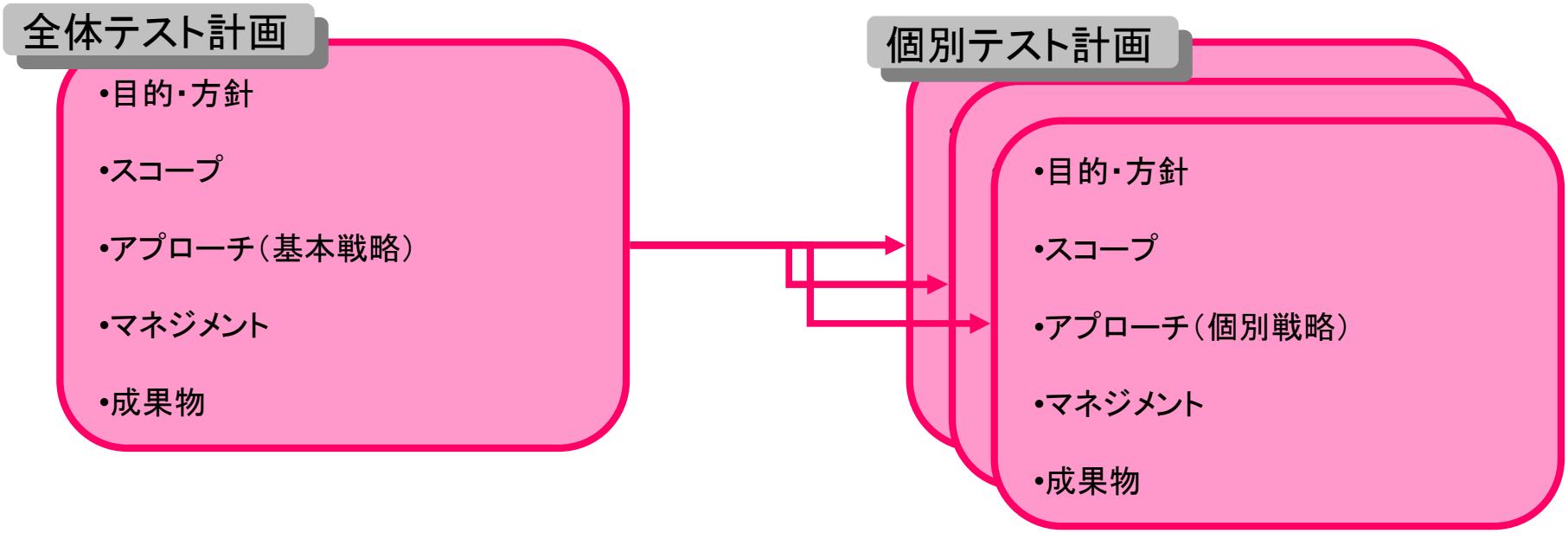
- アプローチを考えてから、それを実現するためにどうするか？という順序で考えること　←ココ大事なところ！！
  - 段取り→アプローチをどう実現するか？
    - 人材の確保
    - 教育
    - テスト環境の調達や準備
    - 開発含めたメンバー間のコミュニケーションを円滑にする仕組みの準備
    - テストの結果を伝える手段→成果物に関するとりきめ
    - ツール
  - コスト→コスト面で実現可能か？

- 段取りを考えてもコスト面からみて実現不可能なアプローチで会った場合に、アプローチに対して調整を入れていく

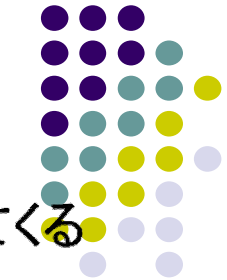
やりたいことを「どう実現するか？」と考えること！

テスト計画の段階、テスト戦略の段階

- テスト計画に段階があるように、戦略にも段階があり、各段階においてそれぞれの要素を検討することになる
  - 全体テスト計画
    - 開発プロジェクトを通じて行うテストテスト全体の進め方の指針をまとめる
      - どのようなテストレベル、どのようなテストタイプが適用されるかを決める
  - 個別テスト計画
    - 全体テスト計画で決めた個別のテストに対して進め方をまとめる
      - 個別テスト計画の分割方法も全体テスト計画で決める

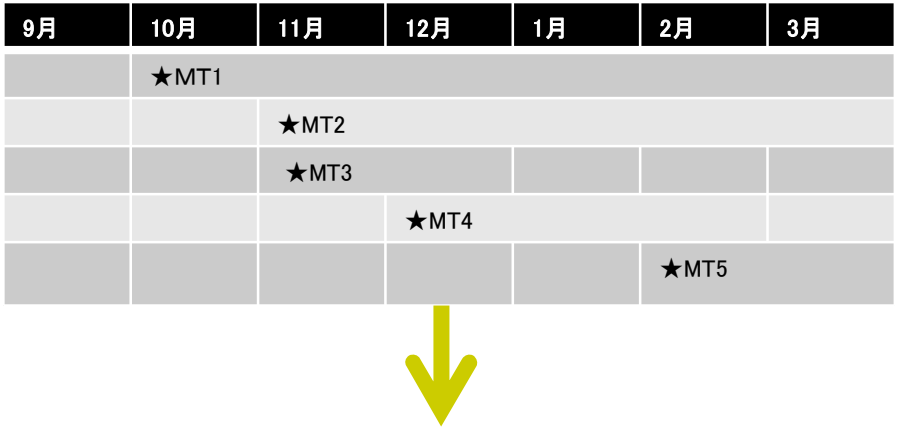


個別テスト計画の考え方のポイント



- 個別計画も基本は全体テスト計画と一緒であるが細かいところでは違いが出てくる
  - 方針の考え方
    - 全体テスト計画の目的に沿って、該当するテストにて留意すべきポイントをまとめる
      - リスク、開発プロセス・スケジュールなどを考慮。
    - テストとしてそれらに対しどう対処していくかを決める。
  - スコープの考え方
    - 全体テスト計画で定義したテストアイテムの機能を特定する。
      - 詳細な機能は時間と共に明らかになるのであくまで計画時に分かる程度でよい。
    - 機能・非機能要件の中から品質リスクを元に優先度付けを行う。
  - テストアプローチ(テスト戦略)の考え方
    - テストタイプのブレイクダウンを行ってからテストの配置を考える(テストカテゴリ)
      - 何に着目してテストするかを明らかにする
    - テストの配置
      - テストカテゴリ、品質リスクを元にいつ何をテストするかを決める
    - 配置したテストそれぞれに目的を設定
      - 目的の例
        - 仕様書に新規に追加された機能記述が正しく実装されていることの検証
        - 仕様書の記述に変更の無い機能記述が正しく実装されていることの検証
        - GUI・メッセージの文言や入力受付の確認
        - ユーザが使う状況を想定して各機能を流れで操作したときに想定どおり動くことの確認。
        - 既に確認が終わっている部分から不具合が発生しないことの確認

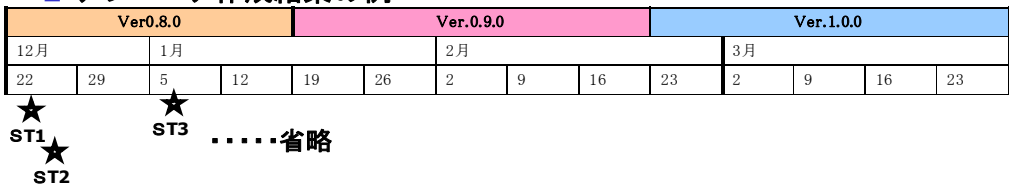
個別テスト計画でのテストの割付例



	目的	テストアイテム（テスト対象項目）	テストレベル	テストタイプ	役割
MT1	・自身で書いたコードが想定どおり動くことを確認	モジュール	単体テスト	機能テスト 構造テスト	プログラマ
MT2	・既存モジュール含め、ブロック単位でビルドし機能が正しく動くことを確認	ブロック	ブロックテスト	機能テスト	開発チーム
MT3	・ブロック間シーケンスの確認 ・S/Wシステムテストへのリリース判定	複数ブロック（評価用試作機）	ブロック統合テスト	機能テスト 構造テスト	システム検証チーム
MT4	・ソフトウェア全体の機能要件・非機能要件の確認 ・システムテストへのリリース判定	S/Wシステム（量産試作機）	S/Wシステムテスト	機能テスト 非機能テスト	商品検証チーム
MT5	・H/W含めた製品としての機能要件・非機能要件の確認 ・出荷判定	システム	システムテスト	機能テスト 非機能テスト	QAチーム

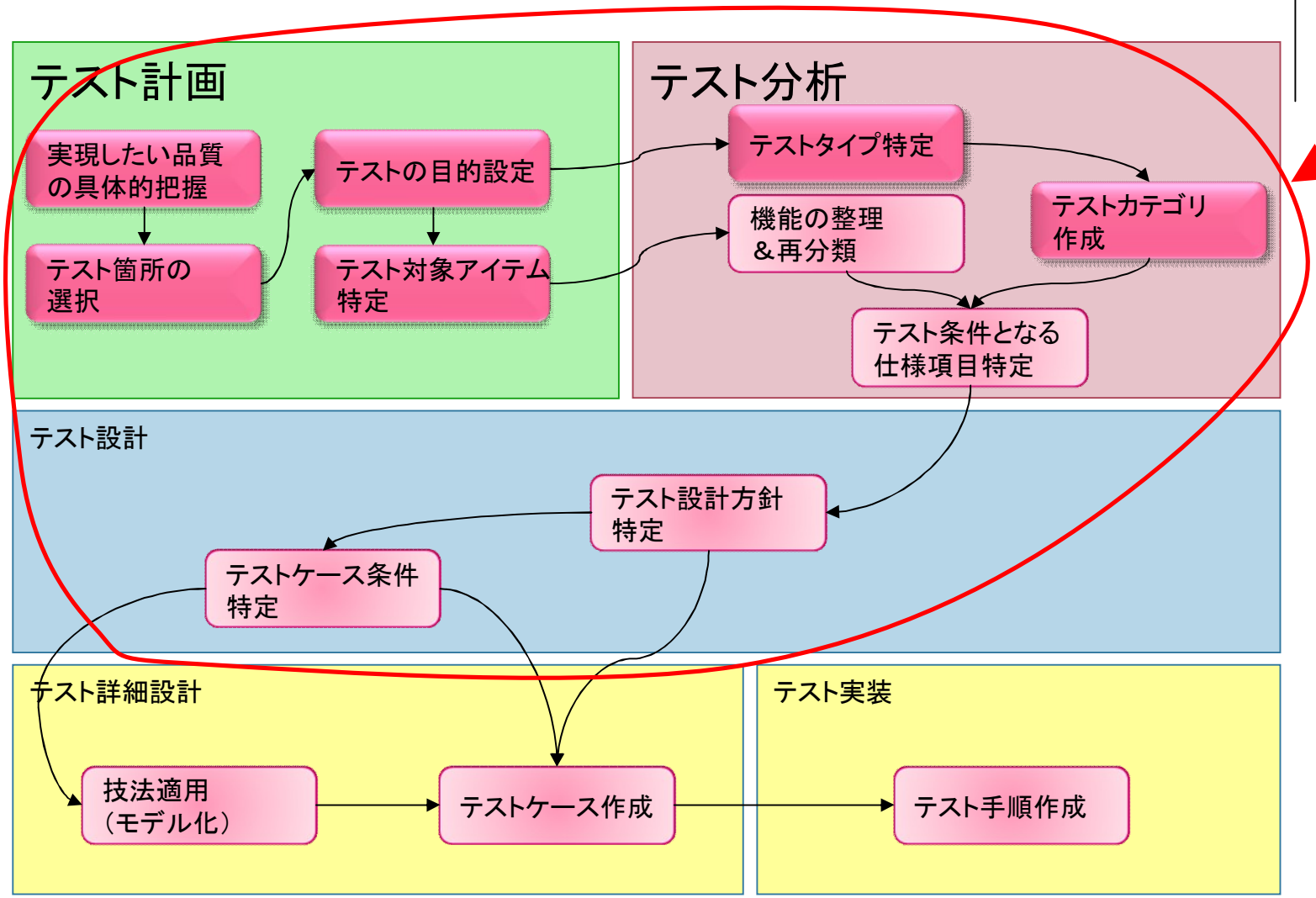
MT4のテストアプローチ

■ アプローチ作成結果の例



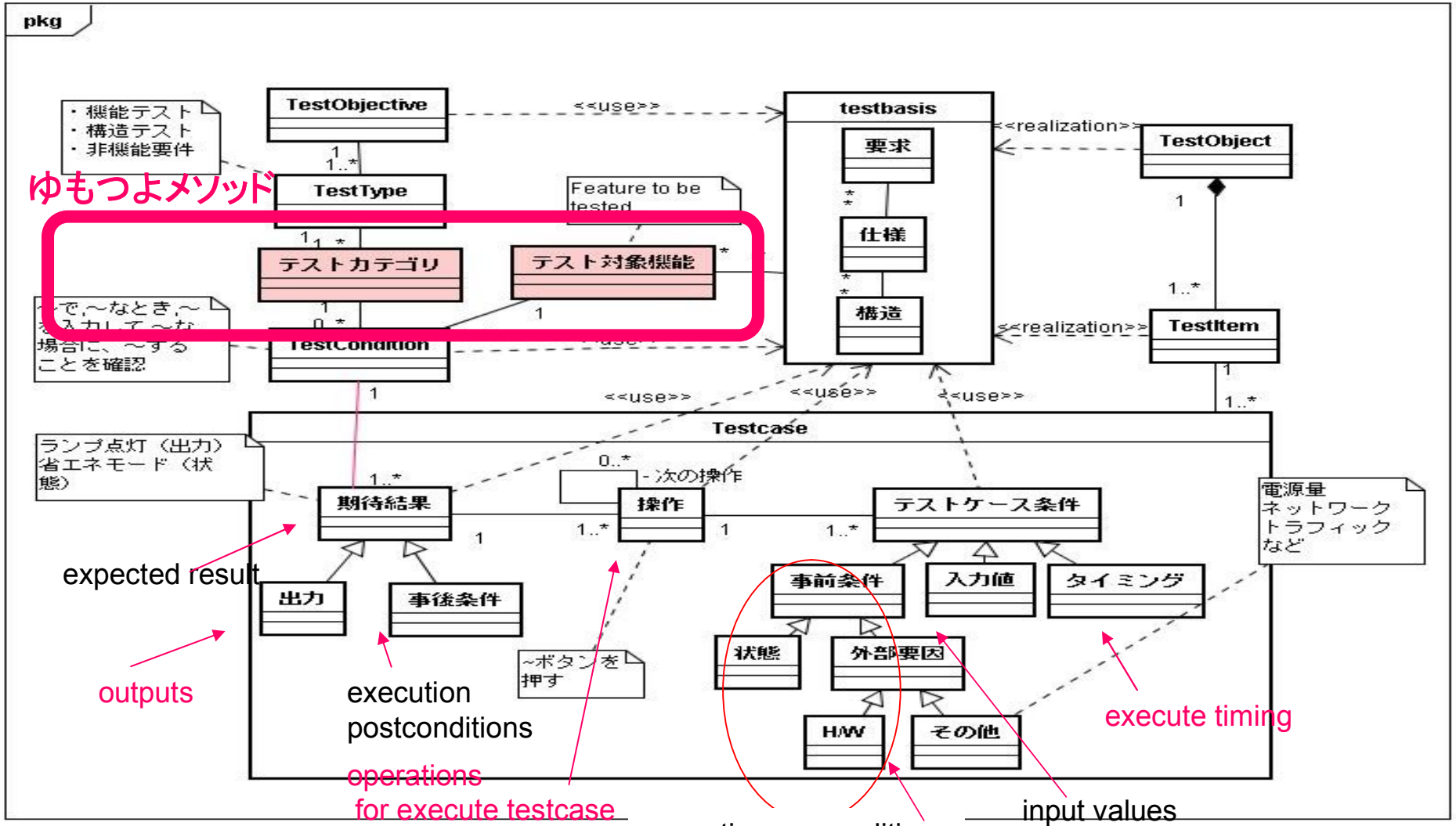
時期		目的	テスト対象項目	テストカテゴリ	テストベース
Ver 0.8	ST1	・仕様書に新規に追加された機能記述が正しく実装されていることの検証	新規機能	操作 外部出力 エラー 状態遷移	商品要求 仕様書
	ST2	・仕様書の記述に変更の無い機能記述が正しく実装されていることの検証	既存機能	操作 外部出力 エラー 状態遷移	商品要求 仕様書
	ST3	GUI・メッセージの文言や入力受付の確認	新規機能 既存機能	GUI/メッセージ	メニュー&メッセージ一覧表
Ver 0.9	ST4	以下省略			
	ST5				

テスト設計と繋げる



ゆもつよメソッド

テスト戦略のアウトプットと他の関係を整理

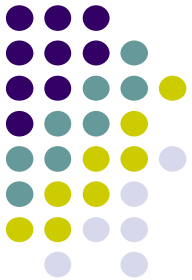


テスト戦略のアウトプットをベースにすること！

- 戦略で明らかにしたテストの目的、テスト対象アイテム、テストタイプを次のフェーズのインプットにすること

- 詳しくは

まとめ



- テスト戦略とは？
  - いろいろな「テスト戦略」を整理
    - プロダクト・プロジェクト・組織の分析
    - テストポリシー
    - 戦略の基本パターンとトレンド
    - 戦略オプション
    - 基本戦略
    - 個別戦略
- テスト戦略立案の進め方
  - テスト計画を進めるときの作業フロー
  - 全体テスト計画でのアプローチ作成までのステップ
  - 個別テスト計画の考え方のポイント
  - テスト設計と繋げる

Testing is such a creative work!

ありがとう
ございました