単なる仕様チェックから卒業してテスト技術力を高めていくために
～抑えておきたいキホンのキ～

池田 暁
NPO法人ASTER　理事
NaITE（長崎ＩＴ技術者会）代表

2018/11/22（木）於　長崎県美術館

自己紹介

自己紹介

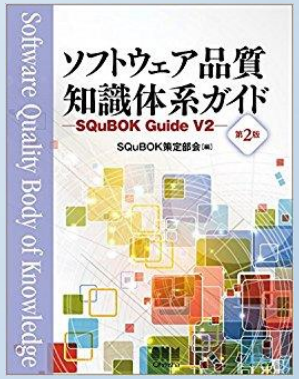
NPO法人ASTERやNaITE、SQiPやAFFORDD等に参画しています

情報通信系→医用系→自動車系と渡り歩いています
現在は社内のテストプロセス改善活動を取りまとめています

テストに関する書籍を執筆したり、イベントや勉強会にて講師をしたり、
コンテンツを作って公開したりしています

長崎県長崎市の出身です

今日は地元で発表できてとても嬉しいです！



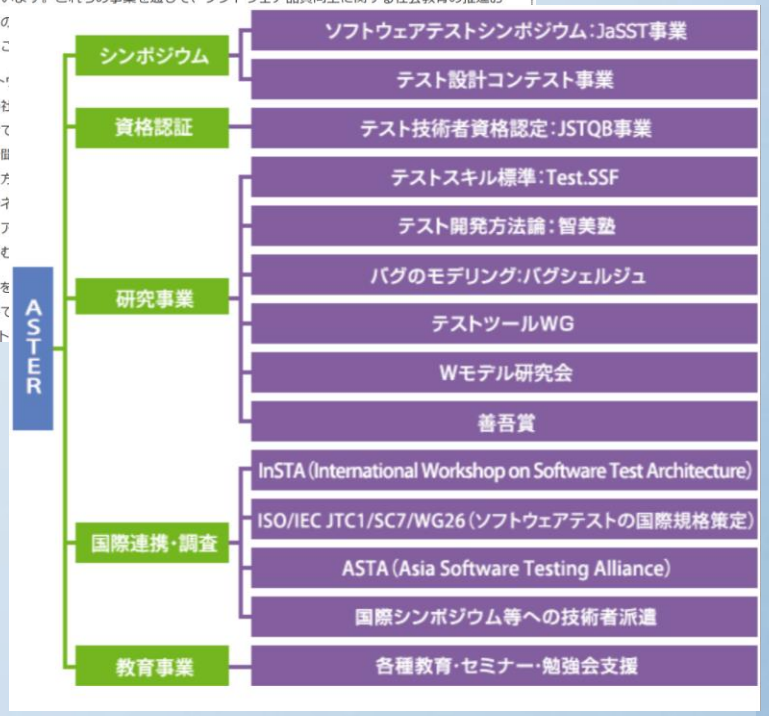
NPO法人ASTERとは

ソフトウェアテスト技術の普及振興に取り組んでいるNPO

全国各地でJaSST（ソフトウェアテストシンポジウム）やセミナーを開催しているほか、勉強会を支援します

全国各地でJSTQBによるテスト技術者認定資格試験を運営しています

その他、様々な調査活動や研究活動に取り組み、
様々なコンテンツを提供しています



是非ASTERの公式ページをご覧下さい！

http://aster.or.jp/

NaITE（長崎ＩＴ技術者会）とは

長崎出身や在住，かつて在住など，長崎になんらかの関わりや興味を持っている技術者の交流会としてスタートしました

長崎とありますが，興味がある方はどなたでも気軽にご参加いただけます

長崎現地にかかわらず，全国各地で勉強会やイベントを開催しています

SIG活動にも取り組み，マインドマップから始めるソフトウェアテスト　読書補助ガイド」や「はじめてのバグ票システム　～導入実践ガイド」を無料にて公開しています！

■NaITE公式サイト

http://naite.swquality.jp/

■Facebookページ

https://www.facebook.com/NagasakiITEngineers/

2015年4月に開始した 完全ボランティアの有志活動です！

NaITE（長崎ＩＴ技術者会）とは

#	開催日	内容	開催地
第19回	2017/01/21	数学を学ぼう　～数学の知識を利用したソフトウェアテスト～	神奈川
第20回	2017/04/08	プロセスモデル CMMI にまつわるちょっと深イイ話	神奈川
第21回	2017/04/22	PSP概説＆体験ワーク（おかわり）	東京
第22回	2017/05/20	はじめてのバグ票システム～導入実践ガイド　勉強会	神奈川
–	2017/07/01	Agile Japan 2017　長崎サテライト with NaITE	長崎
第23回	2017/07/16	Scrum入門＆Agile japan 2017 長崎サテライト参加報告	東京
第24回	2017/09/17	テストカタマリー　ワークショップ＆解説	神奈川
第25回	2017/11/11	欠陥モデリングワークショップ＆解説	神奈川

#	開催日	内容	開催地
第26回	2018/01/20	『ゆるファシ』体験ワークショップ	神奈川
–	2018/02/02	3rd 長崎 Software Quality and Development Gathering	長崎
第27回	2018/05/27	ケースメソッド体験ワークショップ	神奈川
第28回	2018/07/21	プロダクトオーナーシップ ＆ Agile Japan 2018 参加報告	神奈川
-	2018/09/15	Agile Japan 2018 長崎サテライト with NaITE	長崎

定期的に，長崎内外で勉強会やってます！

是非ご参加ください！ ＆ 一緒に企画しましょう！

http://naite.swquality.jp/

はじめに

よりよいテストのために必要な技術・知識

ドメインの専門技術・知識

テスト技術の専門技術・知識

開発の専門技術・知識

このあたりの話は、井芹さんや
実行委員の皆さんのセッションで得られます！

支える・繋がる

テストの基礎

本講演ではこのあたりの話をします

本日のゴール　（本日はテストの“キホンのキ”がテーマです）

- テストには日々取り組んでいるけれど、改めてどういった活動でどういった技術が必要なんだろう？
- 本セッションはテストに取り組み始めた方や**テスト初級者向け**に行います。テストといいつつ単なる仕様チェックになりがちな状態を卒業したい、テスト技術の全体像を掴みたい、テストの技術を高めるために勉強を始めたりスキルアップに取り組みたいと考えている方向けに、抑えておきたいキホンのキをお話しします。
  - **テストを行なう意義**を振り返り、**テストに必要な思考**をおさらいします
  - 単なる仕様チェックから抜け出すための手段の１つとして、**マインドマップを使ったテスト分析・設計の基本**を紹介します
  - **テスト技術の全体像**をイメージするために、構成要素を5つの軸から捉えます
  - そのうえでそれらの**勉強やスキルアップを始めるためのモデルケースやヒント**をお伝えします。
- テスト技術力を高めていくためには**テストそのものについて自分なりのイメージや地図を持つ**ことが重要です。是非この機会にイメージを掴み、具体的な行動を起こすヒントを得ましょう！

ここで問題です　–長崎の童歌

でんでらりゅうば
でてくるばってん
でんでられんけん
でーてこんけん
こんこられんけん
こられられんけん
こーんこん

でんでらりゅうアプリに関する仕様

指定のURLにブラウザでアクセスするとテキストボックスがひとつ、判定ボタンがひとつ表示される

でんでらりゅうの歌詞を入力し、判定ボタンを押すと、入力が間違っていないか判定される

判定結果が正しければ歌が流れる

判定結果が誤っていればＮＧと表示される

- さぁどういうテストをしようか？
- 解法を探し、手を打たなければ。。。
  - さっき同値分割/境界値分析は習いましたが。。。

地図がないと闇雲になる

地図がないと、むやみに解法を探すことになり不効率

道具を知らなければ、手を打てない

道具の種類と効果を把握しておかないと、適切な手を打てない

本日のお話で意識してほしいこととご注意

意識してほしいこと

- 自分の知っていることと知らないこと
- 自社/現場の状況と比較し、何が同じで何が異なるのか
- 自社/現場の現状況をよりよくするためにどのキーワードが使えそうか
- 自社/現場の先々をよりよくするためにどのキーワードにアンテナをたてておけばよいか

ご注意

- 個別の技術詳細については関連文献に委ねます（後で深掘り・調査できるようにポインタを示します）
- 「ソフトウェア品質」そのものについての議論は取り扱いません
- 本資料に登場する会社名、製品名などは、一般に各社の登録商標または商標です

初級者は明日からの改善のために使えるキワードードやヒントを得て下さい
中級者以上の方は、自分の知識の棚卸しや整理としてお聞き下さい

本日のアジェンダ　五つのキ

1. 【機】とらえ直そうテストの意義
2. 【規】テストに必要となる思考
3. 【紀】テストの分析設計～マインドマップを利用して
4. 【伎】テストの全体像をイメージするための5つの軸
5. 【企】始めてみよう　テストのスキルアップ
6. おわりに

1～3に重点を置いて話します

機：物事がおこるきっかけ

1.【機】とらえ直そうテストの意義

ソフトウェアテストってなんだろう？

ソフトウェアテストってなんだろう？

- ソフトウェアをテストすることだと言うのは簡単だが
  - ソフトウェアを動かしてみて，変な動きを確認すること？
  - 処理速度が速いかどうか試すこと？　　etc…

テストの経験は豊富なはず

- 今までの人生，散々テストを受けてきたはず
  - 学校の定期テスト，大学受験，車やバイクの免許，入社試験
- つまり，皆さんはテストのプロ

でも，説明するのは案外難しい

- 改めて考えてみると漠然としていませんか？
- だから，テストを行うことの意義も理解しにくいし，定義も理解できない
- 意義とイメージがつかめれば，テストの重要性も理解できる

ソフトウェアテストを行う意義

ソフトウェアテストを行なう意義

「ソフトウェアテストを行うと，ソフトウェアが作られていく過程で入り込んでしまう“バグ”を発見することができ，そのバグを開発者が修正することによって，ソフトウェアを利用者が安心して利用することができるようになる．」（Akira Ikeda, Mikio Suzuki, 2007）

- バグ（不良）を発見することができる
  - テストをやらないと，多くのバグは発見できない
  - テストで発見したバグを開発者がデバッグすることで，品質が上がる
- ソフトウェアを利用者が安心して利用することができる
  - 変な動きをしないからこそ毎日継続的に使い続けられる
  - テストは，実際に使う立場になって行うことが重要
    - 自らお金を払って買ってもいいと思える製品を作るという意識
    - テストは，お客様に安心感という価値を提供する

バグが与える影響　とある場面　～お客様の立場～

- ある日スマホを買ってきたら，次のような現象が出た
  - アンテナは3本立っているのに，メールが送信できない
  - メールを表示したら文字化け
  - アドレス帳に登録できない，何も操作しないのにデータが消える
  - 目覚ましが設定どおりに鳴らない
  - 操作中に無反応になったり，勝手に電源が切れる　　などなど・・・

- どう思うでしょうか？
  - あきれる，イライラする，腹が立つ
  - 場合によってはクレーム電話
  - 最悪，同じメーカのものは二度と買わないかもしれない
  - さらに，同じメーカの全然別の製品も買わないかもしれない

ソフトウェアにバグが残っている（混入している）と，
不快な気分になり，安心感もなくなり，不信感を持つ

バグが与える影響　とある場面　～メーカの立場～

- ゲームソフトの出荷後，フリーズなど致命的な問題が発見された
  - 回収・交換する場合，どういう対応が必要になるのか
    - ① ゲームソフトの回収
    - ② バグの調査と分析
    - ③ バグの修正
    - ④ 修正されたバグについてのテスト
    - ⑤ 新しいバージョンの製品の生産
    - ⑥ 新しいバージョンのお客様への送付

- 以下の例で考えてみる
  - 出荷本数：10万本，回収費用：1本700円
  - ②～④にかかる人数：15人（時給2000円）
  - ②～④にかかる時間：１ヵ月（8h×30d=240h）
  - 生産費用：1本500円，送付費用：1本700円

バグが与える影響　とある場面　～メーカの立場～

- ① ゲームソフトの回収
  - 10万本　×　送料700円　＝　7,000万円
- ② バグの調査と分析～ ④ 修正されたバグについてのテスト
  - 15人　×　時給2,000円　×　240時間　＝　720万円
- ⑤ 新しいバージョンの製品の生産
  - 10万本　×　一本あたり500円　＝　5,000万円
- ⑥ 新しいバージョンのお客様への送付
  - 10万本　×　送料700円　＝　7,000万円

- ①～⑥までの合計
  - 7,000万円 + 720万円 + 5,000万円 + 7,000万円 ＝ 約２億円！
  - このほか，損害賠償や，新聞/TV広告，電話窓口の設置などのコスト
  - コスト以外にも，不買運動や風評被害，最近なら批判Blogなどブランドに影響

たった一つのバグが数億円の損害を与えるほか
ブランドに大きな影響を与えることもある！

（参考）「史上最悪のソフトウェアバグ」ワースト10

1962年7月22日──火星探査機『マリナー1号』：

- マリナー1号は**打ち上げ時に予定のコースを外れた**が，これは飛行ソフトウェアのバグが原因だった。地上の管制センターは大西洋上でロケットを破壊した。事後調査により，鉛筆で紙に書かれた数式をコンピューターのコードに置き換えるときにミスが起き，これが原因でコンピューターが飛行コースの計算を誤ったことが判明した。

1982年──旧ソ連のガス・パイプライン：

- シベリアを横断するガス・パイプラインの管理に旧ソ連が購入したカナダ製のコンピューターシステムに，米中央情報局(CIA)のスパイがバグを仕掛けたことがあるという。旧ソ連は当時，米国の機密技術を密かに購入しようと──または盗み出そうと──しており，このシステムを入手したのもその一環だった。だが，計画を察知したCIAはこれを逆手にとり，旧ソ連の検査は問題なく通過するが，いったん運転に入ると機能しなくなるように仕組んだとされる。この結果**起きたパイプライン事故は，核爆発以外では地球の歴史でも最大規模の爆発だった**という。

1985～1987年──セラック25：

- 複数の医療施設で放射線治療装置が誤作動し，過大な放射線を浴びた患者に死傷者が出た。セラック25は2種類の放射線──低エネルギーの電子ビーム(ベータ粒子)とX線──を照射できるよう，既存の設計に「改良」を加えた治療装置だった。セラック25では電子銃と患者の間に置かれた金属製のターゲットに高エネルギーの電子を打ち込み，X線を発生させていた。セラック25のもう1つの「改良」点は，旧モデル『セラック20』の電気機械式の安全保護装置をソフトウェア制御に置き換えたことだった。ソフトウェアの方が信頼性が高いとの考えに基づく判断だった。
- しかし，技術者たちも知らなかった事実があった──セラック20およびセラック25に使われたOSは，正式な訓練も受けていないプログラマーが1人で作成したもので，バグが非常にわかりにくい構成になっていたのだ。「競合状態」と呼ばれる判明しにくいバグが原因で，操作コマンドを素早く打ち込んだ場合，セラック25ではX線用の金属製ターゲットをきちんと配置しないまま高エネルギーの放射線を照射する設定が可能になっていた。これにより**少なくとも5人が死亡し，他にも重傷者が出た**。

WIRED「「史上最悪のソフトウェアバグ」ワースト10を紹介」より引用

（参考）「史上最悪のソフトウェアバグ」ワースト10

1988年──バークレー版UNIX(BSD)のフィンガーデーモンによるバッファー・オーバーフロー

- 最初のインターネットワームとなった通称『モーリス・ワーム』は，バッファー・オーバーフローを悪用し，**1日足らずで2000台から6000台のコンピューターに感染**した。原因となったのは，標準入出力ライブラリー・ルーチン内の「gets()」という関数のコードだ。「gets()」関数はネットワーク越しにテキストを1行取得するように設計された。しかし，残念ながら「gets()」関数は入力を制限するようには作られていない。そのため，あまりにも大きな入力があった場合には，接続可能なあらゆるマシンをワームが占拠する元凶になった。
- プログラマーは「gets()」関数を使用コードから排除することで問題に対処しているが，C言語の標準入出力ライブラリーからこれを削除することは拒否しており，この関数は現在も存在している。

1988～1996年──『ケルベロス』の乱数生成アルゴリズム

- ケルベロスは暗号を使ったセキュリティーシステムだが，乱数発生器に与えるシード(種)が適切でなく，真にランダムな乱数が生成されていなかった。その結果，ケルベロスによる認証を用いているコンピューターについて，**非常に簡単な方法で侵入可能な状態が8年間にわたって続いた**。このバグが実際に悪用されたかどうかは，今も定かではない。

1990年1月15日── 米AT&T社のネットワーク停止

- 米AT&T社の長距離電話用交換機『4ESS』を制御する最新版のソフトウェアにバグが入りこんだ。このため，4ESSは隣接するマシンの1つから，ある特定のメッセージを受け取るとクラッシュするようになってしまった──そしてそのメッセージとは，クラッシュした交換機が復帰した際に，隣接する交換機に送信するものだった。
- ある日，ニューヨークの交換機がクラッシュし再起動した。するとそれが原因で隣接する複数の交換機がクラッシュし，これらの交換機が再起動すると隣接する複数の交換機がさらにクラッシュし，この現象が延々と続いた。しばらくすると，114台の交換機が6秒ごとにクラッシュと再起動を繰り返すようになった。この影響で**およそ6万人の人々が9時間にわたって長距離通話サービスを利用できなくなった**。修復のため，技術者たちは1つ前のソフトウェアをロードした。

1993年──インテル社製『Pentium』(ペンティアム)による浮動小数点数の除算ミス

- 米インテル社が大々的に売り出したPentiumチップが，特定の浮動小数点数の除算で誤りを引き起こした。たとえば，4195835.0／3145727.0を計算させると，正しい答えの1.33382ではなく1.33374となる。0.006%の違いだ。
- 実際にこの問題の影響を受けるユーザーはごくわずかだったが，ユーザーへの対応から，同社にとって悪夢のような事態につながった。概算で300万～500万個の欠陥チップが流通していた状況で，インテル社は当初，高精度のチップが必要だと証明できる顧客のみをPentiumチップの交換対象とした。しかし，最終的にインテル社は態度を改め，不満を訴えるすべてのユーザーのチップ交換に応じた。この欠陥は結局，インテル社に**約4億7500万ドルの損害を与えた**。

WIRED「「史上最悪のソフトウェアバグ」ワースト10を紹介」より引用

（参考）「史上最悪のソフトウェアバグ」ワースト10

1995年／1996年──『Ping of Death』

- [ピング・オブ・デス，不正なピングパケットによる攻撃]分割送信されたIPパケットの再構成を行なうコードのチェックとエラー処理が不十分だったため，インターネット上の好きな場所から不正な形式のピングパケットを飛ばすことで，**さまざまなオペレーティング・システム(OS)をクラッシュ**させることができた。影響が最も顕著に現れたのはウィンドウズ搭載マシンで，この種のパケットを受け取ると，「死のブルー・スクリーン」と呼ばれる青い画面を表示して動作が停止してしまう。しかしこのバグを利用した攻撃は，ウィンドウズのみならず，マッキントッシュやUNIXを使ったシステムにも多くの被害をもたらした。

1996年6月4日──『アリアン5』フライト501

- 欧州宇宙機関の開発したロケット，アリアン5には，『アリアン4』で使われていたコードが再利用されていた。しかし，アリアン5ではより強力なロケットエンジンを採用したことが引き金となり，ロケットに搭載された飛行コンピューター内の計算ルーチンにあったバグが問題を起こした。エラーは64ビットの浮動小数点数を16ビットの符号付き整数に変換するコードの中で起こった。アリアン5では加速度が大きいため，64ビット浮動小数点で表現される数がアリアン4のときよりも大きくなってオーバーフローが起こり，最終的には飛行コンピューターがクラッシュしてしまった。
- フライト501では，最初にバックアップ・コンピューターがクラッシュし，それから0.05秒後にメイン・コンピューターがクラッシュした。その結果，エンジンの出力が過剰になり，**ロケットは打ち上げ40秒後に空中分解してしまった**。

2000年11月──パナマ国立ガン研究所

- 米マルチデータ・システムズ・インターナショナル社(本社ミズーリ州)が製作した治療計画作成用ソフトウェアを使っていたパナマの国立ガン研究所で，放射線治療で照射する放射線量の計算を誤る一連の事故が起きた。
- マルチデータ社のソフトウェアでは，健康な組織を放射線から守るための「ブロック」と呼ばれる金属製のシールドの配置を，コンピューターの画面上に描いて決めるようになっていた。しかし，同社のソフトウェアではシールドが4個しか使えなかったにもかかわらず，パナマ人の技師たちはこれを5個使いたいと考えた。
- 技師たちは，真ん中に穴を持つ1個の大きなシールドとして，5個のシールドをまとめて表示させれば，ソフトウェアをだますことができることを発見した。だが，そうした配置にした場合，穴の描き方によってこのソフトウェアが返す計算結果が違ってくることにはまったく気づいていなかった。ある方向に向けて描くと正しい照射量が計算されるが，違う方向に描くと必要な照射量の最大2倍の量を推奨してきたのだ。
- 少なくとも**8人の患者が死亡**し，さらに20人が過剰照射によって深刻な健康被害を受けたとみられている。技師たちは，コンピュータによる計算結果を手作業で再チェックする法的義務を負っていたため，**殺人罪で起訴されることになった**。

WIRED「「史上最悪のソフトウェアバグ」ワースト10を紹介」より引用

ソフトウェアテストを行う意義

- ソフトウェア製品に
  - バグが残っている（混入している）と，不快な気分になり，安心感もなくなり，不信感を持つ
  - 混入したたった一つのバグが数億円の損害を与えるほか，ブランドに大きな影響を与えることもある

ソフトウェアテストを行なう意義

「ソフトウェアテストを行うと，ソフトウェアが作られていく過程で入り込んでしまう“バグ”を発見することができ，そのバグを開発者が修正することによって，ソフトウェアを利用者が安心して利用することができるようになる．」（Akira Ikeda, Mikio Suzuki, 2007）

はて？
なんだか意義が足りないような･･･？

ソフトウェアテストを行う意義

テストはお客様，企業双方にメリットがある！

ソフトウェアテストを行なう意義

「ソフトウェアテストを行うと，ソフトウェアが作られていく過程で入り込んでしまう“バグ”を発見することができ，そのバグを開発者が修正することによって，ソフトウェアを利用者が安心して利用することができるようになる。
　また，出荷後にバグが出ないことでソフトウェアの回収と修正に必要なコストを削減し，企業のイメージ低下，ひいては倒産を防ぐことができる。」
（Akira Ikeda, Mikio Suzuki, 2007）」（Akira Ikeda, Mikio Suzuki, 2007）

- たった一個のバグが，お客様に不利益を与えるどころか，企業の業績を左右することもある
- 場合によっては企業の倒産により職を失ったり，企業や社会に損害を与えたことで，個人に損害賠償責任が発生することも

ソフトウェアテストにしっかりと取り組むということは，
バグというモンスターから
実際のお客様のみならず，企業や社会，
そして自分や家族を守ることでもある

（参考）テストとは？（ISTQB-FL）

- テストとは何か？
  - テストとは何か，に対する一般的な認識は，テストを実施すること，すなわち，ソフトウェアの実行であることが多い。ソフトウェアの実行はテストの活動の一部であり，全部ではない。
  - テストの活動は，テスト実施の前後にも存在する。例えば，計画，コントロール，テスト条件の選択，テストケースの設計と実行，実行結果のチェック，テスト完了基準の検証，テストプロセスやテスト対象システムに関する報告，テストのまとめや終了作業（テストフェーズが完了した後）がある。テストにはドキュメント（ソースコードを含む）レビューや，静的解析を実施することも含む。
  - 動的テストと静的テストは，方法は違っても同じような目的のために使え，テスト対象のシステムだけではなく，開発やテストのプロセス改善のための情報提供もできる。

- テストの目的
  - 欠陥を摘出する。
  - 対象ソフトウェアの品質レベルが十分であることを確認する。
  - 意志決定のための情報を示す。
  - 欠陥の作りこみを防ぐ。

ISTQBテスト技術者資格制度　Foundation Level シラバス 日本語版 Version 2011.J02　から引用

【演習】調べてみようソフトウェアテストの定義

○○○氏の定義

IEEE・ISO・JIS等の規格での定義

職場やプロジェクトでの定義

規：行動や判断のよりどころとなる基準。

2.【規】テストに必要となる思考

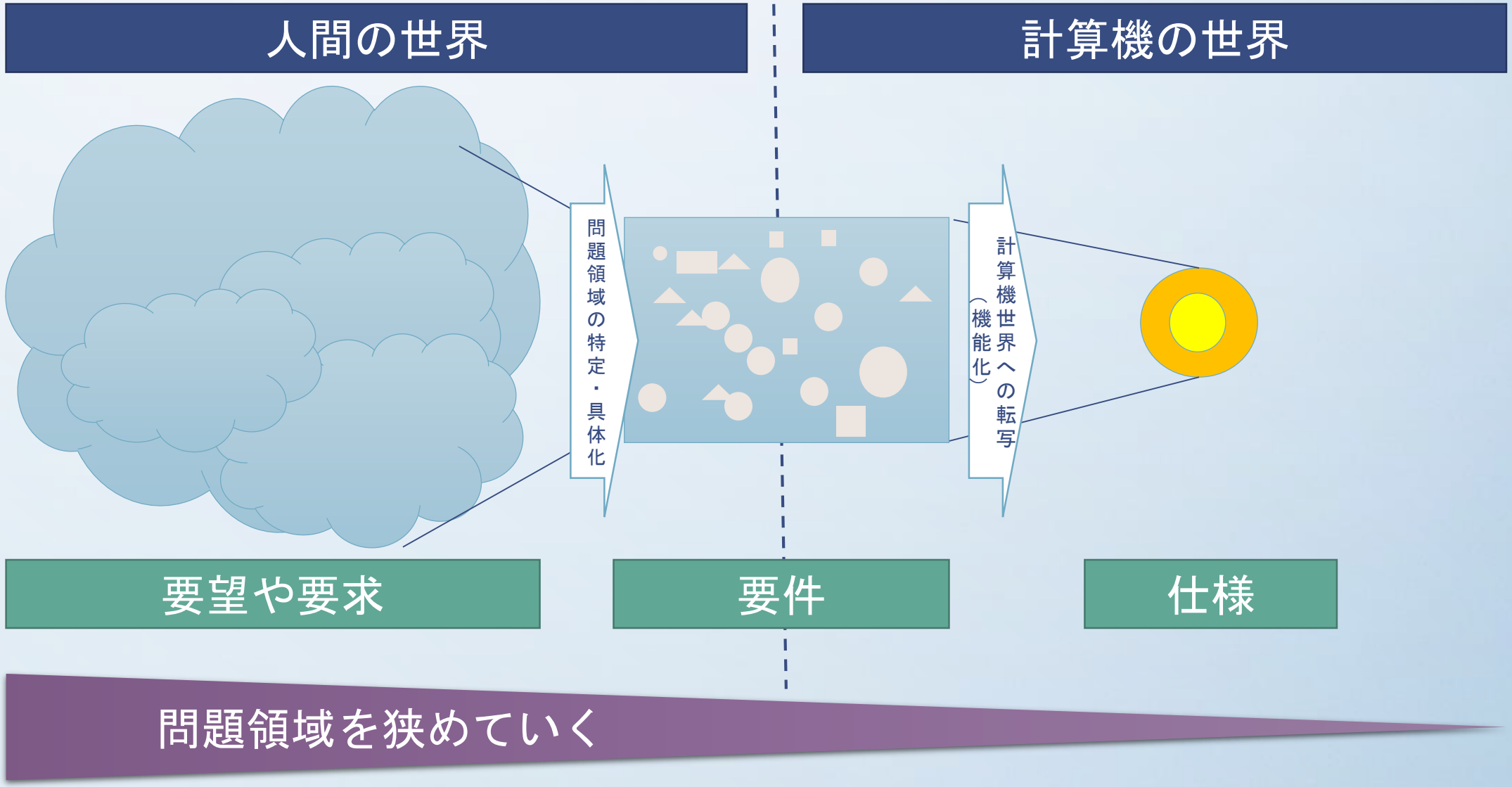
ソフトウェア開発で必要となる基本的な思考

設計のための思考

テストのための思考

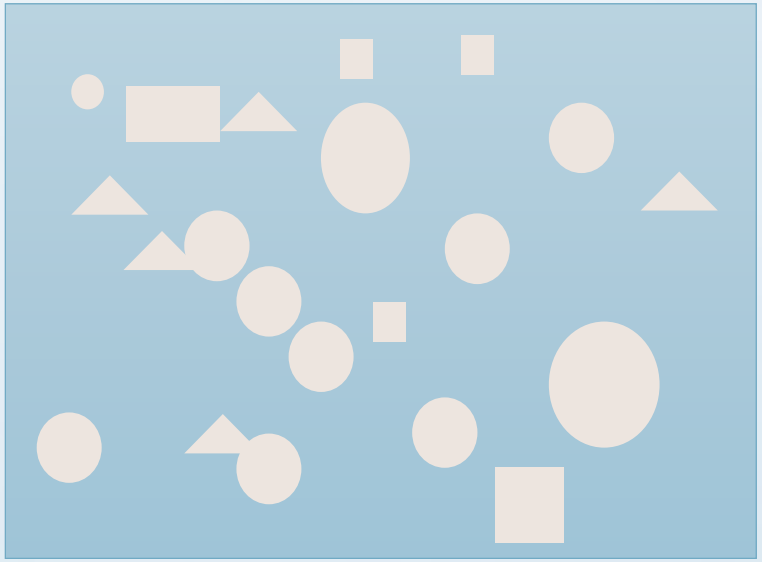
※ご注意
様々な考え方があると思いますが、
本講演ではわかりやすさを重視して、このように単純化してお話しします

設計とは問題領域を狭めていく行為

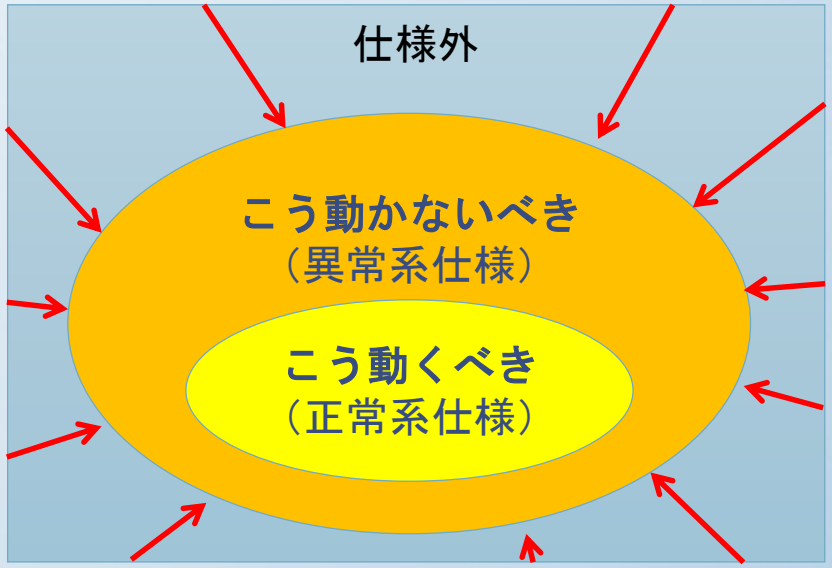


設計時の思考

設計では要件を，計算機世界上で「こう動くべき」「こう動かないべき」に分類・具体化・定義することで問題領域を狭めていく（計算機の振る舞いを定義する→仕様化）



「こう動くべき」「こう動かないべき」に定義していくことで，計算機として扱う問題領域を狭めていく

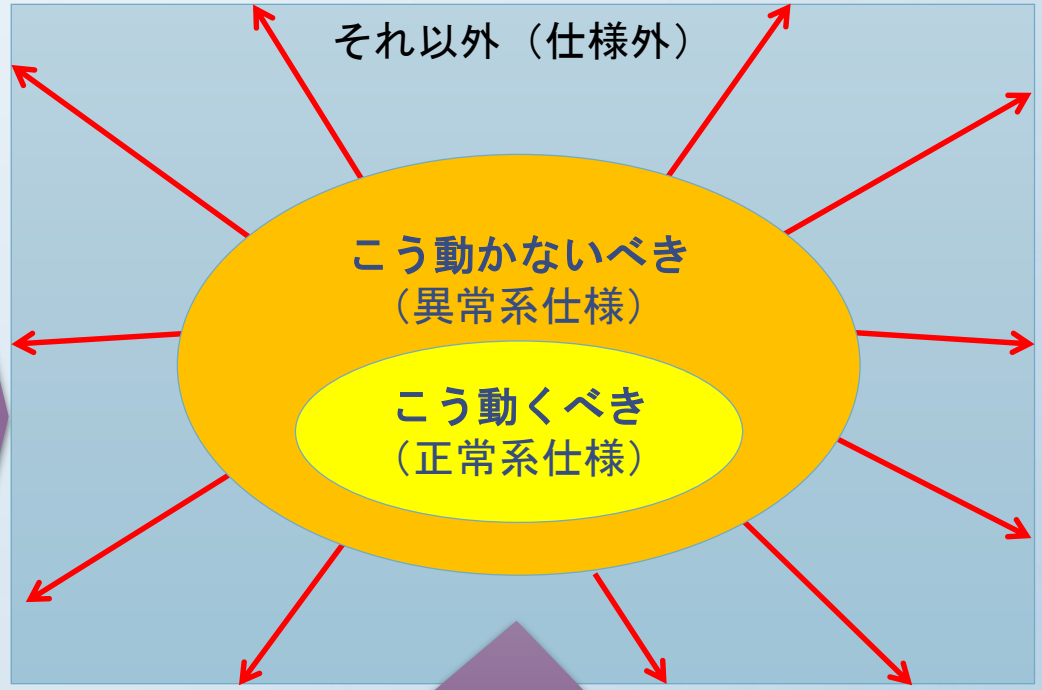


テスト時の思考

テストは仕様化された事柄について「その通りに動くか」を確認するだけでは不十分で，「それ以外で何も起きないのか」も確認せねばならない。

仕様を確認するだけでは
単なる動作チェック

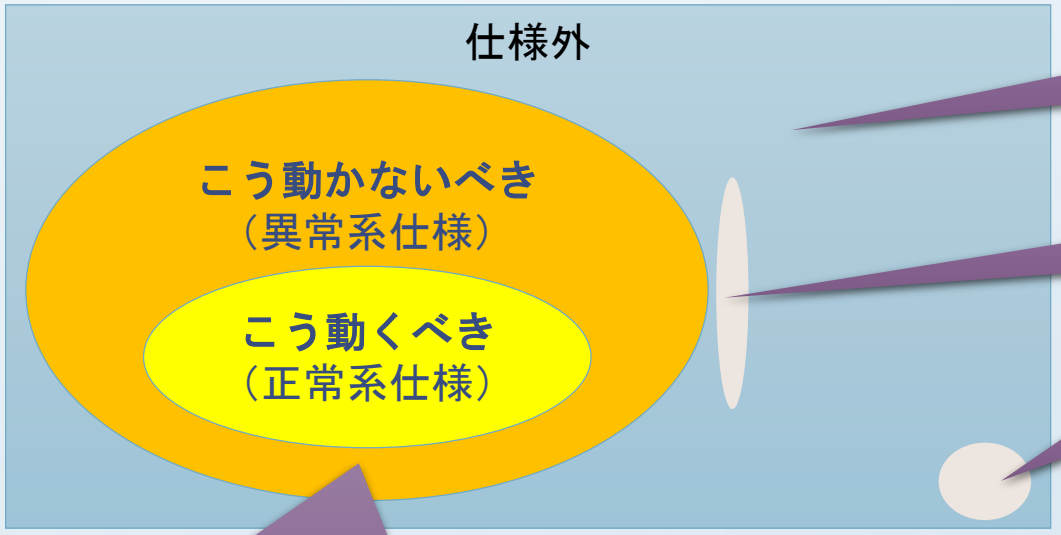
「仕様がその通りに動くか」に加え，「それ以外で何も起きないか」問題領域を広げて探索していく



テストではむしろ
「仕様外」を扱うことが重要

テストすべきことの発想（認知）

- チェックすべき対象は「仕様として書かれていること」，テストすべき対象は「仕様外」と捉えるとわかりやすくなる
- 仕様書に書かれていない「仕様外」をどれだけ発想・認知できるかが勝負



仕様書に書かれていないため，
思いつかなければならない

仕様領域に近いところは
比較的仕様書から発想しやすい

仕様から遠いところは
仕様書によらない発想も必要

「行間を読む」ということ

仕様書に書かれていることから
チェックすべきこと発想すれば良い（容易）

この仕様外に対応していくための1つの手段として「テスト分析」と「テスト設計」があり，テストすべき事柄を発想して整理する手法として「テスト分析設計手法」がある

テストにおける思考のまとめ

- テストを考えるとき，設計時とは違う思考パターンであることを理解しておく（逆方向の思考である）
  - 設計は問題領域を狭めていく思考
  - **テストは問題領域を広げていく思考**
    - テストは仕様化された事柄について，「仕様化されたものがその通りに動くか」と「それ以外で何も起きないのか」と問題領域を広げていく
    - **仕様を確認するのはチェック，仕様外を確認するのをテストと考えると良い**
- これらの一連の行為として「テスト分析」と「テスト設計」があり，テストすべき事柄を発想して整理する手段として「テスト分析設計手法」がある

テストに取り組む場合，思考を切り換える必要がある
いわゆる「帽子のかぶり直し」である

【演習】抑えてみようテストの思考

【仕様】

こう動くべき

こう動かないべき

それ以外

紀：筋道や順序を追って整理・記録する

3.【紀】テストの分析設計
～マインドマップを利用して～

皆さんどのようにテストケースを作っていますか？

(仕様例)
ボタンを押すと音が出る

(テストケース例)
ボタンを押すと音が出る<u>ことを確認</u>

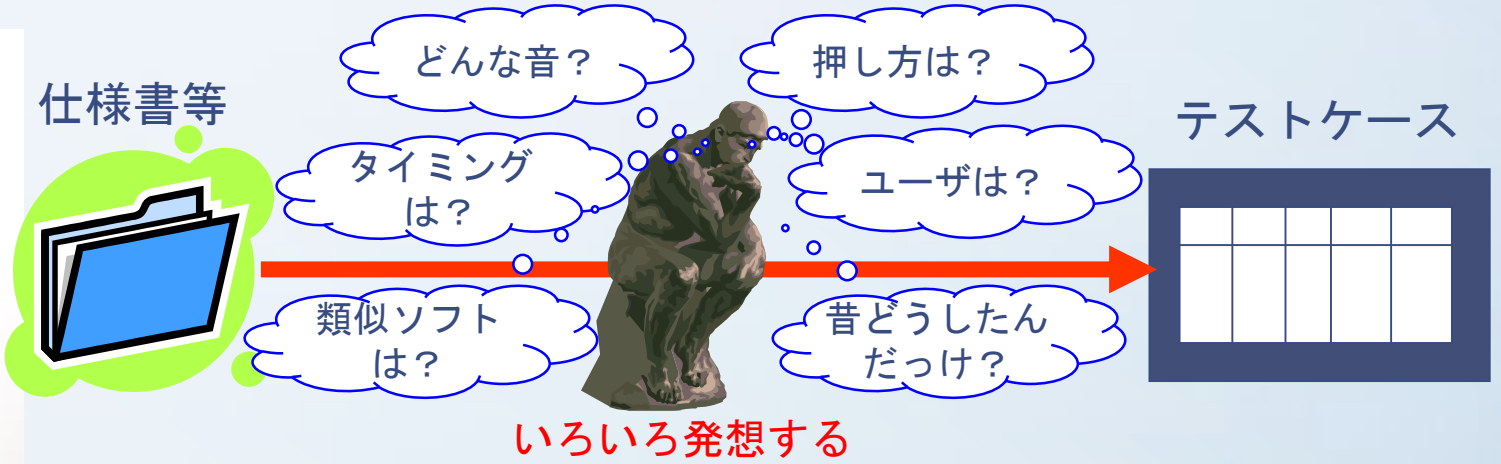
初級者の悩み
- テストケースのヌケが多い！
- 異常系のテストケースが抜ける！
- 機能を組み合せを考慮したテストケースがかけない！
- テスト技法の使いどころがわからない！
- 組織で積み上げられたノウハウが活用できない！
- 自分の経験が再利用できない！

などなど……

語尾を付け足して完成させる，**<u>単なるチェック止まり！</u>**

皆さんどのようにテストケースを作っていますか？

上級者



テストケースを書く前に，思考を発散させながら，かつＭＥＣＥを意識して考える．なので，初級者に比較してテストケースの抜けが少なくなる！
また，弱点をつくようなテストケースが作成できる！

上級者はテスト観点を発想/検討したうえで戦略的にテストケースを作成する
- テストを行うにあたって，テスト観点をしっかりと考える
  「機能」「プラットホーム」「エンドユーザ」「ドメイン特性」「組織のノウハウ」など
- テスト観点の階層や関連，組み合わせを考える
- 不適切な仕様（仕様バグ）は摘出　→　設計部門に確認・修正依頼
- テスト観点全体の構成を俯瞰して，重み付けや実行順番など考える

などなど……

仕様書に書かれていないことも発想する

実に多くを考えることが必要であるが…

考えるっていっても
頭の中だけじゃ
大 変 だよな

とりあえず，
テストケース表を使って
考えようかな？

どうせ最終的には
エクセルの表になる
わけだし

ただ、テストケースの表現形式を使ってテスト設計を行うのは難しい

では，どういった記法(ツール)を使うとよいのだろうか？

表で観点を出す，
つまりブレストは
難しいな～

試行錯誤できる
記法がいいな

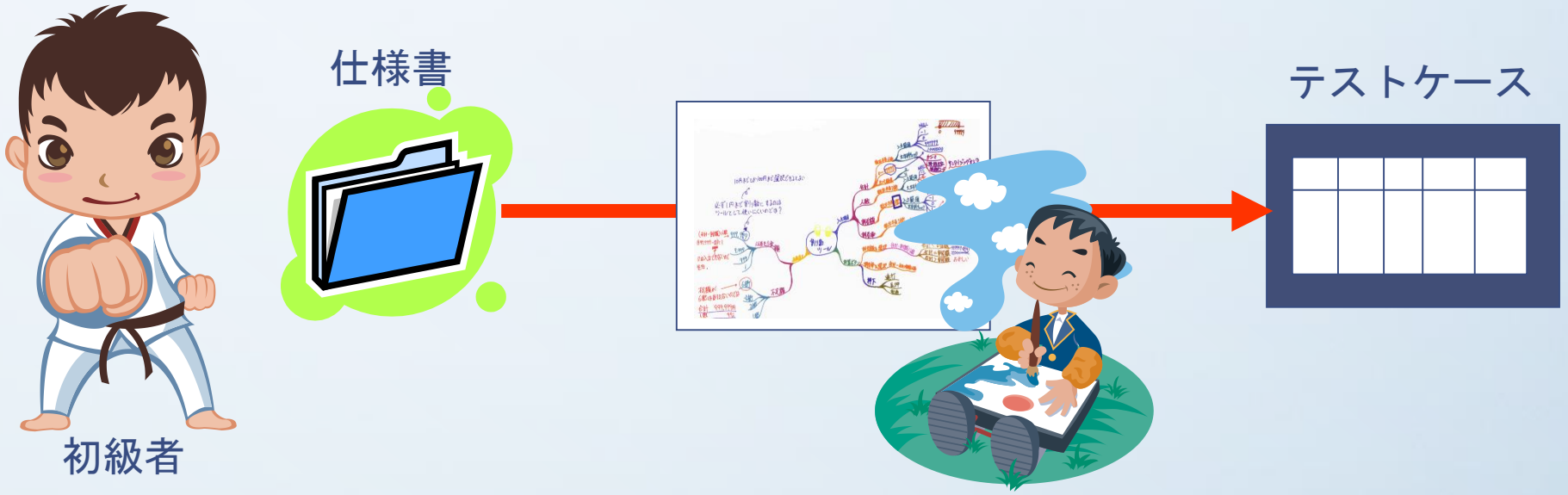
できれば構造化
しやすいものが
いいかも

整理するためには
全体を俯瞰
できなくちゃ

でも，発想力を刺激し
てくれるものでないと，
観点が抜けちゃう！

じゃぁ，そのような特徴をもつ
発想支援ツール，マインドマップを
使ってみたら？

マインドマップの利用イメージ（コンセプト）



マインドマップを書くことで，
- 単純な仕様の転記がなくなる
- 仕様書に書かれていないことにも思考が誘導される
- 「考える」行為を明確に実行できる

つまり，「テストの思考」を実践するための道具としてマインドマップを活用する

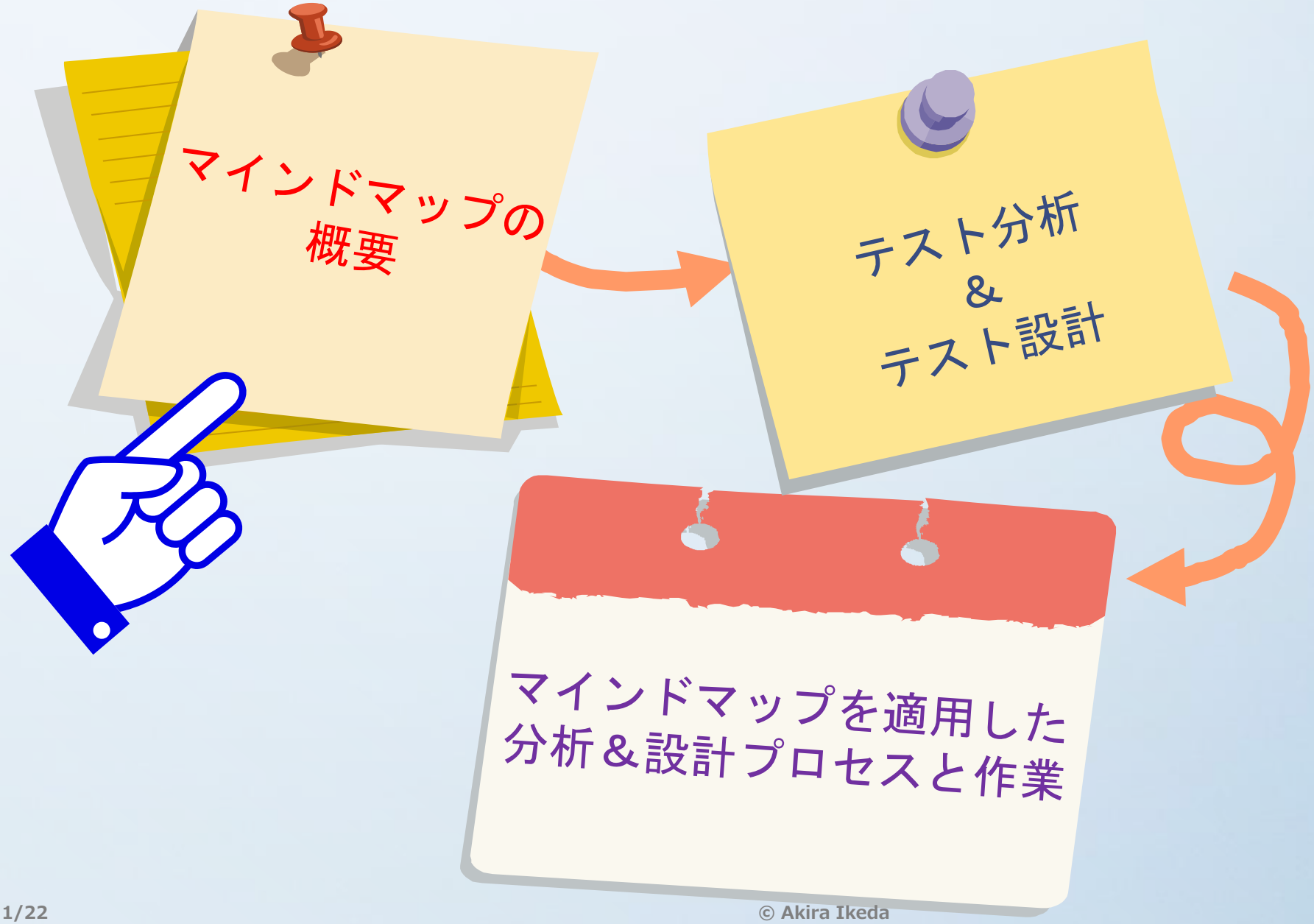
テストケースの品質に大きな影響を与え，
かつ，仕様外への発散思考が特に重要な
- テスト分析
- テスト設計

にマインドマップを使ってみよう

- マインドマップの概要
- マインドマップの適用を見据えた
  テスト分析＆テスト設計の作業と勘所
- マインドマップを使った作業手順

を説明します

単なる仕様チェックから卒業するための
手段の1つとして持ち帰って下さい



マインドマップとは？

- トニー・ブザンにより考え出された図解技法
  - 脳の仕組みを取り入れたもの
  - 思考に沿って描いていく
  - 図を取り入れる
  - 自分の深層意識にアクセスする

Wikipediaによる解説

　表現したい概念の中心となるキーワードやイメージを図の中央に置き，そこから放射状にキーワードやイメージを繋げていくことで，発想を延ばしていく図解表現技法。

　この方法によって複雑な概念もコンパクトに表現でき，非常に早く理解できるとされ，注目され始めている。

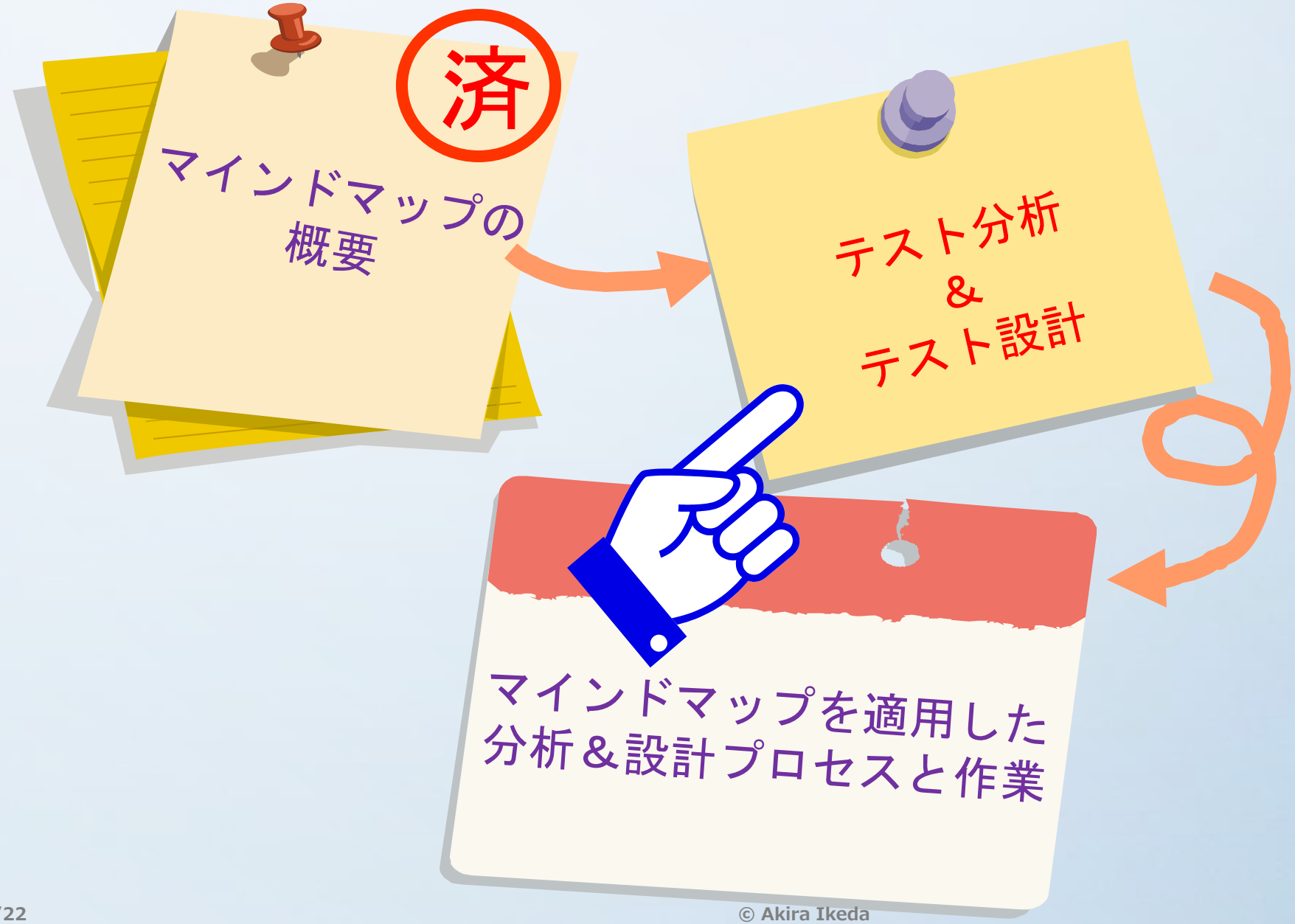
　人間の脳の意味ネットワークと呼ばれる意味記憶の構造によく適合しているので，理解や記憶がしやすい。

また本来は紙とペンで描くものだが，コンピュータ上で描くための専用ソフトウェアもいくつか存在する。

マインドマップの特徴の一例

バードビュー	•全体を俯瞰し易い
MECE	•項目それぞれが重複することなく，全体集合として漏れがない
学習が容易	•基本的なルールは単純で，紙とペンがあれば始められる
半構造	•フリーなルールであるために，柔軟に構造を変更可能
発想力が刺激される	•描いているうちに他の項目との関連などから新たな発想が生まれやすい •深層意識へのアクセスし，情報を引き出す
思考の流れの見える化	•中心から外に対して思考が放射的に広がる

これらの特徴を上手く生かそう♪



「テスト分析」と「テスト設計」

テストケースの作成 → テストの実行

作業/概念で分ける

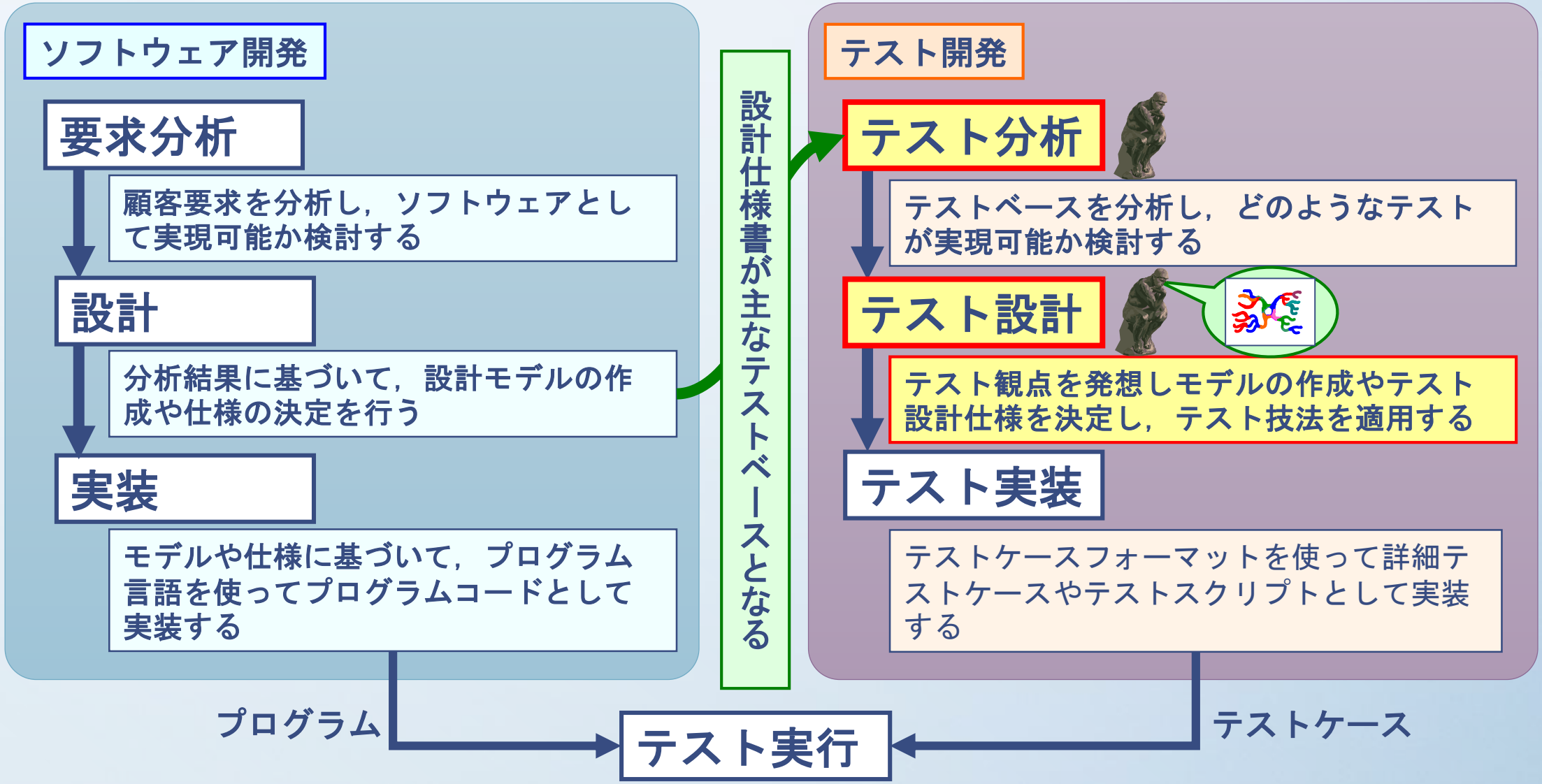
テスト分析 → テスト設計 → テスト実装 → テスト実行 → テスト報告

テスト分析	テスト設計	テスト実装	テスト実行	テスト報告
仕様等の理解・整理・検討・テスト観点の目付け	テスト観点の発想・検討・整理，テスト技法の適用	詳細テストケース・スクリプト等の作成（適切なフォーマットに表現）	テスト実装により作成されたテストケースを実行し，ログを取る	テストの結果やログを整理し、当該テストレベルでのテスト活動を評価する

ここにマインドマップを使うぞ！

※注意
本講演ではテスト観点の発想はテスト設計で行ないますが、
他の手法ではテスト分析で行なう場合もあります。

開発作業とテスト作業のおおまかな対比イメージ



テスト分析の作業と勘所（1）

設計仕様やテストへの要求の理解・整理・検討

- 知らないものはテストできない！
- 設計仕様に関する思想や意図，構造や構成物を理解する
  - どういったものを作ろうとしているかのイメージをつかむ
    - 設計者が特に重要と考える部分は，テストでも重要度が高いことがほとんど
    - 顧客先でどのように利用されるのか
  - テストへの要求や制約、品質目標等の認識

テスト設計の手がかりを作る

- テストすべき“要求”や“要件”，“仕様”や“機能”
- ソフトウェアのウィークポイントの目付け
- 過去のテストの経験からひっかかるところ、気配を感じるところ
- その他の疑問点
  - 疑問が生じる箇所は仕様がこなれていない可能性がある

テスト分析の作業と勘所（2）

テスト分析 テスト設計

仕様の抜け漏れの発見と修正へのアクション

- 仕様の欠陥を発見したら，設計部門にフィードバックし，仕様の高品質化をはかる
  - 仕様書の高品質化は即ちテストベースの高品質化であり，そこから生み出されるテスト設計仕様やテストケースの高品質化に寄与する
    - 良い仕様書からは，良いコードと良いテストケースが生まれる
- 欠陥以外でも、疑問を生じたことは積極的に問い合わせる

テスト戦略へのフィードバック

- 分析結果の情報をテストの戦略や計画に反映

仕様が理解できなかったり抜け漏れが多い場合，開発者に見直しを要請する
テスト分析の作業はある意味においてテストの立場からのレビュー行為でもある
どれだけ正しい仕様を深く理解できるかも良いテスト設計への鍵

テスト設計の作業と勘所（1）

テスト観点の発想

- 「なにをテストしよう」「どうテストしよう」が思いつかないと話が始まらない
  - 仕様書の分析結果や過去の経験，組織ノウハウから
  - テストカテゴリの利用
    - Myersの14のシステムテスト・カテゴリ
      - ボリューム，ストレス，効率，ストレージ，信頼性，構成，互換性，設置，回復，操作性，セキュリティ，サービス性，文書，手続き
    - ISO/IEC 25010の品質特性
      - システム／ソフトウェア製品品質
        - 機能適合性，性能効率性，互換性，使用性、信頼性、セキュリティ、保守性、移植性
      - 利用時の品質
        - 有効性、効率性、満足性、リスク回避性、利用状況網羅性
    - 組織に蓄積されたガイドワード
    - テスト設計を進めるなかで得られる新たな発想

テスト設計の作業と勘所（2）

テスト観点の剪定・整理

- テスト観点には重要度や優先度が存在する
  - 全てのテストは多くの場合やりきれないため，テスト観点に優先度をつける
- テストする必要のない観点や優先度・重要度の低い観点は切り落とす
  - 切り落としたテスト観点とその理由は記録に残しておくこと
  - 無秩序に発想したテスト観点を整える
    - 観点の階層や観点間の関連を検討する

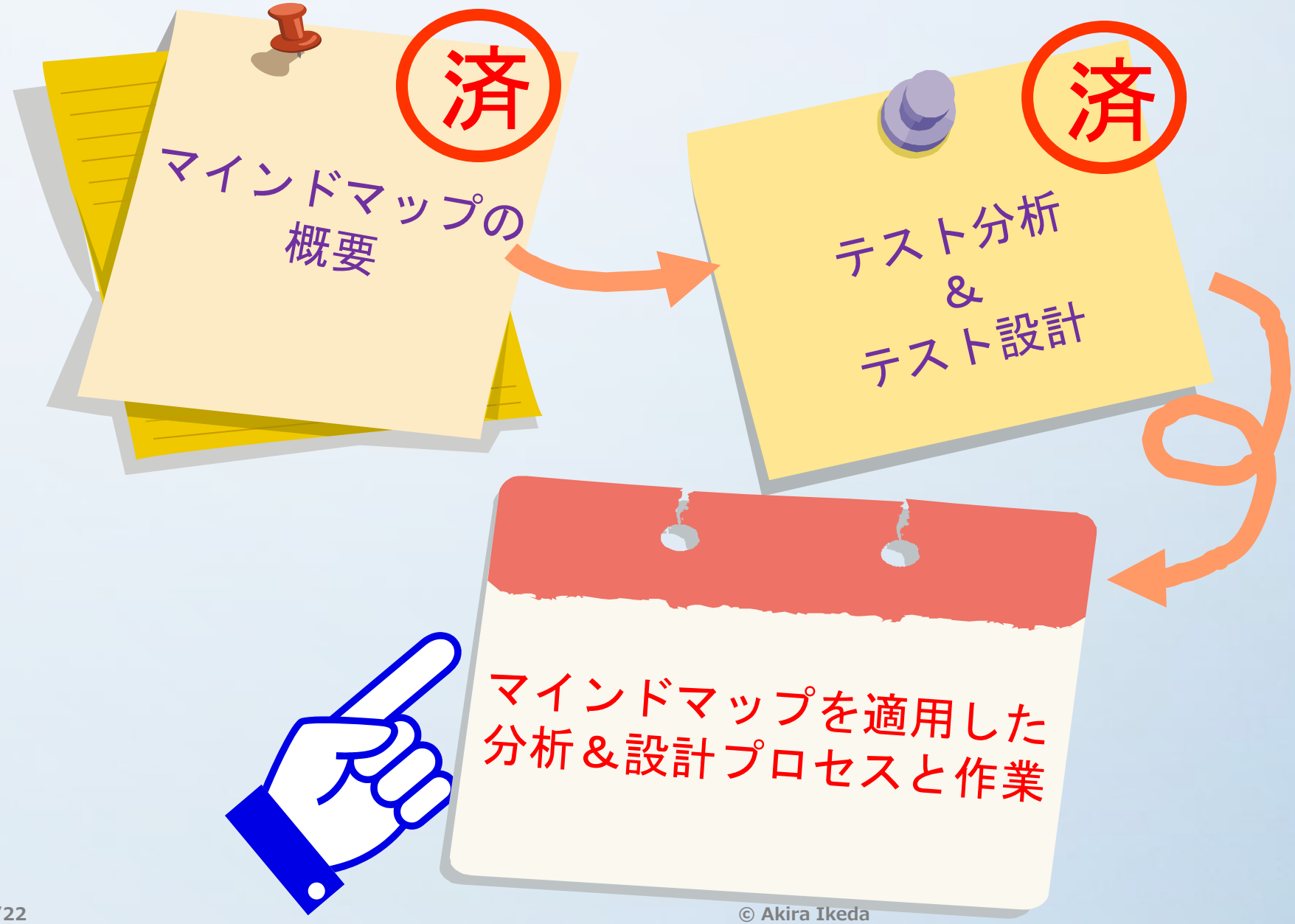
テスト技法の適用

- 観点モデルの観点要素に対して対応するテスト技法をアサインし，実行する

テスト戦略へのフィードバック

- テスト設計結果の情報をテストの戦略や計画に反映

ここでどれだけテスト観点を発想できるかがテスト設計の鍵
しかし，テスト観点をただ洗い出すだけでは不十分
テスト観点の剪定を行い階層や関連関係を整理する



マインドマップ適用時の基本的な作業手順と範囲

テストベース
（仕様書等）

テスト分析

テスト設計に
マインドマップを適用する

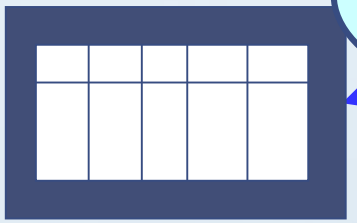
テスト設計
（観点モデルの作成）

Mind Map

直接転記
しない

テスト設計
（テスト技法の適用）

テスト実装



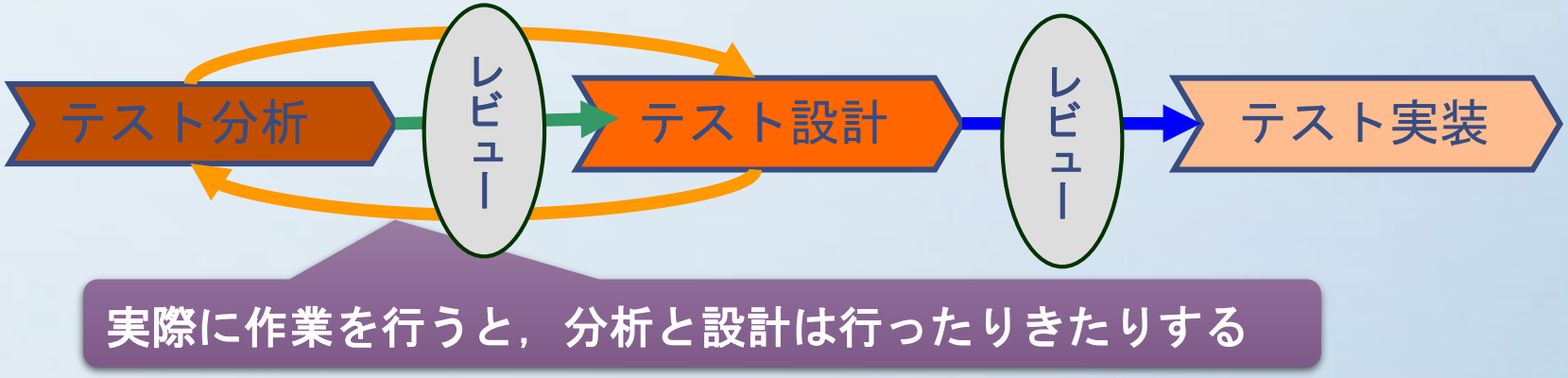
テストケース

分析と設計の成果物として
マインドマップが作成される

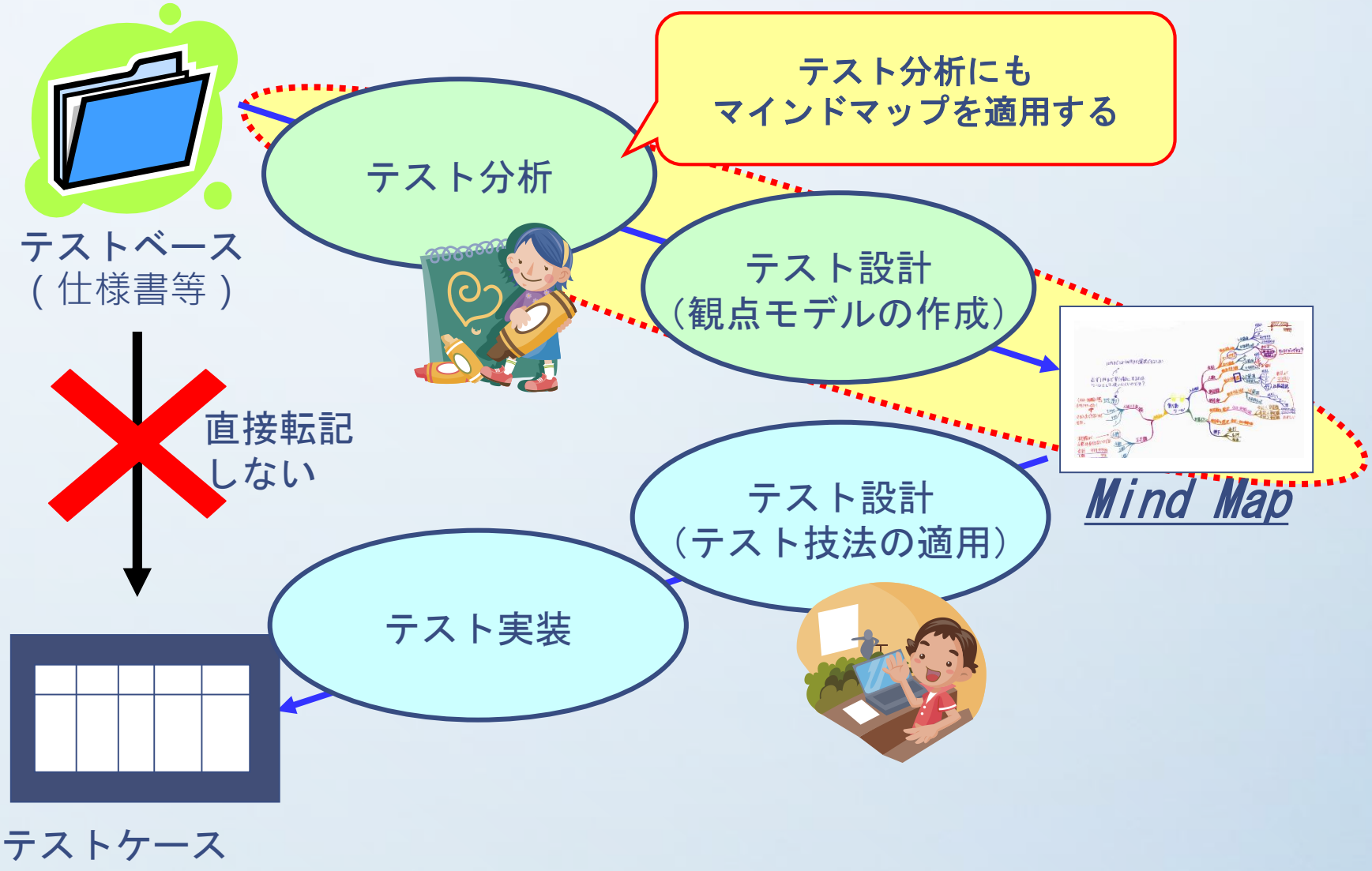
マインドマップではなく，
各種テスト技法を活用する

分析と設計は一緒に考えたほうが良い

- 意識的にテスト分析とテスト設計は分けにくい
  - 作業時の思考は，それほど綺麗に分けられない
    - テスト設計をしていたら，いつの間にか分析を行っていた
    - 分析していたのに，いつの間にかテスト設計視点で考えていた
    - 設計していたら分析が甘いことに気がついた
  - そのため工程を厳密に区切ると，工程終了レビューの回数が増え，工数が増えていく（つまり手戻りの増加）
  - テスト分析とテスト設計とで作業上の責務は分けるが，両方同時に考えたほうがボトルネックが小さくなる



マインドマップ適用時の基本的な作業手順と範囲

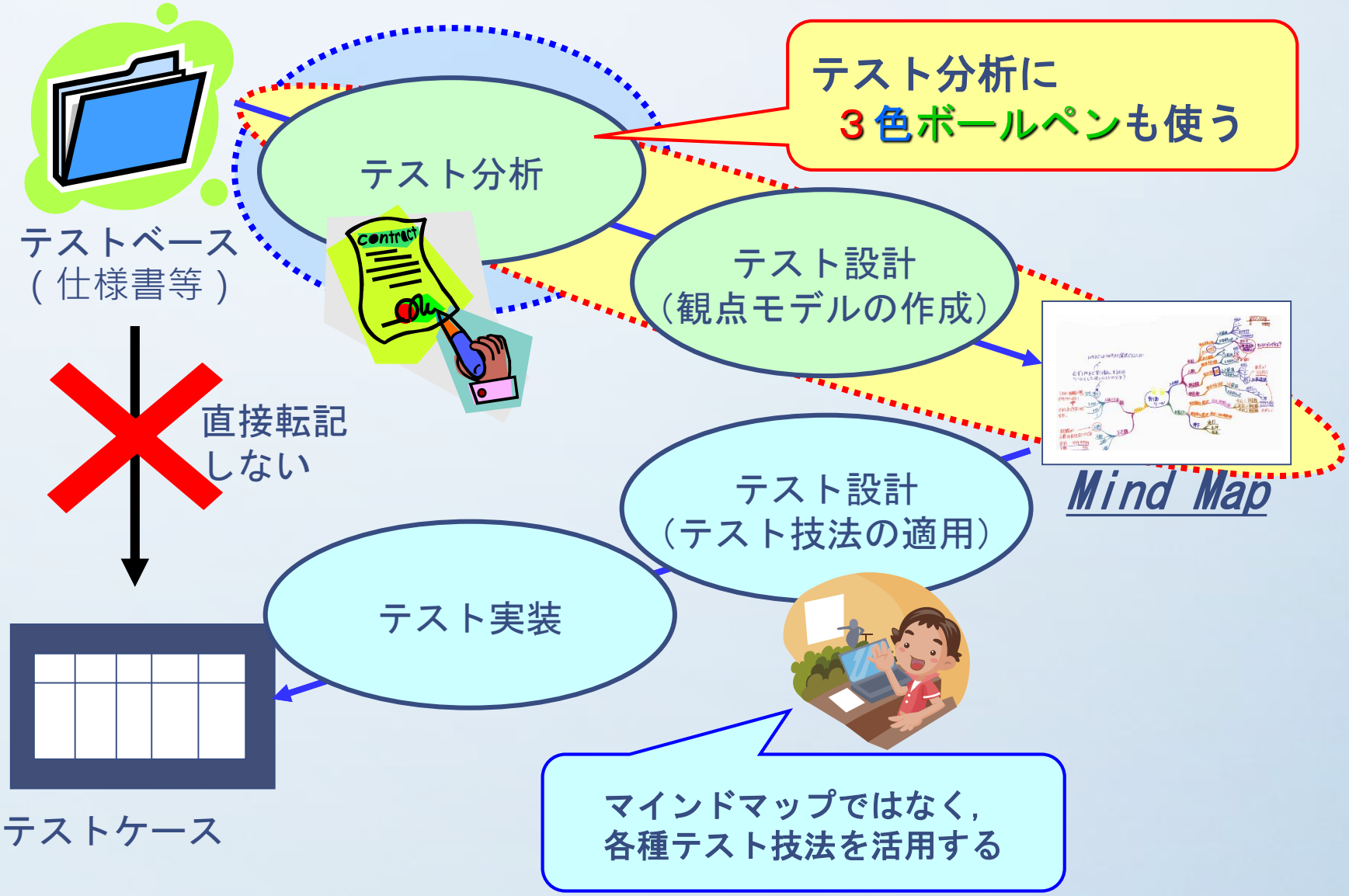


（コツ）テスト分析には３色ボールペンも活用しよう！

- テスト分析をマインドマップだけで行うのはかなり大変
  - セントラルイメージやメインブランチがなかなか決まらない
- まず仕様書を３色ボールペンでチェックし，マインドマップを描く手がかりとする
  - 赤：客観的に「重要」な箇所
  - 青：客観的に「まあまあ重要」な箇所
  - 緑：主観的に「気になる」箇所
- チェックしている時点で，仕様の洩れや抜け，間違いに気がつくことも多い
  - マインドマップを描く前に，テストベースの品質向上できる
  - テスト分析するに値する品質となっているかのチェックとしてもいいだろう

- テストの思考を意識したルールでも良い
  - 赤：「こう動くべき」な箇所
  - 青：「こう動かないべき」な箇所
  - 緑：「それ以外」な箇所

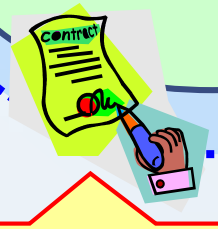
マインドマップ適用時の基本的な作業手順と範囲



マインドマップ適用時の基本的な作業手順と範囲

テストベース
（仕様書等）

テスト分析



テスト分析に
３色ボールペンを使う

テスト分析と設計に
マインドマップを適用する

テスト設計
（観点モデルの作成）

Mind Map

直接転記
しない

テスト設計
（テスト技法の適用）

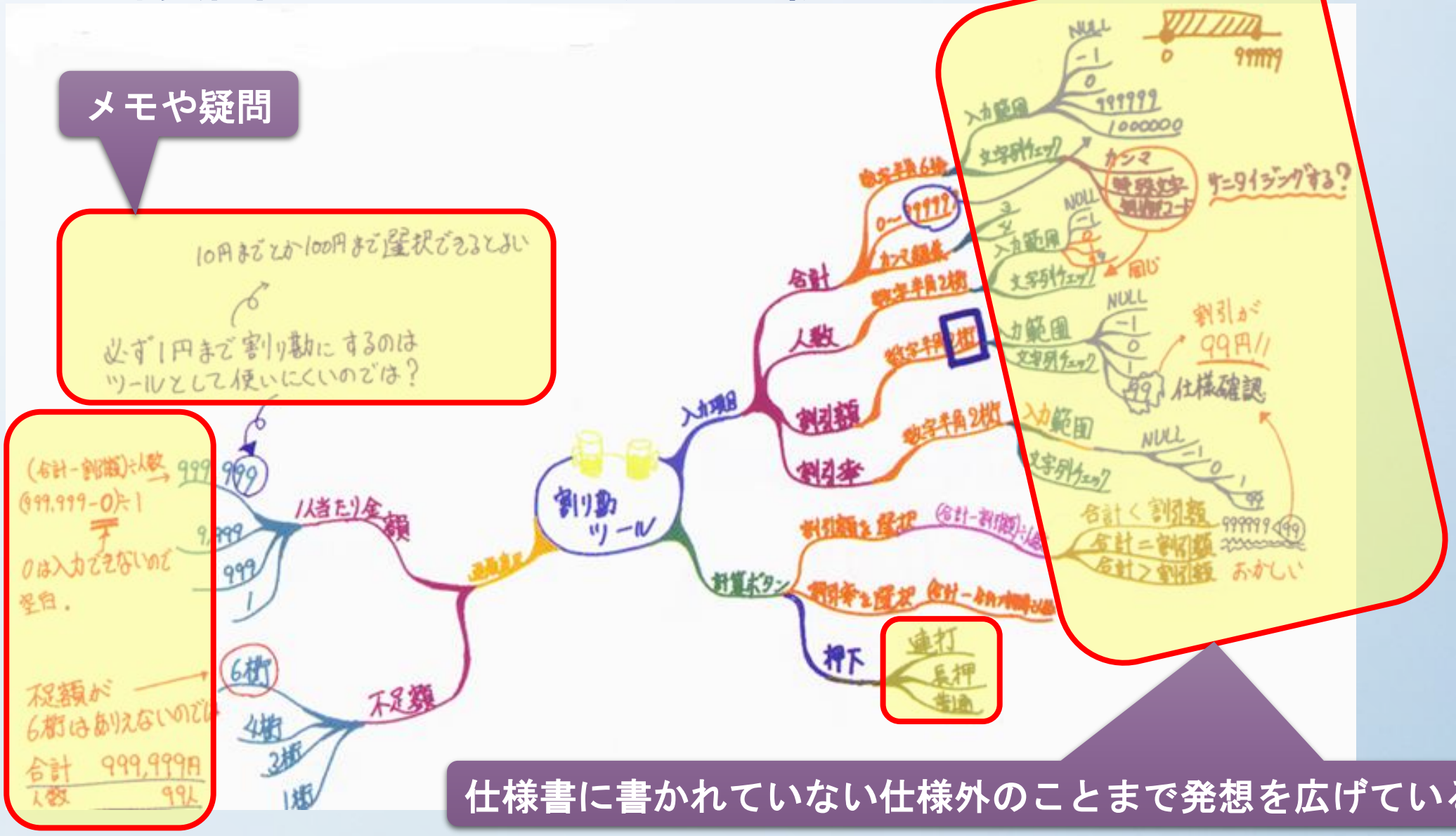
テスト実装

マインドマップではなく，
各種テスト技法を活用する

分析と設計の成果物として
マインドマップが作成される

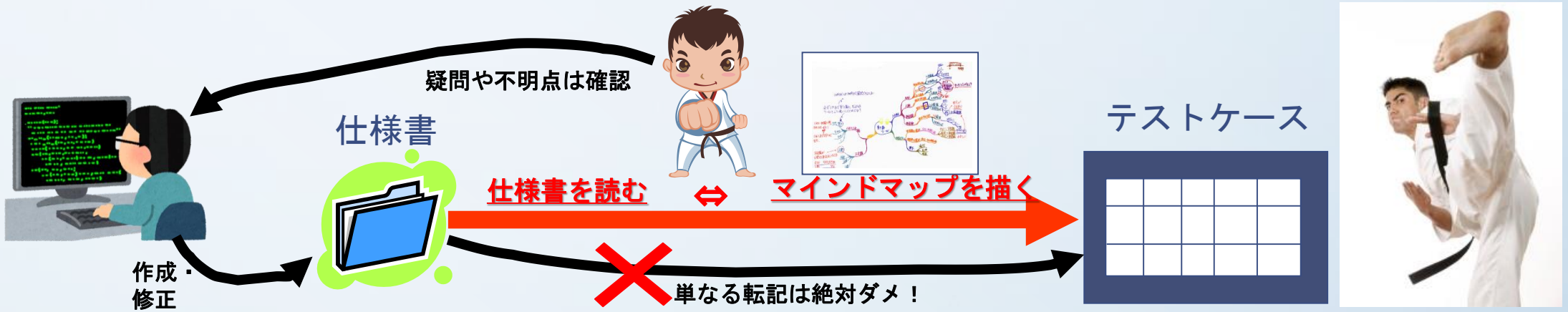
テストケース

テスト設計のマインドマップの例



仕様書に書かれていない仕様外のことまで発想を広げている

マインドマップによるテスト設計で，単なる仕様チェック止まりから卒業！！！



(仕様例)
・ボタンを押すと音が出る

(テストケース)
・
・
・
・
・

考えてみよう！

単なる仕様チェックから卒業するために実践しよう！
- テストケースは仕様を単純に転記しない！
- 仕様に書かれていないことを考える！
- ベースとなる仕様を深く理解するために三色ボールペンを使う！
- 仕様外を発想するためにマインドマップを使う！
- 発想したものからテストしなければならないものをまとめる！
- 最終的にテスト（チェック+テスト＝仕様+仕様外）をテストケースとしてまとめる！

【演習】描いてみようマインドマップ

【身近にある仕様書を元にしてマインドマップでテスト分析設計】

伎：細かいわざ

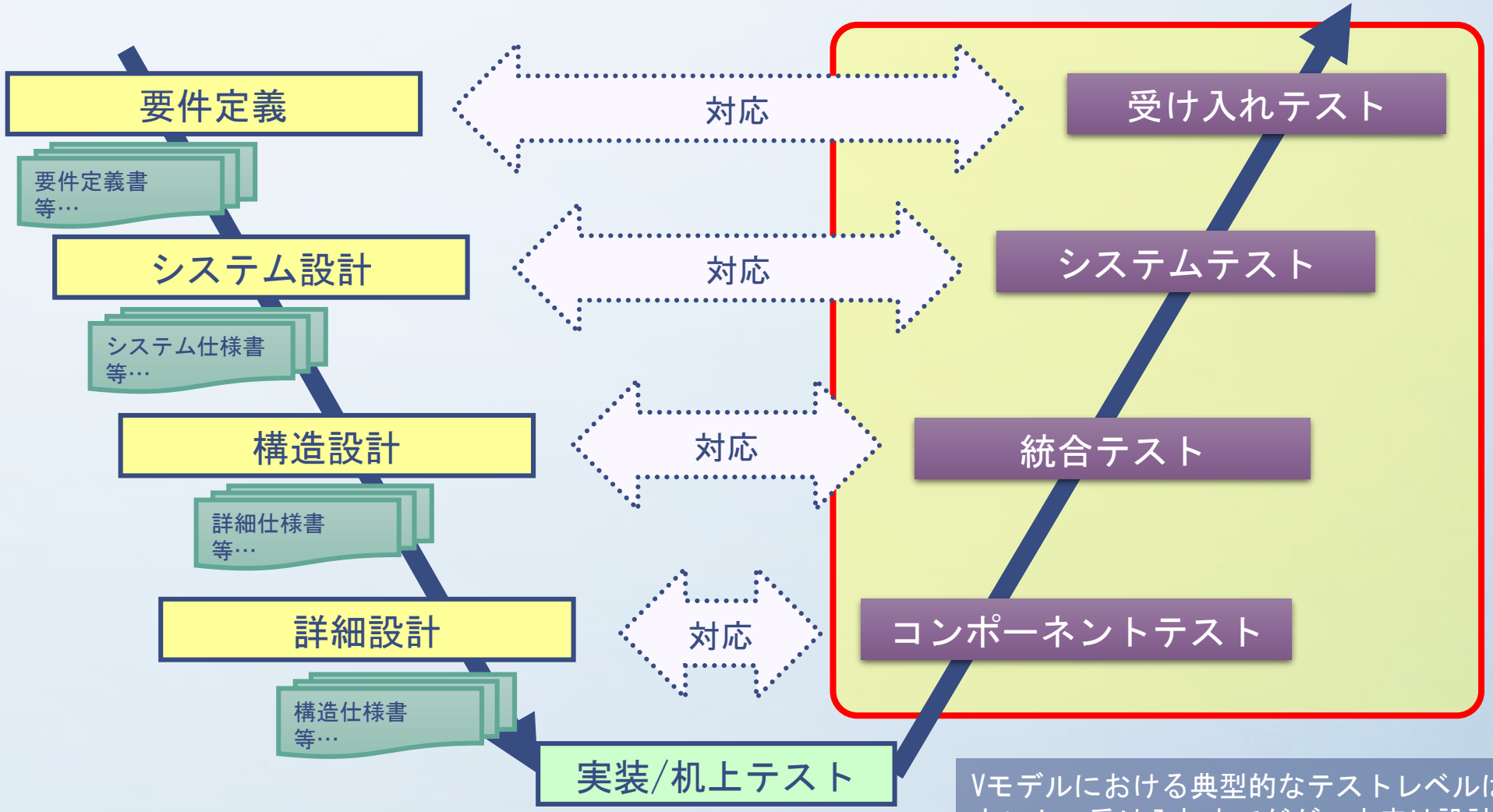
テストの全体像をイメージするための5つの軸

テスト技術を5つの軸で捉える

テスト技術の世界をどう捉えるかは人それぞれだが，本講演ではわかりやすさを重視して5つの軸を設定した

テスト技術を5つの軸で捉える

開発プロセスにおけるテスト（V字モデルでの例）



Vモデルにおける典型的なテストレベルはコンポーネント～受け入れまでだが，本来は設計工程に対応する形でテストレベルを設定する必要がある

テストレベルごとのライフサイクルプロセス

特に準備なく場当たり的にテストを実行（モンキーテスト）

テスト実行

テストケースの作成と実行に概念と作業を分離。テスト技法が普及。

テストケース作成　テスト実行

テストは
実行だけではない！

さらに概念と作業を分割。テストライフサイクルプロセスの考え方が普及。
いかに戦略的なテストを行うかという議論。

分析　設計　実装　実行　報告

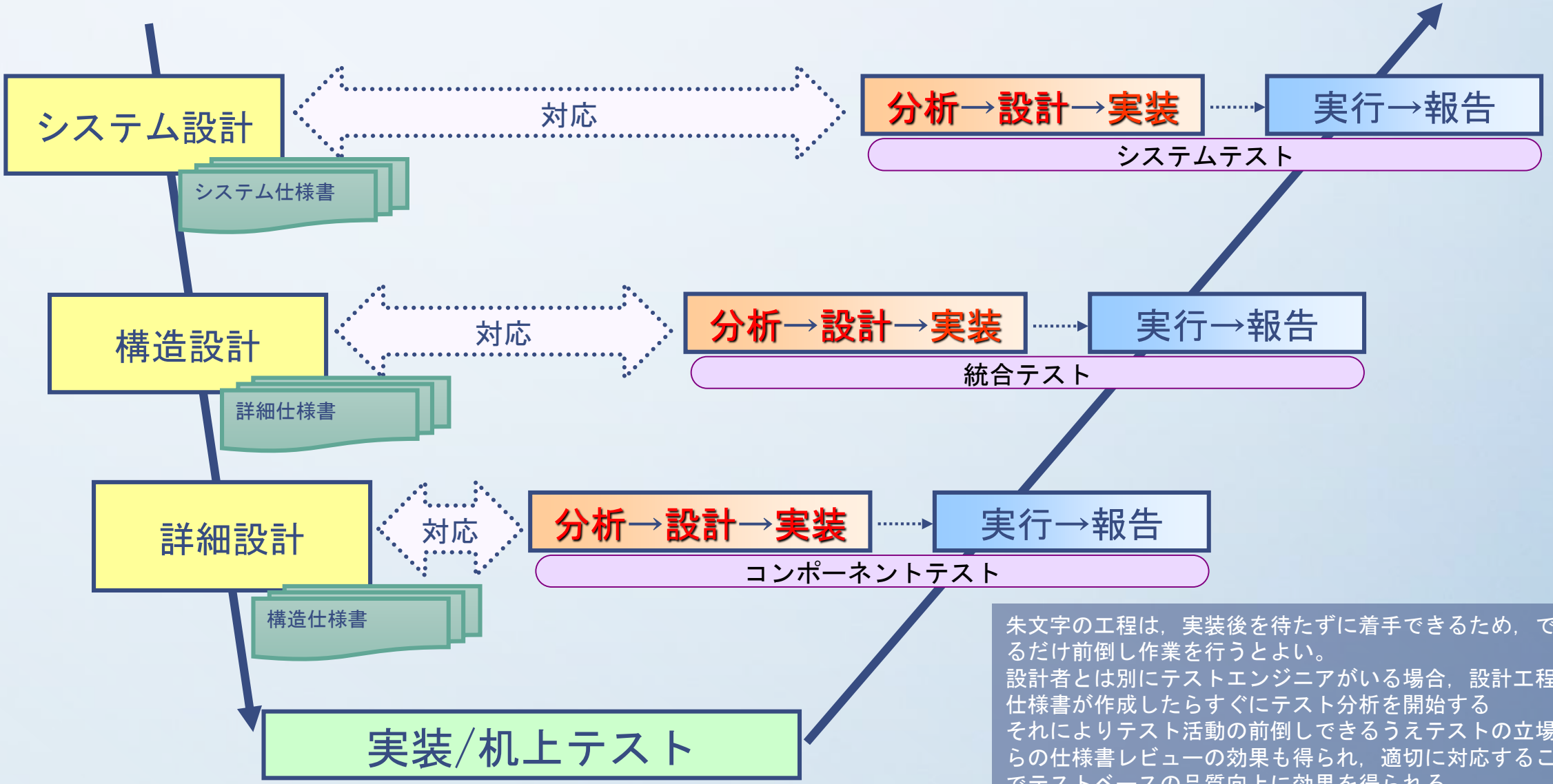
現在は概ねこのようなプロセス

参考：ISTQB-FLシラバスでのテストの活動

テストの活動は，テスト実施の前後にも存在する。例えば，計画，コントロール，テスト条件の選択，テストケースの設計と実行，実行結果のチェック，テスト完了基準の検証，テストプロセスやテスト対象システムに関する報告，テストのまとめや終了作業（テストフェーズが完了した後）がある。テストにはドキュメント（ソースコードを含む）レビューや，静的解析を実施することも含む。

ISTQBテスト技術者資格制度　Foundation Level シラバス 日本語版 Version 2011.J02　から引用

V字モデルへライフサイクルをマッピング



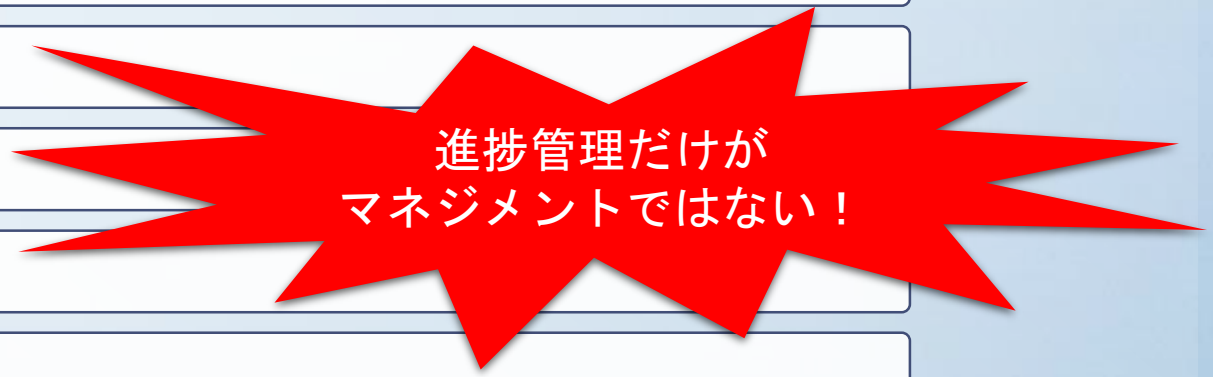
朱文字の工程は，実装後を待たずに着手できるため，できるだけ前倒し作業を行うとよい。
設計者とは別にテストエンジニアがいる場合，設計工程で仕様書が作成したらすぐにテスト分析を開始する
それによりテスト活動の前倒しできるうえテストの立場からの仕様書レビューの効果も得られ，適切に対応することでテストベースの品質向上に効果を得られる。

テスト技術を5つの軸で捉える

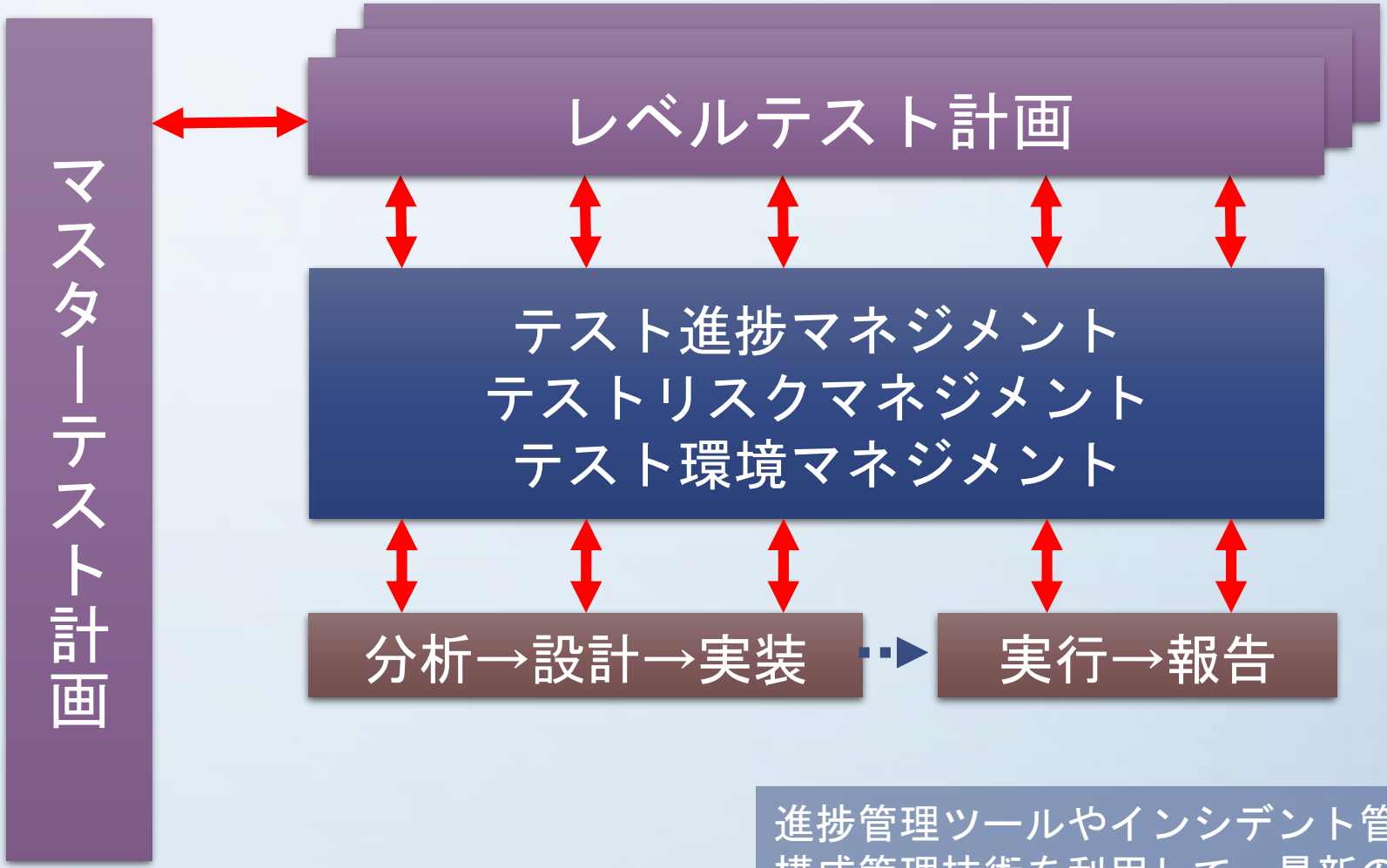
SQuBOKガイド　V2に見るマネジメント

2.18　テストのマネジメント

- 2.18.1.1　テストドキュメントに関する規格（IEEE std 829，ISO/IEC/IEEE 29119-3）
- 2.18.1.2　テストの組織
- 2.18.1.3　テストレベル
- 2.18.1.4　V字モデル（Vモデル）
- 2.18.1.5　W字モデル（Wモデル）
- 2.18.1.6　テスト計画
- 2.18.1.7　テストリスクマネジメント
- 2.18.1.8　テスト進捗マネジメント
- 2.18.1.9　テスト環境マネジメント
- 2.18.1.10　テストに関する規格（ISO/IEC/IEEE 29119 シリーズ）

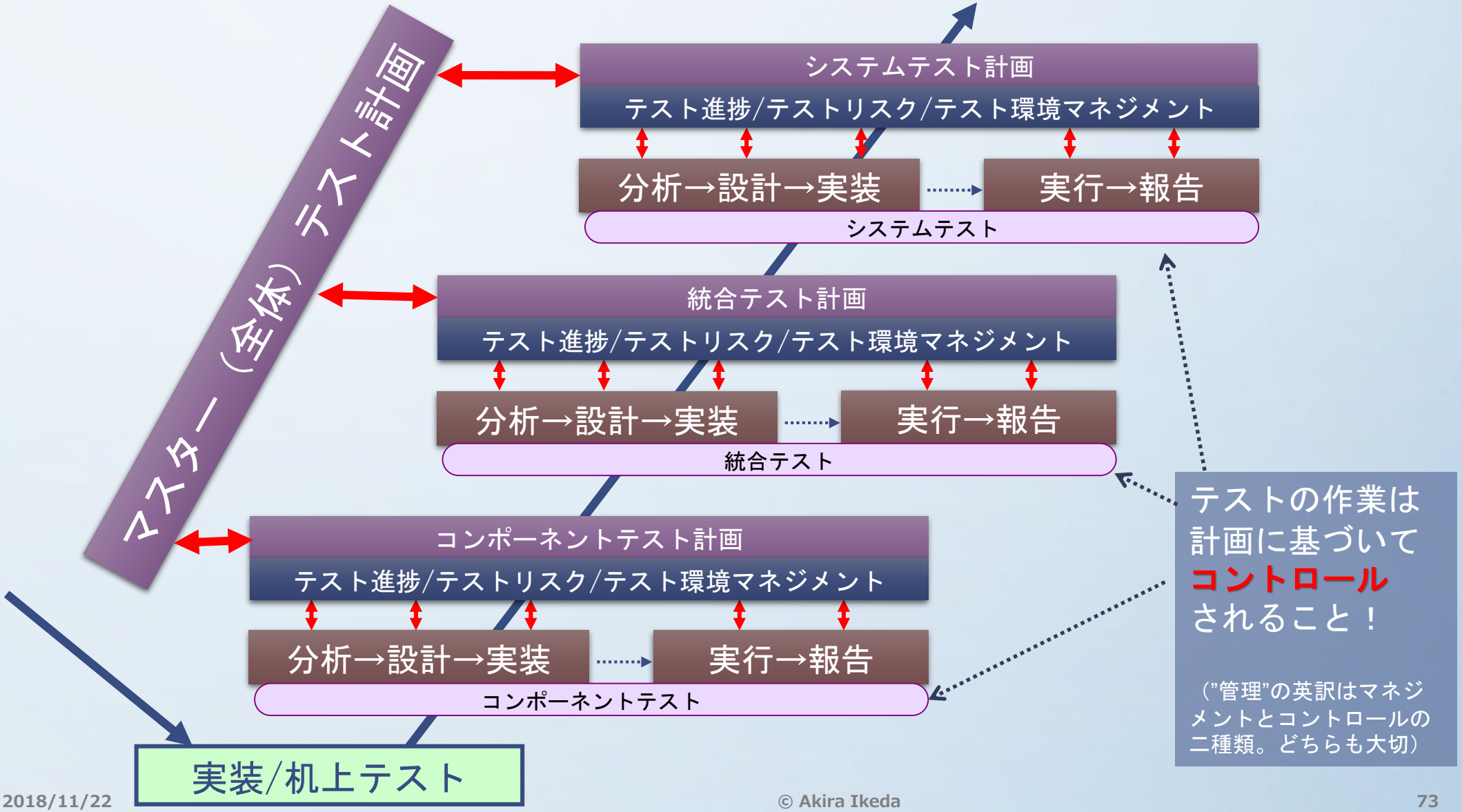


テスト計画とマネジメント，活動の対応イメージ



進捗管理ツールやインシデント管理ツール，構成管理技術を利用して，最新の状況を確実に捕捉，計画にフィードバックする

テスト計画とマネジメント，活動の対応イメージ



テスト技術を5つの軸で捉える

テストの設計技法（テスト観点ベース）

手法名	表現方法	特徴	プロセス
NGT	ツリー	テスト全体をテスト観点で網羅	VSTeP
FV表	表形式	目的機能の切り口でV&Vを網羅	HAYST法
ゆもつよメソッド	表形式	機能×テストタイプで網羅をチェック	ゆも豆
Tiramis	ツリー（MM）	テストカテゴリ分析を実施。機能はMM	-
TAME	ツリー（MM）	テスト設計思考の可視化とレビューによるテスト観点の洗い出しおよび整理	-

SEC BOOKS：高信頼化ソフトウェアのための開発手法ガイドブックから引用
https://www.ipa.go.jp/files/000005144.pdf

マインドマップによる手法以外にもテスト観点ベースのテストの分析設計技法がある
複数の手法を試して自身の職場に適用できるものを活用しよう

そのほかASTER主催テスト設計コンテストの資料も参考になる（http://aster.or.jp/business/contest.html）

テスト技術を5つの軸で捉える

SQuBOKガイド　V2に見るテストの設計技法

- 3.9　テストの技法
  - 3.9.1　経験及び直感に基づいた技法
  - 3.9.2　仕様に基づいた技法
  - 3.9.3　コードに基づいた技法
  - 3.9.4　フォールトに基づいた技法
  - 3.9.5　利用に基づいた技法
  - 3.9.6　ソフトウェアの形態に基づいた技法
  - 3.9.7　組み合わせの技法
  - 3.9.8　リスクに基づいた技法
  - 3.9.19　テスト技法の選択と組み合わせ
  - 3.9.10　テスト自動化技法

テスト技法はホワイトボックスと
ブラックボックスだけではない！

SQuBOKガイド V2に見るテストの設計技法

経験に基づいた技法

- アドホックテスト，探索的テスト

仕様に基づいた技法

- ブラックボックステスト，**同値分割，境界値分析/境界値テスト，**デシジョンテーブルによるテスト，原因結果グラフによるテスト，状態遷移テスト，ランダムテスト，モデルベース土テスト，要因分析技法【富士通】，CFD技法

コードに基づいた技法

- ホワイトボックステスト，制御フローテスト，データフローテスト，データフローテスト，トランザクションフローテスト

フォールトに基づいた技法

- エラー推測テスト，ミューテーションテスト

利用に基づいた技法

- 運用プロファイルによるテスト，ローカライゼーションテスト，ユーザ環境シミュレーションテスト，整合性確認テスト

SQuBOKガイド　V2に見るテストの設計技法

ソフトウェアの形態に基づいた技法

- オブジェクト指向テスト，Webシステムのテスト，GUIテスト，サーバサイドのテスト，データベーステスト，並行プログラムのテスト，プロトコル適格性テスト，実時間のテスト，モバイルアプリケーションのテスト

組み合わせの技法

- 直交配列表（実験計画法）を用いたテスト，All-pair法を用いたテスト

リスクに基づいた技法

- テストマネジメントにおけるリスクベースドテスト，テスト設計におけるリスクベースドテスト

テスト技法の選択と組み合わせ

- 機能的なテスト設計と非確定的なテスト設計の組み合わせ

テスト自動化技法

- -

テストの体系によってはレビュー技術は静的テスト

レビュー方法

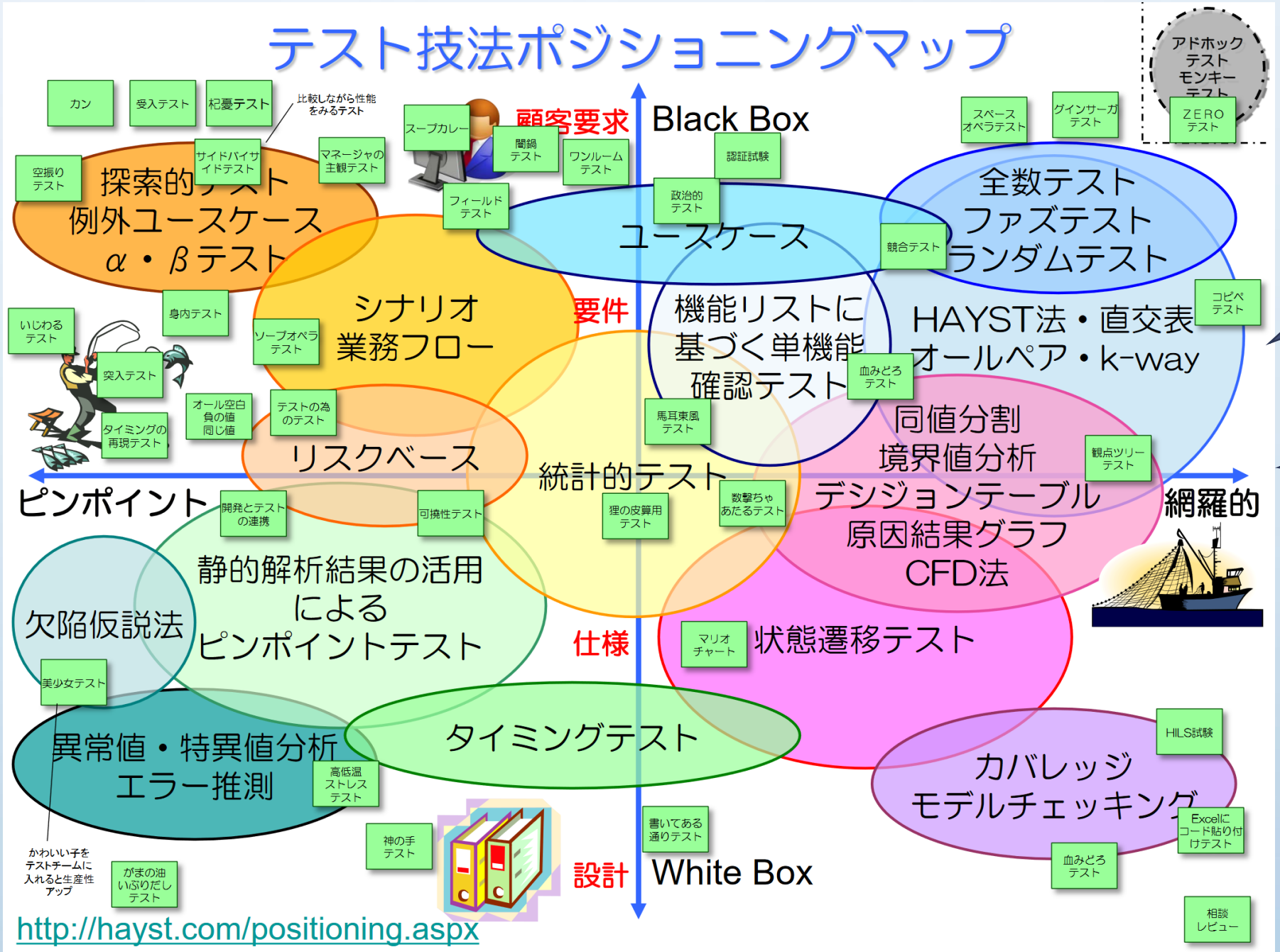
- ピアレビュー、インスペクション、チームレビュー、ペアプログラミング、ピアデスクチェック、パスアラウンド、ラウンドロビンレビュー、ウォークスルー、アドホックレビュー

仕様・コードに基づいた技法

- 形式手法に基づくレビュー、インタフェース分析、複雑度分析、パストレース、ラン・スルー、制御フロー分析、アルゴリズム分析、モジュール展開、七つの設計原理（富士通）、静的解析

フォールトに基づいた技法

- ソフトウェアFMEA、FMECA、FTA（フォールトの木解析）、EMEA（エラーモード故障解析）、CFIA（IBM）、PQデザインレビュー（日立）

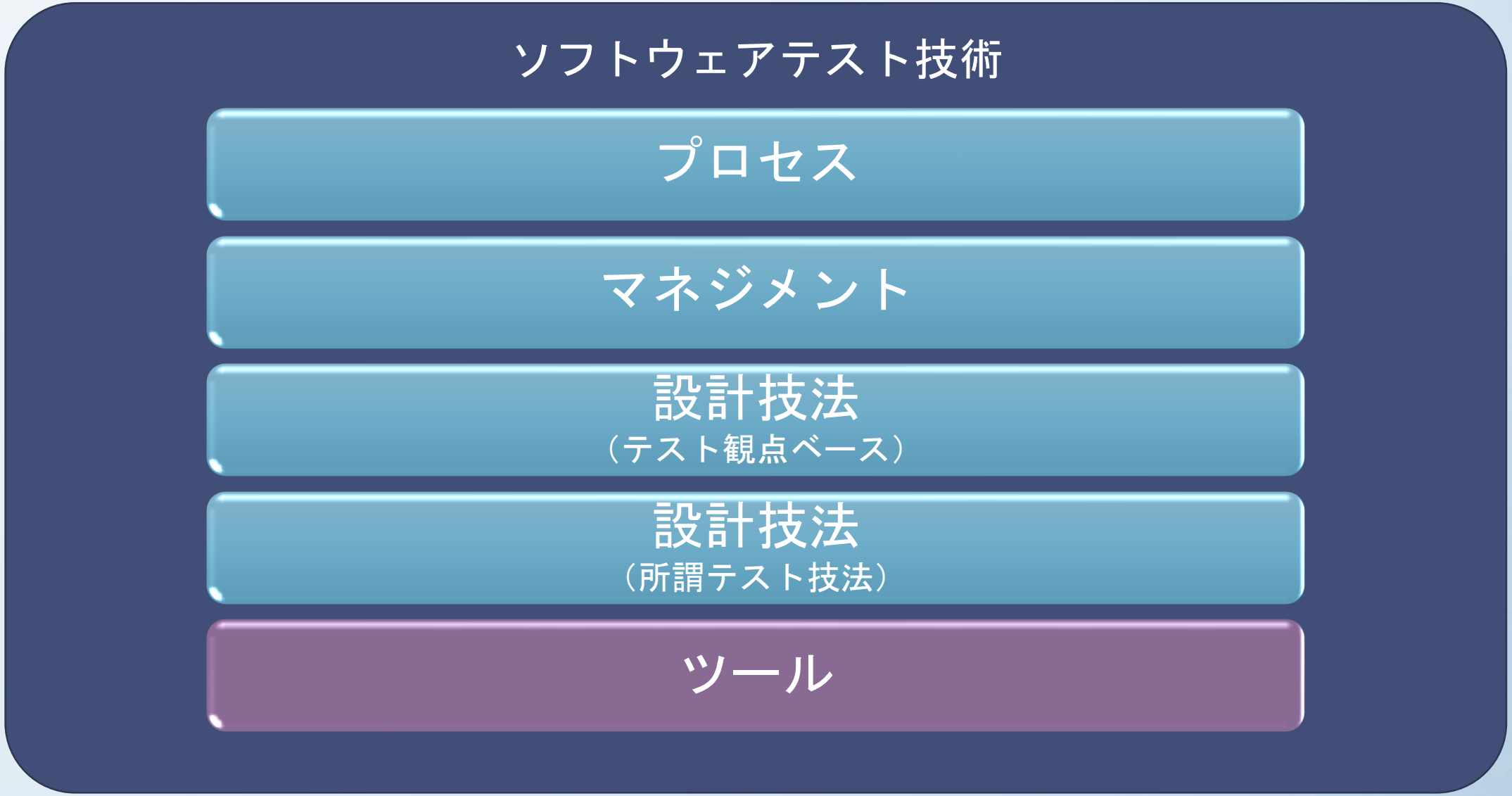


技法にはカバーできる範囲がある。

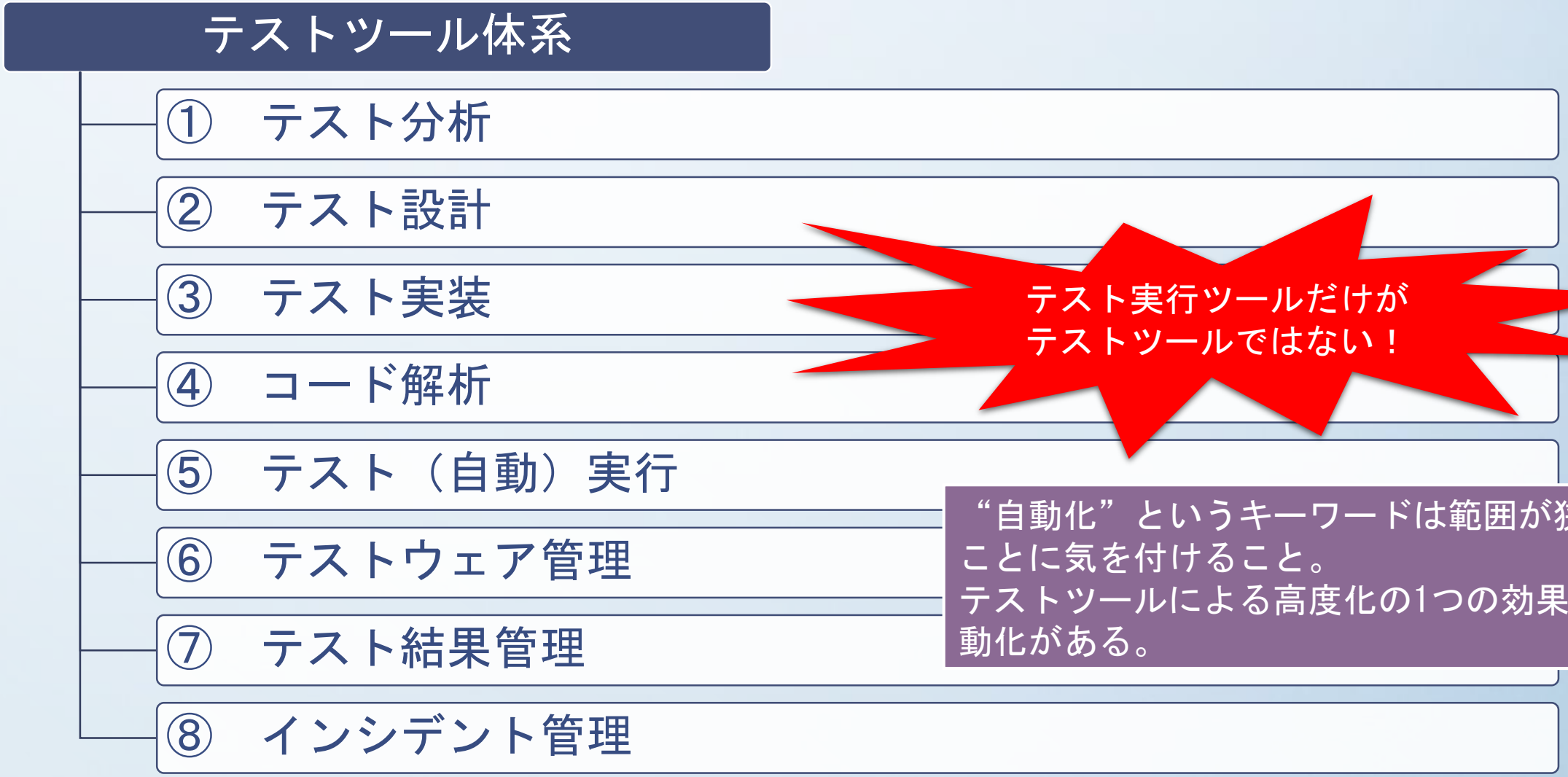
組み合わせて、網羅性を上げる

JaSST'12 Tokyo での「みんなで作ろうテスト技法ポジショニングマップ」
http://jasst.jp/symposium/jasst12tokyo/pdf/S10.pdf

テスト技術を5つの軸で捉える



テストツールまるわかりガイドに見るテストツール



テストツールまるわかりガイドに見るテストツール

テスト分析

- 要件管理ツール

テスト設計

- 状態遷移テストツール，組み合わせテスト支援ツール（直交表，オールペア），原因結果グラフツール，動的解析ツール，カバレッジ計測ツール，その他のテスト設計支援ツール

テスト実装

- スタブツール，シミュレータ，コラボイメージツール，テストジェネレータ，テストケース管理ツール

コード解析

- 静的解析ツール，構造解析ツール

テストツールまるわかりガイドに見るテストツール

テスト（自動）実行

- ユニットテストツール，キャプチャ/リプレイツール，性能テストツール，セキュリティテストツール，テスト自動実行支援ツール

テストウェア管理

- 構成管理ツール

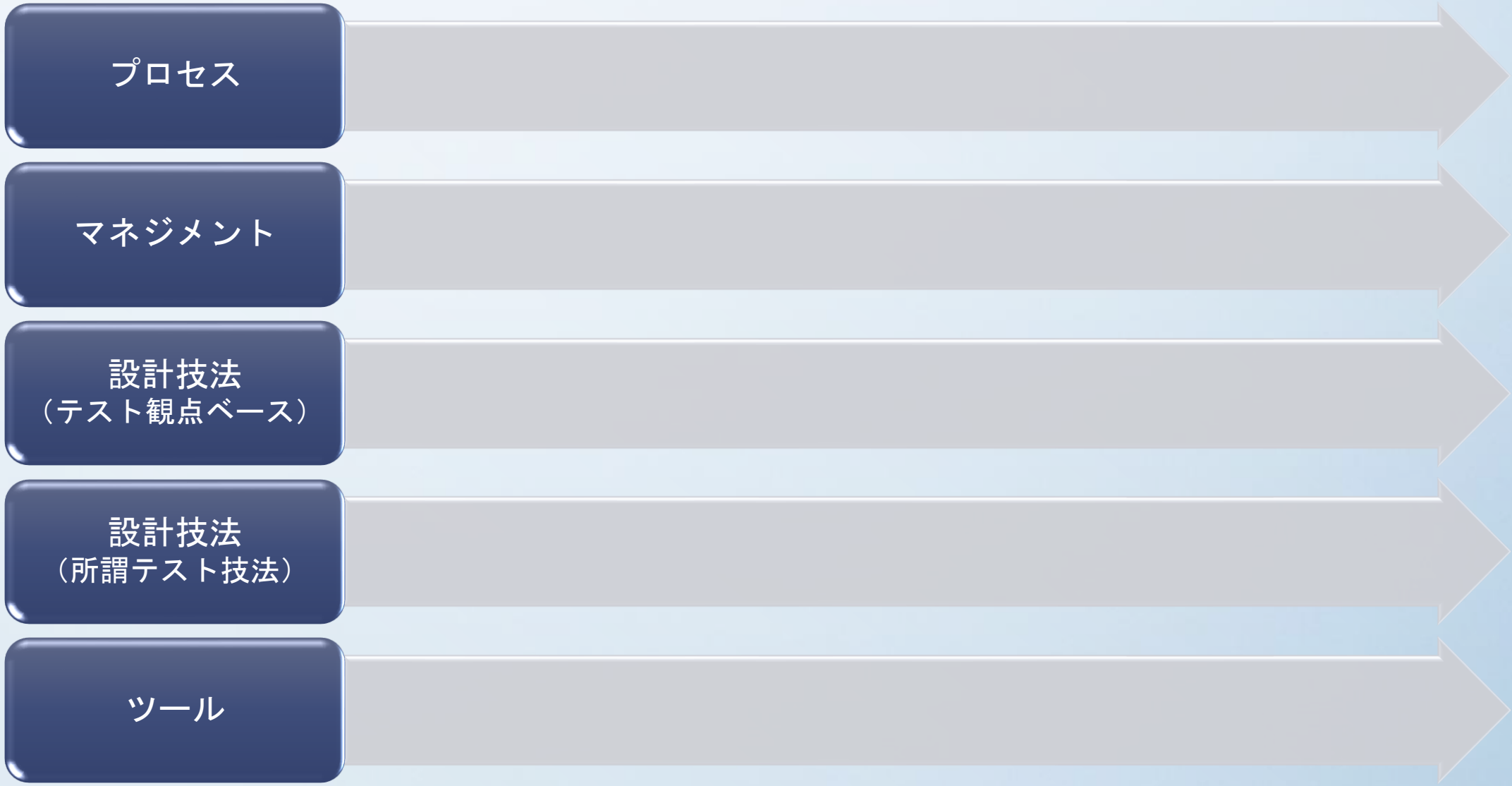
テスト結果管理

- テスト結果管理/テスト結果レポートツール

インシデント管理

- インシデント管理ツール

【演習】5つの軸のそれぞれ何から手をつけよう？



企：先々のことをもくろむ。計画する。

5.【企】始めてみよう テストのスキルアップ

テスト初級者からよくいただく質問

- これからソフトウェアテストの勉強を始めたいのですが，

おすすめの本は？

勉強会とかありますか？

良いセミナーなどありますか？

- これまで皆さんは学校や新人研修では，テストに特化しかつじっくり時間が取られた講義は受けていない場合が多いようです
  - すなわち皆さんはまだほとんどテストについて知りません
  - ですから，まずはじっくり基礎固めすることが必要です

まずは書籍を読もう

- まずはソフトウェアテストという作業や技術のイメージや全体像をつかむことが大切
- 何を始めるにも最低限の知識が必要
- まずは次の4冊を読むことをおすすめ
  （わからないところあっても読みきることが大切）

書籍名	著者
■ソフトウェアテストという作業のイメージを掴むために	
マインドマップから始めるソフトウェアテスト	池田暁，鈴木三紀夫
■テスト技術について広くトピックに触れるために	
知識ゼロから学ぶソフトウェアテスト【改訂版】	高橋寿一
ソフトウェアテスト入門　押さえておきたい　<<要点・重点>>	ソフトウェア・テストPRESS編集部
■テスト技法を学ぶために	
ソフトウェアテスト技法ドリル	秋山浩一

次にWebの記事を読もう

- Webにはたくさんの記事があるがテーマ個別の記事が多く，全体像や外観をつかめるものは少ない
- そういったわけで，書籍で全体像をつかんだあとに読み，知識を強化したり，違った見方を得る
- まずは次の3つの記事から始めるのをおすすめ

タイトル	URL
新人注目！　テストを極める最初の一歩	http://gihyo.jp/dev/serial/01/test_newface
ソフトウェアテスト　基本テクニック	http://gihyo.jp/dev/serial/01/tech_station
テストリーダーへの足がかり，最初の一歩	http://gihyo.jp/dev/serial/01/vital_point

現場で実践し復習しよう

- 知識がたまったら，実際の業務で実践
  - 実践することで理解が深まり，活用するためのコツや悩みを得ることができる
- 実践で得たコツや悩みを意識しながら再度本と記事を読む
  - 自分の理解の誤りやそれまで気がつかなかった重要なことを知識とする
  - スラスラと読めるようになっていたら，確実にスキルアップ（レベルアップ）

書籍名	著者
マインドマップから始めるソフトウェアテスト	池田暁，鈴木三紀夫
知識ゼロから学ぶソフトウェアテスト　[改訂版]	高橋寿一
ソフトウェアテスト入門　押さえておきたい　<<要点・重点>>	ソフトウェア・テストPRESS編集部
ソフトウェアテスト技法ドリル	秋山浩一

タイトル	URL
新人注目！　テストを極める最初の一歩	http://gihyo.jp/dev/serial/01/test_newface
ソフトウェアテスト　基本テクニック	http://gihyo.jp/dev/serial/01/tech_station
テストリーダーへの足がかり，最初の一歩	http://gihyo.jp/dev/serial/01/vital_point

コミュニティに参加しよう

- 本を読み，実践することで知識は増え理解は深まってゆくが，同時にわからないことや悩みも増える
- そこで，同じくソフトウェアテストを生業としているor興味がある人達と会話することで，解決するヒントや新しいアイデアを得る
- 参加費無料，MLベースの活動，ときどき勉強会
  - 勉強会は終了後の懇親会まで参加すると良い

コミュニティ名	URL
TEF（ソフトウェアテスト技術者交流会）	http://www.swtwst.jp
SQiPコミュニティ	https://www.juse.or.jp/sqip/community/sqip/

シンポジウム・ワークショップに参加しよう

- シンポジウムでは他社の様々な取り組みが発表され，テストに関連するたくさんのツールに触れる機会が得られる
- 幅広いテーマの先端的な知見や実践事例が得られることで，それまでに得た知識の応用のヒントが得られ，また自分の仕事をさらに改善することができる
- 様々な論文発表や企画セッション
  - 直接講師に質問することも可能なため，情報交換会は積極的に参加するとよい

コミュニティ名	URL
JaSST（ソフトウェアテストシンポジウム）	http://www.jasst.jp/
WACATE（ソフトウェアテストワークショップ）	http://wacate.jp/
SQiPシンポジウム（ソフトウェア品質シンポジウム）	https://www.juse.jp/sqip/symposium/

これまでの知識を整理しよう

- これまでに得られた知識をレポートとして整理する
- 頭の中だけだと，どんどん記憶＝知識は失われていく
- 自分のために資料化することで，知識を整理することができ，また失われない情報となる
- さらに資料化する過程で，さらなる疑問や発想が得られる

知識の整理例
読んだ本や記事のまとめ（感想文）
実践して得られたノウハウや悩みリスト
コミュニティで参考になったメールのやりとりを整理したもの
勉強会やシンポジウムの参加レポート

現場に展開しよう

- 知識やノウハウは自分のものだけにせず，同じ現場のメンバに広める
- 現場を意識して資料を作る過程で，知識と理解がさらに深掘りされる
- 資料は個人・組織としてノウハウが凝縮された貴重な資料となる
  - 組織改善に貢献するアウトプットを作成する行為とも言えます
  - 是非それは自分の成果であると上長にアピールしましょう
  - 社外活動への理解を得るきっかけともなる

現場への展開例
レポートをメールで配信
社内SNSでの記事作成
社内勉強会の実施
活動施策として現場提案

勉強会やシンポジム・カンファレンスで発表しよう！

- 社外勉強会で発表することで，さらに知識の高度化できる
- 社外発表の経験と実績を得ることができる
- 発表資料は技術者としての自分の名刺として使える
- 聴講者からたくさんかつ多様なフィードバックを得られる
- 何より，スキルアップしたことの充実感や達成感を得られる

発表で得られる物

- 社外勉強会での発表の経験や発表資料
- 発表までに整理したり調べたりした知識
- 聴講者からの沢山のフィードバックや
- 技術者のコミュニティ仲間
- 充実感や達成感，さらなるモチベーション

（参考）成熟度モデル，認定資格，スキル標準

資格試験

個人を測る

- ISTQB

スキル標準

個人を測る　→　集約して組織を測る

- Test.SSF

テストプロセス成熟度モデル

組織を測る

- TMMi
- TPI / TPI NEXT
- CTP
- STEP

技術力アップのために利用できる無償の資料（一部）

テストの基礎的技術を学びたい

- ISTQB-FLシラバス＆用語集（ISTQB） http://jstqb.jp/syllabus.html
- ASTERセミナー標準テキスト（ASTER） http://aster.or.jp/business/seminar_text.html

テスト設計について知りたい

- テスト設計コンテスト関連資料（ASTER） http://aster.or.jp/business/contest.html

テストツールについて知りたい

- テストツールまるわかりガイド（ASTER） http://aster.or.jp/business/testtool_wg.html

バグ管理について知りたい

- はじめてのバグ票システム導入実践ガイド（NaITE） http://naite.swquality.jp/?page_id=40

テストスキル標準について知りたい

- Test.SFF（IVEC・ASTER） http://aster.or.jp/business/testssf.html

テストに関する最新動向や事例情報を知りたい

- JaSST（ASTER） http://www.jasst.jp/
- SQiPシンポジウム（日科技連SQiP） https://www.juse.jp/sqip/symposium/

その他

- テストエンジニアステーション（技術評論社） https://gihyo.jp/ad/2008/test
- 山浦恒夫の“くみこみ”な話（ITMedia） http://monoist.atmarkit.co.jp/mn/kw/yamaura_kumikomi.html

技術力アップのために参加するべき国内イベント

シンポジウム・カンファレンス・ワークショップイベント

- JaSST　http://www.jasst.jp/
- SQiPシンポジウム　https://www.juse.jp/sqip/symposium/
- テスト自動化カンファレンス　https://sites.google.com/site/testautomationresearch/
- WACATE　http://wacate.jp/　←チラシを配布していただきますので参考になさってください

コミュニティ（全国）

- TEF（テスト技術社交流会）　http://www.swtest.jp/index.php?swtest.jp/wiki/forum
- SQiPコミュニティ　https://juse.or.jp/sqip/community/sqip/

コミュニティ（九州内中心）

- NaITE（長崎ＩＴ技術者会）　http://naite.swquality.jp/
- 九州ソフトウェアテスト勉強会　https://www.facebook.com/groups/Testing.Qshu/
- テスト酒場in福岡　https://test-sakaba.connpass.com/
- NetFocs　https://netfocs.connpass.com/
- 長崎 WordPress ユーザーの集い　https://www.facebook.com/groups/656863351010318/
- ぺちぱな　https://atnd.org/groups/phper-na

【演習】企画してみようスキルアップ

5年後に身につけておきたいスキル	3年後達成しておきたいこと	これから1年の間に取り組むこと

5. おわりに

よりよいテストのために必要な技術・知識

ドメインの専門技術・知識

テスト技術の専門技術・知識

開発の専門技術・知識

このあたりの話は、井芹さんや
実行委員の皆さんのセッションで得られます！

支える・繋がる

テストの基礎

本講演ではこのあたりの話をしました

本日はテストの“キホンのキ”がテーマでした

- テストには日々取り組んでいるけれど、改めてどういった活動でどういった技術が必要なんだろう？
- 本セッションはテストに取り組み始めた方や**テスト初級者向け**に行います。テストといいつつ単なる仕様チェックになりがちな状態を卒業したい、テスト技術の全体像を掴みたい、テストの技術を高めるために勉強を始めたりスキルアップに取り組みたいと考えている方向けに、抑えておきたいキホンのキをお話しします。
  - **テストを行なう意義**を振り返り、**テストに必要な思考**をおさらいします
  - 単なる仕様チェックから抜け出すための手段の１つとして、**マインドマップを使ったテスト分析・設計の基本**を紹介します
  - **テスト技術の全体像**をイメージするために、構成要素を5つの軸から捉えます
  - そのうえでそれらの**勉強やスキルアップを始めるためのモデルケースやヒント**をお伝えします。
- テスト技術力を高めていくためには**テストそのものについて自分なりのイメージや地図を持つ**ことが重要です。是非この機会にイメージを掴み、具体的な行動を起こすヒントを得ましょう！

みんなで作ろう“キホンのキ“

テストケース設計のキホンのキの例

- 軌：仕様どおり（有則，禁則）
- 柵：仕様に存在する際（境界）
- 奇（危）：仕様から外れたところ（無則）

○○○のキホンのキ

- _____
- _____
- _____

それぞれのキホンのキを考えてみよう！

Thank you! Nagasaki!

本講演が皆さんにとりまして

テスト技術を高める第一歩となる情報を

お伝えできていれば幸いです！

是非，一緒に

ソフトウェア開発を
よりよくしていきましょう！

NaITE （長崎IT技術者会）

長崎に興味がある、住んでいた、友達がいる等、長崎に何かしらの縁がある人で活動中のIT技術コミュニティです。地元長崎は勿論、全国各地で活動中、ITエンジニアが主役として活躍できる、長崎県人会的に柔らかに活動中です！

長崎での技術イベントや全国各地での勉強会、SIG活動等に積極的に取り組んでいます！

■技術イベント
年に一度ソフトウェア品質と開発を扱うイベント長崎QDGを開催、その他Agile Japan 長崎サテライト等を開催しています。

■勉強会
定期的に開催する勉強会を通して技術情報を発信したり、議論を促進することで、技術者の技術向上に取り組んでいます。

■SIG
特に興味のあることついて継続的に議論したり、技術ノウハウをまとめたりする研究活動に取り組んでいます。

#	内容	開催地
第26回	『ゆるファシ』体験ワークショップ	神奈川
–	3rd 長崎 Software Quality and Development Gathering	長崎
–	NaITE-Night vol.1	長崎
第27回	ケースメソッド体験ワークショップ	神奈川
第28回	プロダクトオーナーシップ & Agile Japan 2018 参加報告	神奈川
–	Agile Japan 2018 長崎サテライト with NaITE	長崎
–	NaITE-Night vol.2	長崎

ガイドを無償公開中！ ⇒

◎バグ票システム検討ＳＩＧ
「**はじめてのバグ票システム　～導入実践ガイド　1.1版**」
http://naite.swquality.jp/?page_id=40

■その他の活動
ソフトウェア技術の話題を肴に議論を楽しむNaITE-Nightの開催や、他団体・他コミュニティ主催イベントへの出講出展を通して、幅広い技術交流を図っています。

公式サイト

- NaITE（長崎IT技術者会）公式サイト
  http://naite.swquality.jp
- Facebookページ「NaITE（長崎ＩＴ技術者会）」
  https://www.facebook.com/NagasakiITEngineers/
- Twitter公式アカウント：@NagasakiITEng
- お問い合わせ：http://naite.swquality.jp/?page_id=544

２０１８年も技術イベントを開催し、長崎を盛り上げています！

2015年から毎年NaITEのフラグシップイベントを開催！
今年は“レビュー”技術を軸に幅広いセッションを設けました！
当日は長崎内外から多くの技術が集まり、熱気あふれる議論や積極的な交流をはかりました！

名称 ： 3rd 長崎 Software Quality and Development Gathering
日時 ： 2018/2/2(金) 12:00～17:20　場所 ： メルカつきまち
主催 ： NaITE（長崎IT技術者会）
協賛 ： AFFORDD（派生開発推進協議会）
プログラム：
スペシャルセッション
『ソフトウェア品質レビューの全て：生い立ちから先進技術まで』：細川 宣啓 氏（日本アイビーエム）
セッション：
『単なる仕様チェックから卒業するために』池田 暁 氏（NaITE）
『高速２パーソンインスペクションのススメ』：原 佑貴子 氏（日本アイ[illegible]）
『いろいろ使えるセーフウェアのためのモデルSTAMPとその分析法』：日下部 茂 氏（長崎県立大学）
『PSP活用SIG活動報告』：くまきち 氏
『チームで、長期間で、たくさんのソフトウェアを快適に開発し、価値を生み続けるためのエンジニアリング』：金子 昌永 氏（日立製作所）
他、参加者によるＬＴ（ライトニングトークス）

共同実行委員長
池田 暁（NaITE）
日下部 茂（長崎県立大学）
実行委員
大月 美佳（佐賀大学）
小笠原 秀人（東芝）
小畑 慶次（デンソー技研センター）
佐藤 陽春
すずき しょうご（NaITE）
ツノダ シュン（NaITE）
手島 尚人（GMOペパボ）

2016年・2017年に引き続き、今年も開催！
Agile japan 2018 の基調講演はもちろんアジャイル実践者によるセッションなど充実した内容をお届けしました！

名称 ： Agile Japan 2018 長崎サテライト with NaITE
日時 ： 2018/9/15(土) 12:00～16:40　場所 ： 長崎市立図書館
主催 ： NaITE（長崎IT技術者会）
プログラム：
Agile Japan 2018 基調講演：
『モブプログラミングと“フロー”の力』：Woody Zuil 氏
『Japan Taxi の挑戦』：川鍋 一朗 氏（JapanTaxi）
セッション：
『コミュニケーションから始めるアジャイル』：手島 尚人氏（GMOペパボ）
『レジエンス工学の手法によるアジャイルプロセスのモデリング』：日下部茂 氏（長崎県立大学）
『知識集約型の製品開発においてプロダクトオーナーがやるべき３つのこと』：関 満徳 氏（グロースエクスパートナーズ）
他、参加者によるＬＴ（ライトニングトークス）

共同実行委員長
池田 暁（NaITE）
手島 尚人（GMOペパボ）
実行委員
大月 美佳（佐賀大学）
日下部 茂（長崎県立大学）
ツノダ シュン（NaITE）
会場スタッフ
すずき しょうご（NaITE）

2019年に4th 長崎Software Quality and Development Gathering を開催！
発表者・協賛企業/団体・スポンサー・実行委員を募集中です！