テスト設計技法，その前に

～フェイスアップ，次にビルドアップ，その先にマインドアップ～

池田 暁
NPO法人ASTER　理事
NaITE（長崎ＩＴ技術者会）代表

2019/08/30（金）
於　札幌市教育文化会館

自己紹介

© Akira Ikeda

自己紹介

池田 暁 氏 （ASTER）

情報通信系某社にて組込みシステムの設計、品質保証業務を経て、技術支援部門にてテスト技術を中心にアジャイル開発やMBD/SPL等の導入を支援。その後所属を移し医用系機器開発プロジェクトに参画、テスト管理業務やプロセス活動に従事。
昨年日立オートモーティブシステムズ(株)に所属を移し、全社を対象としたテストプロセス改善や研究による新技術導入、教育や社内イベントでの技術力向上に取り組んでいる他、テスト自動化チームを率いてプロダクトの品質確保や効率向上、技術高度化に取り組んでいる。
そのほか、日立グループ横断部会活動のテスト分野主査としてグループ全社のテスト技術力向上や技術者育成活動に取り組んでいる。
社外においてはNPO法人ASTERや技術コミュニティに所属し、国内でのテスト技術の普及啓発に取り組んでいる。ASTER理事，NaITE代表、SQiP運営委員，AFFORDD運営委員等。

著書：

『[改訂新版]マインドマップから始めるソフトウェアテスト（共著）』、『SQuBOKガイド（共著）』，『ソフトウェアテストの基礎（共訳）』等。

© Akira Ikeda

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やテストチーム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

その昔，2年間北海道・江別に住んでいました

今日は，思い出の土地で講演できてとても嬉しいです！

© Akira Ikeda

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，「はじめてのバグ票システム～導入実践ガイド」など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/

是非ご参加ください！ ＆ 一緒に企画しましょう！

http://naite.swquality.jp/

© Akira Ikeda

マインドマップから始めるソフトウェアテストとは

- 2007年に発行したソフトウェアテストの書籍
  - 2007年6月に技術評論社より
  - ソフトウェア・テストPRESS vol.3～5の記事をベースに大幅加筆
- テスト初級者向けに次を解説
  - テストの作業工程
  - テストの思考過程
- ポイント
  - テスト技法ではなく，テストプロセスを学べる
  - テストでのマインドマップの応用例を学べる
  - ブックガイドにて今後のテストのスキルアップのための情報を知れる

© Akira Ikeda

マインドマップを始めるソフトウェアテストが発行されてから変わったこと

- テスト技術者がテスト分析やテスト設計でマインドマップを使うことが当たり前となった
  - 勉強会に行くと当たり前のように使っている
- テストプロセス「分析」「計画」「設計」「実装」「実行」「報告」が一般的に認知され，
  テストプロセスを整備する現場が増えた
  - 具体例から，具体的な活動として定義できるようになった
- テストは楽しく想像的な技術であることが認知された
  - Excelチョメチョメからの脱却

© Akira Ikeda

[改訂新版]がでました！

- 2019/4/13発売
- 池田 暁，鈴木 三紀夫 著
- 技術評論社，単行本（ソフトカバー）: 224ページ
- 2,678円

基本的な構成は大きく変更しない

コラムを追加

陳腐化している第Ⅲ部を廃止し，旧第Ⅳ部を新第Ⅲ部として加筆修正する

ブックガイドの情報を更新

その他，情報の追加や文章表現，用語の調整

© Akira Ikeda

参考：改訂新版の概観

© Akira Ikeda

北海道との縁を少し・・・

© Akira Ikeda

北海道にて2年間育てていただきました

- 2年間，北海道情報大学大学院に所属していました
- 視覚障害者の方向けのバスの音声時刻表に関する研究に取り組みました
- 学部時代は，遠く，初めての土地で何もわからないなか，北海道の方々に温かく付き合っていただきました
- 江別に一人暮らししていましたが，札幌にも良く買い物等にきていました

© Akira Ikeda

初回である，JaSST’06 Sapporo に
東京からの応援メンバーとして参画してました

JaSST'06 in Sapporo 開催日：2006年10月17日（火）

JaSST'06 in Sapporoは、おかげさまで１００名を越えるご参加を頂き、盛況のうちに開催を終了いたしました。多くのご参加を誠にありがとうございました。

プログラム	ワークショップ
事前アンケート	テスト技術者資格制度
基調講演	事例・論文発表
パネルディスカッション	開催要項
チュートリアル	スポンサー協賛企業

© Akira Ikeda

11年前に札幌でワークショップを行ないました

年度別　チュートリアル／ワークショップ／JaSSTセッション一覧

年度	チュートリアル／ワークショップ／JaSSTセッション		
2006	チュートリアル	知識ゼロから学ぶソフトウェアテスト	高橋 寿一 （ソニー）
	ワークショップ	ソフトウェアテストの現状	JaSST札幌実行委員会
2007	チュートリアル	品質と生産性がアップする組合せテストの実践	秋山 浩一（富士ゼロックス）
	チュートリアル	入門：ユーザビリティの概念とその事例 ～仕様書レビューにおける「視座」の重要さ～	尾形 慎哉（U'eyes Design）
	ワークショップ	テスト設計	松田 英克（JaSST'07 Sapporo 実行委員）
	ワークショップ	ピアレビュー	安達 賢二（JaSST'07 Sapporo 実行委員）
	ワークショップ	変更要求の仕様化とテストでの活用	本多 慶匡（JaSST'07 Sapporo 実行委員）
2008	ワークショップ	テストをもっと創造的に　分析・設計エクササイズ	鈴木 三紀夫（TIS） 池田 暁（日立情報通信エンジニアリング）
	ワークショップ	HAYST法による組合せテスト実践トレーニング	秋山 浩一（富士ゼロックス）
	JaSSTセッション	寸劇から出荷判定の基本を学ぶ	S-open（ソフトウェア技術ネットワーク）メトリクスSIG有志"劇団メトリ"
	JaSSTセッション	実演ユーザビリティテスト	佐々木 誠司（JaSST Hokkaido）

© Akira Ikeda

11年前に札幌でワークショップを行ないました

・テスト勉強道場：みんなでやってみよう！

■テストをもっと創造的に　分析・設計エクササイズ

鈴木 三紀夫（TIS）
池田 暁（日立情報通信エンジニアリング）

講演資料
（PDF: 143K）

© Akira Ikeda

本日は北海道と北海道の技術者に少しでも貢献したい

私を学生として向かい入れ，

マインドマップに関連する技術交流の場を与えていただき，

叱咤激励をいただきながら育てていただいたことに感謝しています

本日は，そのご恩を少しでもお返しできるよう，

拙いながら，頑張ってお話ししたいと思います！

© Akira Ikeda

はじめに

© Akira Ikeda

JaSST’19 Hokkaido のテーマ(抜粋)

今年のテーマは

「マインドアップ ～テスト設計技法を知っていたら十分と思ってないかい？～」

です。

※"マインドアップ"は造語になります。"知力の向上"，"前へ向く気持ち"という思いを込めています。

© Akira Ikeda

JaSST'19 Hokkaido のテーマ(抜粋)

テスト設計技法には，基本的な技法から応用的な技法と，世の中には様々な技法が存在します。
これらの技法を見聞きし，学んでみたものの，なかなか使う機会がない，使ったことがない，と嘆いていませんか？
技法を実際に使用するためには，その技法の目的・用途を良く知る必要があります。
そして，それらの**技法を実務で適用するためには，事前にいくつか準備することがあります**。
個人の暗黙知になっている場合もあれば，組織の形式知になっている場合もあります。
今回のシンポジウムでは，**技法を適用する前に行うことはなにか？**ということを考え，技法を"知っている"から，"使える"という一つ上の段階にあがるための一助になれば幸いです。

© Akira Ikeda

本セッションでのテーマ設定

テスト設計技法を知っていたら十分と思っていないかい？

↓

テスト設計技法以外のことをお話しする

技法を**実務**で適用するためには，事前にいくつか準備することがあります
技法を適用する前に行なうことはなにか？

↓

テスト設計技法を適用する前のことをお話しする

※”マインドアップ”は造語になります。
”知力の向上”，”前へ向く気持ち”という思いを込めています。

↓

スキルアップについても考えてみる

© Akira Ikeda

アブストラクトの再共有

- 日々の業務でテスト設計技法を上手く使えないと相談を受けることが少なくありません。相談者は決して努力していないわけでは無く，テストに関する研修を受けたり，書籍を読んで勉強をしたりしています。それにもかかわらずそのような悩みを持つ方がいらっしゃいます。テスト設計技法を上手く使うためには，テスト設計技法そのものの理解を深める以外に，事前にやっておくべきことや考慮すべきことがあります。テスト設計技法を使う局面という足下だけを見ていても難しく，顔を上げてテストプロセス上の“それ以前”，つまりテスト分析設計に目を向けることが大切です。そして，顔を上げるだけではなく，テスト分析設計はもちろんテストそのものに取り組むための意識向上が必要となります。それらに取り組む先で，テスト設計技法をうまく使うためのマインドが得られることでしょう。

© Akira Ikeda

アブストラクトの再共有

- 本基調講演では，JaSST'19 Hokkaido のメインテーマである「マインドアップ」につなげていくために「フェイスアップ」「ビルドアップ」という観点から，これまでの自身の経験も交えながらお話しいたします。
- 「フェイスアップ」パートでは，テスト設計技法を上手く使うために，テストプロセスの視点から，事前に取り組むべきことについて，2019年4月に発行した「[改訂新版]マインドマップから始めるソフトウェアテスト」をベースにお話します。
- 「ビルドアップ」パートでは，テスト設計技法を使えるエンジニアになるため/育てるために活用できる技術やコミュニティ等を紹介し，今後どのように取り組んでいくべきかを考えます。
- 90分という短い時間ですが，一緒に「その先のマインドアップ」の姿を捉えましょう！

© Akira Ikeda

Mind set!

あなたがUPしたいマインドとは？

あなたが考えるマインドアップとは？

考えてみよう　→付箋に書いてみよう

- どのようなマインドをアップしたいですか？
- どのようなマインドをアップすることが大切でしょうか？
- などなど，マインドアップについて自分なりのイメージを持ちましょう

共有してみよう

- 同じテーブルの方と共有しましょう

© Akira Ikeda

考えてみよう！

本題に入る前に頭の体操をしましょう

考えてみよう！

あなたは，ある日
プロ野球球団の投手として働くことになりました

あなたのミッションは試合の勝利に貢献することです

一日野球教室にて球種は教わりました

さて，それだけで試合に勝てるでしょうか？

© Akira Ikeda

【演習】考えてみよう！

- 勝つためにはどういった事が必要でしょうか？
- あなたが“投手として” やっておくべき事を考えてみましょう

#	やっておくべきこと	時期
1		
2		
3		
4		
5		

© Akira Ikeda

第0部
まずは足下

まずは実務でのテスト設計技法あれこれを
確認しましょう

【演習】なぜテスト設計技法を使いたいのか？

- 我々はなぜテスト設計技法を使いたいのでしょう？
- その理由をあらためて考えてみましょう

#	使いたい理由
1	
2	
3	
4	
5	

© Akira Ikeda

なぜテスト設計技法を使いたいのか

- テスト設計技法を使うと，妥当なテストケースにコントロールできる
  - テストケースが減る：テストをやりすぎてたいた状況を改善
  - テストケースが増える：これまでのテストケースが少なすぎた
- テスト設計技法を使うと，テストケースの偏りを改善できる
  - 組み合わせテストなどにおいて，満遍なテストケースのパターン配置が得られる
- テスト設計技法を使うと，テストケースを自信を持って提案できる
  - 「頑張って考えました」ではなく，「こういった論理に基づいて作成しました」と説明できると，テストケースの存在の根拠が強まる
    - 根拠が薄いテストはやる意味があるのか？
- その他の効果を期待できる
  - テスト設計技法を使うと，仕様の抜け漏れを発見できる
  - 仕様を多角的に理解できる（仕様を理解するためにテスト設計技法を使える）
- その他…

© Akira Ikeda

テストの研修を受けるが…

テスト設計技法をきちんと理解できなかった

テスト設計技法は理解したが，使いどころがわからない

© Akira Ikeda

きちんと理解できなかった

- 目的意識を持って受講していない
  - 改善したい課題を持っていないと表面的な理解にとどまってしまう
- 自分の業務のドメインへの適合度が低い
  - 例えば，Web系エンジニアが，制御系エンジニア向けの研修の内容を理解するのは苦労する
    - 課題や事例，設問などの例が，自分のドメインとあっていないなど
    - 講師の略歴が適合してしない可能性もある
- 前提知識が不足している
  - 数学の知識，ソフトウェア開発の知識，プログラミングの知識，ドメインの知識，その他…
    - 例えば，制御フローテストでは，グラフを知っておく必要がある
    - 例えば，状態遷移テストでは，グラフに加え行列を知っておく必要がある
    - 例えば，ランダムテストであれば，確率統計の知識が必要だろう
    - 例えば，サーバ・クライアント方式のテストは，開発方式の知識がないと理解が難しい
    - 例えば，ローカライゼーションテストは，その国の言語や文化，法律などを知っておく必要がある
    - などなど…

© Akira Ikeda

余談：書籍も同様

- テスト設計技法に関してだけではないが，書籍を読んでもよくわからないとか，難しいとか，読みにくいと思ったとき，そのほとんどの場合は「その書籍を読むための実力が備わっていない」という原因がある

- 目的意識を持って読んでいない
  - 改善したい課題を持っていないと表面的な理解にとどまってしまう
- 自分の業務のドメインへの適合度が低い
  - 例えば，Web系エンジニアが，制御系エンジニア向けの書籍の内容を理解するのは苦労する
    - 課題や事例，設問などの例が，自分のドメインとあっていないなど
    - 著者の略歴が適合してしない可能性もある
- 読むための前提知識が不足している
  - 数学の知識，ソフトウェア開発の知識，プログラミングの知識，ドメインの知識，その他…
    - 例えば，制御フローテストでは，グラフを知っておく必要がある
    - 例えば，状態遷移テストでは，グラフに加え行列を知っておく必要がある
    - 例えば，ランダムテストであれば，確率統計の知識が必要だろう
    - 例えば，サーバ・クライアント方式のテストは，開発方式の知識がないと理解が難しい
    - 例えば，ローカライゼーションテストは，その国の言語や文化，法律などを知っておく必要がある
    - などなど…

© Akira Ikeda

余談：ワタシ オシエテモラウヒト の害悪

- どれだけ目的を持って，課題に合致した研修を受けたとしても，
  ワタシ オシエテモラウヒト という意識では何も身につかない
- 研修や勉強会では，自ら学び，講師から学び取るという強い意識が必要

- 申し込んだ時点で安心してしまい，お客さん気分になることは避けよう
  - お客さん気分になると，その研修の達成基準が「どれだけもてなされたか」「いい気分になったか」となってしまう
  - 俗に言う「**勉強会温泉**」状態になってしまう

- 参加者は，そのような状況になるとスキルアップは望めなくなるため，意識を変えよう
- 管理者は，このようなワタシオシエテモラウヒトという意識を持っていたり，勉強会温泉に浸かるのが目的という部下は候補から外すか，意識を変革させる必要がある
  （研修や勉強会に業務として参加するのは，工数とお金を消費する！）

© Akira Ikeda

使いどころがわからない

- 技法が解決できる課題を理解していない
  - テスト設計技法は必ず解決したい課題があって生み出されたものである
    - SQuBOKのトピック記述が参考になる
- テスト対象の理解が不十分である
  - テスト対象を理解していなければ，そもそもどのような技法を使えるかも思いつかない
    - 仕様を読んでない（眺めただけ）
    - 仕様の範囲や，仕様と仕様のつながりを理解していない
    - 仕様が書かれた思想や背景を抑えていない
    - 仕様として書かれていない暗黙の仕様に気がついていない
    - その他，テストに求められる要求や制約などを抑えていない
- 仕様を理解した上で，どのようなテストをすべきかという観点をまとめていない
    - 全体像がないと仕様を一行一行読み進めながら都度テストケースを作ろうという悪手に陥る
      - コピー＆ペースト程度のテストケースしか生み出されない（そもともテスト設計技法が適用されない）
      - 仕様記述上のキーワードと一致したテスト設計技法しか思いつかない

© Akira Ikeda

参考：SQuBOKにおけるトピックの構造

- 目的
- 方法
- 効果
- 留意点
- 関連知識領域／トピックス
- 参考文献
- 関連文献

研修では，多くの場合，方法（手順）と直接的な効果だけが示されることが多い
目的を押さえにくいので，使いどころがピンとこないということになる
また，設計技法について知っているから使えるようになるためにはその場の解説だけでは難しく，
現実的には複数の文献や論文などの情報もあたる必要がある

© Akira Ikeda

足下を踏む固めることが大切

テスト設計技法をきちんと理解できなかった

- 目的意識を持って受講しよう
- 自分のドメインに適合している研修を選ぼう
- 前提知識を学び直そう

もう一度足下を確認して，
不十分ならば踏み固めよう！

テスト設計技法は理解したが，使いどころがわからない

- 技法が解決できる課題を理解しよう
- テスト対象をよく理解しよう
- テストの観点の全体像をまとめよう

前準備が大切！
（本日の本題）

まずは足下固め，加えて，前準備に取り組もう

© Akira Ikeda

余談：必殺技症候群に陥るべからず

- 見た目の派手さに憧れて，まだ身につけられない技法を学ぼうとする
  - ホイミすら覚えていないのに，いきなりベホマやベホマズンを求める
    ※注：ホイミやベホマ，ベホマズンはドラゴンクエストに登場する呪文名，高度な呪文は高レベルにならないと習得できない
- そのテストでは使う必要がない設計技法も無理やり適用しようとしてしまう
  - 技法は知ってしまうと使いたくなる
- ブームが過ぎたので使わなくなる
  - テスト技術にもブームがあるが，過ぎ去ると興味を失い利用しなくなる

技法を沢山知っておくのは必要であるが，
着実に身につけ，
真っ当に利用し，
生かし続けることが重要である

© Akira Ikeda

第１部
Faceup
～顔を上げて，その前を意識～

テスト設計技法を利用する前に必要なテスト分析・設計と確認し，
やりかたの一例としてマインドマップ活用例を紹介します

テスト設計技法を使う前にある，テスト分析・設計

テスト設計技法を利用する前準備では
どうったことに取り組めばよいでしょうか

© Akira Ikeda

再掲：テスト設計技法は使うための前準備

1.技法が解決できる課題を理解しよう

2.テスト対象をよく理解しよう

3.テストの観点の全体像をまとめよう

1.は研修や勉強にて解決が可能
2.および3.は近年テストプロセスで対応が進んでいる

© Akira Ikeda

おさらい：V字モデルでのテストレベル

© Akira Ikeda

テストライフサイクルプロセスの例

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

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

© Akira Ikeda

V字モデルへライフサイクルをマッピング

© Akira Ikeda

「[改訂新版]マインドマップから始めるソフトウェアテスト」におけるプロセス

仕様分析	テスト計画	テスト設計	テスト実装	テスト実行	テスト報告
• どのようなテストを行う必要があるのかを検討 使う準備	• 仕様分析によって洗い出された情報を計画として作成する。	• テスト項目を洗い出し，テスト項目の組み合わせをテスト仕様としてまとめていきます。ここで**テスト設計技法を適用**します。 使う	• テスト仕様書の内容に基づいて，実行可能なレベルのテストケースを作成していきます。	• テストケースを実行するとともに，その結果を記録していきます。また，バグが発見された場合，その情報をインシデントレポート（バグ票）として発行します。	• テストの終了後，テストの実施結果を基に，テストリーダやプロジェクトマネージャ向けに報告書を作成します。

© Akira Ikeda

ISTQBシラバスにおけるテストプロセス

テスト計画	テストのモニタリングとコントロール	テスト分析	テスト設計	テスト実装	テスト実行	テスト完了
•テストの目的と，状況により課せられる制約内でテストの目的を達成するためのアプローチを定義する	•テスト計画書で定義したテストモニタリングのメトリクスを使用して，テスト計画書の内容と実際の進捗を継続的に比較する	•テスト可能なフィーチャーを識別し，テスト条件を決めるためにテストベースを分析する •何をテストするか，テストアイテムの確定	•テスト条件を高位レベルテストケース，高位レベルテストケースのセット，およびその他のテストウェアへ落とし込む •どうテストするか，高位テストケースを作成する，ここで**テスト技法を利用**する	•テスト実行に必要なテストウェアを作成，および／または完成させる •実行可能な完全なテストケースを作成する	•テスト実行スケジュールに従ってテストスイートを実行する	•完了したテストの全活動のデータを集め，プロジェクトから得たこと，テストウェア，およびその他の関連する情報すべてをまとめる

使う準備　使う

現在，国際的にも分析と設計に取り組むのは必須

© Akira Ikeda

その他，参考になるテストプロセス

- Test.SSFのプロセス
  - ASTERのサイトから参照可能
  - [改訂新版]マインドマップから始めるソフトウェアの3章コラム「Test.SSFにおけるテストプロセス」
- ASTER智美塾で検討されたプロセス
  - これまでのJaSSTでの発表資料が参照可能
- テスト設計コンテストのプロセス
  - テスト設計コンテストのチュートリアル資料などで参照可能

© Akira Ikeda

テスト分析・設計の要素

- 先に挙げたようなテストプロセスや，国内で提案・導入が進むテスト分析設計手法によって，実施することや手順，分担は変わるが，概ね以下のようなことを実施する

情報入手
- テストベースを入手する

→

情報理解
- 入手したテストベースを読み込む（純粋理解）
- テストベースをテストの立場で読み込む（テストとして理解）

→

テスト観点の全体像策定
- 読み込んだ内容を元にテストをモデル化
  - テスト戦略の策定
  - テストの観点の全体像を検討，モデル化（文書化）

→

テスト設計技法適用
- テスト設計技法を割り付け，利用してテストケースを作成

テスト分析	テスト設計	テスト実装
テスト分析	テスト設計	テスト実装
テスト分析	テスト設計	

© Akira Ikeda

テスト分析・設計の要素

- 先に挙げたようなテストプロセスや，テスト分析設計手法によって，実施することや手順，分担は変わるが，概ね以下のようなことを実施する

情報入手
- テストベースを入手する

→

情報理解
- 入手したテストベースを読み込む（純粋理解）
- テストベースをテストの立場で読み込む（テストの視点で理解）

→

テスト観点の全体像策定
- 読み込んだ内容を元にテストをモデル化
- テスト戦略の策定
- テストの観点の全体像を検討，モデル化（文書化）

→

テスト設計技法適用
- テスト設計技法を割り付け，利用してテストケースを作成

テスト設計技法を利用する前の準備として，朱文字のようなことを実施しておく必要がある

© Akira Ikeda

国内におけるテスト分析設計手法

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）

© Akira Ikeda

コツ：実務を考慮してテストプロセスを設計すべし

- テスト分析設計は様々なプロセスパターンや手法があるが…
  - 世の中には沢山の選択肢がありますが，現場の技術力や開発プロセス，社内規格・基準との整合性などの関係で，そのままではうまく利用できない場合がある
    - 基本的なテストプロセスやアクティビティへの知識がない
    - 社内で定めている開発規格や基準がテストを考慮していない，改訂する必要がある
    - 業界標準への準拠が必須である
      - 医用系はFDA，オートモーティブ系はA-SPICEに準拠していることを求められる
    - ビジネス形態によっては，開発・テストプロセスをカスタマイズできない
      - 顧客先プロセスに準拠することが契約の場合など
      - 開発プロセスを変えられない，納品成果物が規定されている，etc…

世の中のプロセスや手法をテンプレートとして，
必要に応じ，テストプロセスを設計するという考え方が重要

© Akira Ikeda

余談：アジャイル開発への対応はプラクティス目線で

- テスト分析設計を工程と捉えると対応が難しくなる
  - 特別なイテレーションやタイムボック，スプリントを設けるという発想になりやすいため

- テスト分析設計の要素を抑え，それを採用するアジャイル開発のフレームワークやプラクティスに導入する
  - 例えば，ストーリーカードにテスト分析・設計の要素を加える
  - 例えば，プロダクトバックログにテスト分析・設計の要素を加える
  - 例えば，TDDをVOTDDで実践する
  - Etc…

© Akira Ikeda

【演習】自分流のテスト分析設計を考えてみよう

- 実務の状況を踏まえ，自分流のテスト分析設計プロセスを考えてみよう

プロセス	やること	活用できる手法	成果物
テスト分析			
テスト設計			

© Akira Ikeda

テスト分析設計方法の一例～マインドマップを活用する～

テスト分析設計方法一例として
マインドマップによるテスト分析設計を紹介します

© Akira Ikeda

テスト分析と設計が無い場合

初級者

仕様書等

単なる転記

テストケース

(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>**

※チェックとテストの違いについては，本資料最後にある「参考スライド」を参照下さい

© Akira Ikeda

テスト分析と設計がある場合

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

© Akira Ikeda

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

© Akira Ikeda

実に多くを考えることが必要であるが…

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？

© Akira Ikeda

マインドマップの利用イメージ

初級者

仕様書

マインドマップを描く

テストケース

マインドマップを描くことで,
- 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

© Akira Ikeda

テスト分析・設計にマインドマップを活用

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

テスト分析・設計の1つのやり方として紹介します

© Akira Ikeda

© Akira Ikeda

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。

© Akira Ikeda

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♪

© Akira Ikeda

© Akira Ikeda

この例での「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
仕様等の理解・整理・検討・テスト観点の目付け

テスト設計
テスト観点の発想・検討・整理，テスト技法の適用

テスト実装
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）

テスト実行
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

テスト報告
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。

© Akira Ikeda

開発作業とテスト作業のおおまかな対比イメージ

© Akira Ikeda

テスト分析の作業と勘所（1）

テスト分析 → テスト設計

設計仕様やテストへの要求の理解・整理・検討（純粋理解）

- 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観点，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（テストの視点で理解）

- 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
- テストタイプの候補抽出
- 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

© Akira Ikeda

テスト分析の作業と勘所（2）

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

© Akira Ikeda

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

© Akira Ikeda

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

テスト設計技法の適用

- 観点モデルの観点要素に対して対応するテスト設計技法をアサインし，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

© Akira Ikeda

余談：テスト観点に関するよくある発言

- いわゆる「テスト漏れ」に対してその理由を聞くと，
  「テストの観点が漏れていました」
  という発言をよく聞く
- テストの観点漏れを防ぐための対策に挙げられることが多いのは「テストレビューを充実します」という発言
- このときテストレビューの充実の意味は「テストレビューの回数を増やします」だったり「時間を増やします」だったり「有識者を増やします」であることが多い
- しかしながらこれは後手後手の対策
- テストの観点漏れを防ぎたいのであればレビューで防ぐのではなく，その前に対策を打っておく必要がある
- さて，あなたの現場では「テストの観点漏れに対してレビュー以外の手」を打っているだろうか？

© Akira Ikeda

© Akira Ikeda

マインドマップ適用時の基本的な作業手順と範囲

© Akira Ikeda

マインドマップ適用時の基本的な作業手順と範囲

© Akira Ikeda

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

© Akira Ikeda

マインドマップ適用時の基本的な作業手順と範囲

© Akira Ikeda

マインドマップ適用時の基本的な作業手順と範囲

© Akira Ikeda

デモムービー

© Akira Ikeda

マインドマップの例

観点を思いつき，そこにテスト設計技法を適用している

マインドマップの例

© Akira Ikeda

マインドマップの例

© Akira Ikeda

整理：マインドマップを使った全体の流れ

考えてみよう！

テスト分析・設計を導入して，単なる仕様チェックから卒業しよう！
- 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も考える
- 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
- テスト観点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
- テストの全体像をまとめ，テスト設計技法を適用してテストケースを作る

© Akira Ikeda

再掲：国内におけるテスト分析設計手法

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）

© Akira Ikeda

コツ：テスト分析設計には思考に役立つ手法を複数持っておくとよい

- 6つの帽子
  - 帽子をかぶり直しながら考えることで思考の偏りを防ぐ
    - 白：客観的，赤：直感的，青：肯定的，黒：創造的，黄：否定的，緑：管理的
- 6W2H
  - 富士ゼロックス秋山浩一さんのHAYST法で利用されている視点
    - 開発者（Why, What, Howto)，ユーザー(When, Where, Who)，お客様のお客様(Whom, How much)
- 強制類推法
  - 刺激ワードを無理に組み合わせて発想を促す
  - 鈴木三紀夫さんの意地悪漢字やHAZOPのガイドワードなども使える
  - 組織によってはテスト観点キーワード集なども利用すると良い
- 類語置換法
  - 思いついたキーワードを類語に置き換えてみると違った発想が得られる
  - 一度書き上がったマインドマップに出現しているテスト観点キーワードを，類語辞典を使って置き換える

© Akira Ikeda

コツ：テスト分析設計には思考に役立つ手法を複数持っておくとよい

- 逆設定
  - 前提を強制的に反転して考えてみる
  - 仕様として書かれていることを“常識”としてとらえ，“非常識”から考えてみる
- 三銃士モデル
  - 世界観，コンテキスト，実装
- CIBFW
  - 電気通信大学西康晴先生による観点
  - 条件（Condition），テスト対象（test Item），ふるまい（Behaviour），嫌なこと（things Faulty or hazardous），世界（World）

© Akira Ikeda

コツ：マインドマップにこだわる必要なし 大切なのは，発想すること

- 探検ネット（花火）
  - KJ法で利用される発想技法
  - 見栄えはマインドマップとも似ている
- マンダラート
  - 3*3のマス目の真ん中にテーマを書き，そこから残り8マスを埋めるように発想する
  - [改訂新版]マインドマップから始めるソフトウェアテストにコラムあり
- はちのすボード
  - 基本的にはマンダラートと同じ
  - ハニカム構造のボードを使って発想を広ていく

© Akira Ikeda

コツ：マインドマップは発散，収束は別手法で

- マインドマップは思考の発散には向いているが，物事の整理にはあまり向いていない
- このため，収束プロセスは別の手法や記法を利用するのがよい
  - マインドマップ　→　ツリー図
  - マインドマップ　→　クラス図
  - マインドマップ　→　ＮＧＴ
  - マインドマップ　→　ＦＶ表
  - マインドマップ　→　テスト観点マトリクス
  - マインドマップ　→　連関図
  - マインドマップ　→　ネットワークモデル
  - マインドマップ　→　KJ法で収束
  - Etc…

© Akira Ikeda

【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】

© Akira Ikeda

プロジェクトへの導入例(1)

あるプロジェクトへの導入事例を紹介します

© Akira Ikeda

あるプロジェクトへの導入事例

© Akira Ikeda

複数人によるテスト分析＆テスト設計

全体の方向性の検討

分析＆設計する上での意識あわせを
マインドマップを共有作成して行う
- 大雑把な仕様やコンセプトの確認
- 使用するテストベースの確認
- 使用するテストカテゴリの確認
- スケジュール等，テスト計画の内容

etc…

分析＆設計の指針

複数人それぞれで分析＆設計

ベテラン

新人

他部門の有識者

複数人で分析＆設計を行うことによって，
多角的な検討を行う
- ベテランや新人
- ドメインごとのエキスパート
- 他部門の有識者

etc…

結果の集約

それぞれの分析＆設計結果を集約

複数人で作成した分析＆設計結果のマップをひとつのマップに集約する
- 集約過程におけるレビュー効果
- ノウハウの水平展開（教育効果）
- テスト分析＆設計と戦略の意識統一

etc…

© Akira Ikeda

現場担当者に聞く，主な得られた効果

テスト観点の全体像が見える

- どこに重きを置くか，どことどこを組み合わせるかの検討が容易に
- 観点ごとに担当者を決めるなど共同作業がやりやすくなる

2. テスト技法が適用し易くなる

- テスト観点ごとに，適用し易いテスト技法をマッピング
  - テスト観点・テストタイプと関連している技法を選択できる

3. 担当者における，テスト意図の見える化

- 各担当者の思考の流れが描かれることで，結果に対する理由が見やすくなる
  - なぜそのテスト観点なのか
  - なぜその階層関係なのか
- 他人の意図が見えるだけではなく自分の意図も見える
  - セルフレビューができる

4. テストに対する意識とモチベーションの向上

- テストは設計されるものであるという認識
- 目に見える成果物が絵で描かれるため，楽しい
- クリエイティブな活動をしているという誇り

© Akira Ikeda

現場担当者に聞く，その他の効果

1. 仕様書のレビュー効果

- 違った観点からの分析により，仕様における漏れ抜けを発見
- 設計部門が仕様書の見直しを行うことで，手戻りの減少

2. テスト戦略へのフィードバック

- テスト観点のモデリングを行うことで，戦略をもったテスト計画が作成可能に
- テスト観点の重要度からテスト実行スケジュールを検討するなど

3. テストケースの増減

- 増えたところ
  - そもそも抜けていたテスト観点や組み合わせに従ったテストケースが増加
    →テストの洩れの防止
- 減ったところ
  - テスト観点の重要度・優先度づけにより，無駄に作りすぎていたテストケースの削減
    →テストの効率化

4. テスト担当者のスキルアップ

- ベテランの描いたマップを参照することで，自分にない“観点”の獲得
- ビギナーの思考が見えることで，適切なアドバイスができる

© Akira Ikeda

過去のJaSST北海道での関連事例

年度別　事例発表一覧

年度	事例／一般発表	
2006	札幌地域におけるユーザビリティテストの普及に向けて	平沢 尚毅 （小樽商科大学）
	複合窓口システム」におけるユーザビリティテスト	佐々木 誠司 （小樽商科大学）
	ユーザビリティラボのご紹介	平沢 尚毅 （小樽商科大学）
	レビュープロセスの現実的な改善手段の提案	安達 賢二 （HBA）
	なんちゃってXPのすすめ	
2007	要求仕様とテストの〇〇な関係 －モレを防ぐ－	
	一歩踏み込んだテスト計画 －SEPGの事例－	
2008	ソフトウェア品質知識体系ガイドSQuBOK®のススメ	
	TestLink はじめの一歩	
	TEF札幌テスト勉強会presents：実践テストプロセス	高木 進也（TEF札幌テスト勉強会）
2009	マインドマップ活用によるシステムテストの改善 - 図解表現技法による理解の共有化と発想力の向上 -	坂本 正（NTTデータMSE）
	White UIオートメーションライブラリとIronPythonを用いた機能テストの実践	岩田 靖史（日本テレソン）
	保守開発における品質改善活動事例 - 金融機関ホストシステム -	吉田 和晃 （日本アイ・ビー・エム・ソリューション・サービス）

・一般発表

マインドマップ活用によるシステムテストの改善
- 図解表現技法による理解の共有化と発想力の向上 -
坂本 正（NTTデータMSE）

論文（PDF: 219KB）
講演資料（PDF: 1.725KB）

© Akira Ikeda

過去のJaSST北海道での関連事例

年度別　事例発表一覧

年度	事例／一般発表	
2006	札幌地域におけるユーザビリティテストの普及に向けて	平沢 尚毅 （小樽商科大学）
	複合窓口システム」におけるユーザビリティテスト	佐々木 誠司 （小樽商科大学）
	ユーザビリティラボのご紹介	平沢 尚毅
	レビュープロセスの現実的な改善手段の提案	安達 賢二
	なんちゃってXPのすすめ	出羽 弘明 ア・テク
2007	要求仕様とテストの○○な関係 －モレを防ぐ－	根本 紀之 ア・テク
	一歩踏み込んだテスト計画 －SEPGの事例－	奥村 有紀
2008	ソフトウェア品質知識体系ガイドSQuBOK®のススメ	大西 建児 （立命館）
	TestLink はじめの一歩	梶野 由貴子 （BeauchiT）
	TEF札幌テスト勉強会presents：実践テストプロセス	上田 和樹 (TEF札幌テスト勉強会) 高木 進也 (TEF札幌テスト勉強会)
2009	マインドマップ活用によるシステムテストの改善 - 図解表現技法による理解の共有化と発想力の向上 -	坂本 正 （NTTデータMSE）
	White UIオートメーションライブラリとIronPythonを用いた機能テストの実践	岩田 靖史 （日本ナレッジ）
	保守開発における品質改善活動事例 - 金融機関ホストシステム -	吉田 和晃 （日本アイ・ビー・エム・ソリューション・サービス）
	潜在不具合発生パターンとテスト設計技法	上野 信 （高度ソフトウェア・テスト技術研究会）

© Akira Ikeda

JaSSTのサイトは事例の宝の山

- JaSSTにはマインドマップの適用事例の他，テストプロセスに関する沢山の事例情報が掲載されています
- 是非宝探ししよう！

プロジェクトへの導入例(2)

VOTDDにおけるマインドマップの導入事例を紹介します

© Akira Ikeda

あるVOTDDを採用したプロジェクトへの導入例

- VOTDDとは検証指向TDDのことで，TDDのテストコードの保守性やテストの網羅性の充実を目的とした手法

© Akira Ikeda

あるVOTDDを採用したプロジェクトへの導入例

VOTDDのサイクル

VERIFY & DEBUG の追加

- テスト設計を洗練する
  - テスト設計の見直しを行なう
    - テストの抜け漏れを防止する
    - テスト設計の最適化
    - テスト設計のレビュー効果により，プロダクト仕様の欠陥や改善点を抽出

REFACTORへのTESTの追加

- テストコードを改善する
  - テストコードの保守性を高める

© Akira Ikeda

テスト設計の全体像設計と共有のためのマインドマップ導入

課題

- VOTDDは開発者とテスト技術者が二人三脚で実施するが，テスト設計の観点や意図の共有がテストコードのみではなかなか難しかった

対応

- テストコードの近くにテスト設計の観点や意図を置く（記入しておく）
- できればコメントではなくて，全体像を概観できると良い
- テストコードの近くにマインドマップを置いておくことで，テストコードの全体像の設計や把握を助ける

解決策

- このVOTDDでは，DoxyGenを使っていたため，テストコードの近くにマインドマップをコードとして置いた
- マインドマップはPlantUML で記述
- 開発者はもともとPlantUMLでプロダクトモデルを描いていたため，導入障壁も低かった

© Akira Ikeda

現場担当者に聞く，主な得られた効果

1. VERFY & DEBUG のスピードが向上した

- テストコードのそばにテスト設計があるため，すぐに参照できるようになった

2. テストコードの全体像が見えるようになった

- テストコードが増えると，全体として何をテストしたいのかがわからなくなる
- マインドマップがあることで，テストコードを全件追わずとも全体像が見えるようになった

3. 開発者がよりテストを意識するようになった

- 導入前はテスト観点モデルはテスト技術者のみの情報になりがちであった
- テストコードの近くに見えることで，開発者がよりよいテストを考えるように意識誘導された

4. テスト技術者の貢献がコードに示されるようになった

- VOTDDだと，テスト技術者の貢献はテストコードに反映されるが，その維持主体は開発者であるためテスト技術者の貢献がわかりづらかった
- 開発者が（主には）扱わないテスト設計が，開発者と同じ土俵であるコードという形で見えるようになるため，開発者に評価されやすくなった

© Akira Ikeda

ご参考：テスト観点モデルの作成ステップ

実導入の経験から例を紹介します

© Akira Ikeda

現場適用での反省：初級者には一気にテスト観点モデルを完成させるのは難しかった

- 某プロジェクトにおいて，テスト初級担当者に対して，テスト設計のトライアルにチャレンジしたが，うまくいかなかった

- よくよくヒアリングしてみると，次のようなことが“実践では”問題となっていた
  - マインドマップに慣れていなかった
  - マインドマップを使って考えろといわれても，何をどういった順番で考えれば良いかわからない
  - マインドマップを描いたとしても，先輩達が望む，組織のこれまでの知見などが反映されていない

- 初級者に「マインドマップでテスト設計して」といっても難しい

テスト観点モデルを作成する際の手順を
初級者向けに形式化して
ステップbyステップで進められるようにした

© Akira Ikeda

テスト観点モデルの作り方（その１）

- 初級者向けのテスト設計プロセスはＣＧ制作プロセスにヒントを得て形式化
  - 段階的にモデルを育てる（素のモデルに，プロジェクト制約や組織・ドメイン知識をまぶす
  - モデルのレビューをきちんと入れる，テスト設計書フォーマットと対応づける

モデリング
- テストベースから自由な発想でテスト観点を発想

レンダリング
- レンダリング指示書に従い，テスト観点モデルに対して以下を実施
  - エフェクト（制約情報やドメイン観点）を追加
  - フィルタ（観点や関連，リスク順位や重要度等の強弱）を反映

カメラテスト
- レンダリング結果に対し公式レビューを実施，テスト観点絵図として確定

ファイナライズ
- テスト観点絵図を所定のテスト設計書フォーマット（テンプレート）に焼き込み

© Akira Ikeda

初級者向けテスト観点モデルの作り方（その２）

- モデリング
  - 仕様書等直接のテストベースを使って，自由な発想でテスト観点モデルを作成
    - CG的には，生ポリゴンのモデルやワイヤフレームが出来上がるイメージ
- レンダリング
  - モデルに対し，レンダリング指示書に従い，「エフェクト」と「フィルタ」を追加・適用し全体を調整
    - エフェクトとはそのプロジェクトが持つ固有の情報や観点で，プロジェクト制約や製品ドメイン観点など
      - CG的には，テクスチャやフォグとか光源とかパーティクルやオブジェクトを追加するイメージ
    - フィルタとはテスト観点や関連が持つ属性の強弱を調整するもので，テスト観点の強度や関連の太さ，リスクの強弱といったものを調整
      - CG的には，明るさやコントラストを調整したりセピア風やキャンバス風に変換したり，被写界深度を設定したりといったイメージ
  - レンダリング指示書ではどのエフェクト・フィルタを適用するのか，どういった順番かといったことを指定する
    - なお，このエフェクトとフィルタはそれぞれ「エフェクトモジュール」と「フィルタモジュール」として別途作成，これらモジュールを独立性高く作成することで再利用がきくようになる（例：FPGAコントローラ観点エフェクト，組込み系汎用エフェクト，高信頼性サーバ向けリスクフィルタ）

© Akira Ikeda

初級者向けテスト観点モデルの作り方（その３）

- カメラテスト（プレビュー）
  - レンダリング工程を経たテスト観点マインドマップについて公式レビューを行う
    - レンダリング結果を様々なカメラから確認する。この時，カメラをどの位置や角度に置くのかが重要。このカメラとは即ちレビューの方針や観点。また，基本的に通常のレビューと同じく様々なメンバで構成されることが望ましい
- ファイナライズ
  - テスト観点絵図（マインドマップ）を組織で規定されたフォーマットやテンプレートに焼き込み（移しかえ）
    - このとき，焼きこみに失敗しないように注意すると共に，ちゃんと焼き込めたかの確認が必要

© Akira Ikeda

コツ：テスト観点（ノウハウ）の蓄積

- テスト観点をためていくことでテスト設計でのノウハウを蓄積していくことができる
  - エフェクトとフィルタというそれぞれの分類ごとにモジュールを作成してためる
- エフェクト，フィルタは（現在のところ）形式は規定していない
  - エフェクトは基本的にキーワードレベル
    - キーワード一覧の体を取る（構造も持たせる場合はツリーやマインドマップ）
    - エフェクトに記載されるキーワードは過去のテスト観点モデルから抽出，整理したもの
    - その他，過去の欠陥情報からも抽出可能
    - 品質特性や規格は独立してエフェクトとしても良い
  - フィルタは方針を表す文章レベル（キーワード抽出しても良い）
    - フィルタはプロジェクト情報や製品戦略，品質目標や部門方針などから導かれる
    - その他，リスク分析情報からも
- その他，テスト観点キーワードリストやテスト観点・テストタイプ・技法対応表などを整備するのも良い

© Akira Ikeda

コツ：あじさい問題

- テスト観点キーワードは便利だが，誰しもが同じイメージを持てるものまで落ちていないと，メンバ間でモデルの認識がブレる
- 例えば，「あじさい」というキーワードがあったとして，読み手によりイメージが異なる
  - 水色もあれば，紫もあり，白もあれば，桃色もある
    - 土壌のアルミニウムイオンにより変わる
  - 花や葉の形が異なる
    - 植物分類学上，さらに分類がある

- テスト設計にマインドマップを使うことの意図のひとつに「本来のマインドマップのようにビジュアルを多用することにより，より具体的なイメージでテスト観点を共有する」ことがある
  - マインドマップを使ったとしても，キーワードベースだと，イメージのブレが発生しやすい
  - イラストや図表を多用することで，イメージのブレを押さえ込む
    - あじさいの絵があれば具体的なイメージとなるし，キーワードに色をつけるだけでも色情報をつけられる

- テスト観点キーワードリストのようなものはたんなるリストだと使い物にならないことに注意する

© Akira Ikeda

さらにその前に考えておくべきこと

テスト分析の前にまだやるべきことはあるのだ

© Akira Ikeda

実際の所，それだけでよいのか

- 先ほどの話は十分な品質や目的にかなったテストベースがインプットされる前提になっているが，現実問題としてテストベースの入手に関する問題がある
  - 品質が悪く，テスト分析しようがない
  - テストベースの量が足りない，そもそもない
  - テスト分析がしっかり行えないと言うことは，後に続くテスト設計や実装も行えないという事である（テスト設計技法では解決できない）

- この問題を解決しないかぎり，そもそもテストプロセスだけの改善ではらちが空かないし，未来永劫同じ問題を抱え続ける

© Akira Ikeda

よりよいテストベースを入手するために

- 品質が悪いテストベースは受け入れない
  - テストベース受け入れレビューを実施する
    - ドキュメントとしての体を成していないものは受け入れない
- テストベースパッケージを定義して，構成管理をする
  - 入手すべきテストベースを事前に識別する
    - 入手すべきは対応する仕様書だけではないはず，十分な量（種類）を確保する
  - パッケージとその構成物は，構成管理を実施して常に最新の情報を得る
    - テストプロセス中にテストベースに変更がかかることは（現実として）よくある

© Akira Ikeda

開発レビューに参画する

- 開発レビューに参画して，仕様書の充実度を向上する
  - 記述自体の充実度や正確性等に加えて，テストに役立つ情報も充実させる
    - ただし記述を混ぜると可読性を悪くしてしまう場合があるので，別紙とするなど工夫が必要
  - テスタビリティ視点からの指摘
    - その仕様はテスト可能か，テストしやすいか，いくつかの手段があるか，テストのための機能はあるか，etc…
  - 過去の欠陥から，設計にフィードバック
    - 欠陥を作り込んだ・作りやすい仕様や構造についてフィードバックし，設計変更してもらう
  - テスト分析の前倒し
    - テスト分析に入る前に，先行で仕様などについて学習の機会となる

© Akira Ikeda

開発レビュー参画時の注意点

- 開発やドメインについての知識がないままにレビューに参画すると次のような事が起きがち
  - 発言できない
  - 誤字脱字程度の指摘しかできず，煙たがられる
  - 開発や技術等の背景を抑えられないまま斜め上のコメントをしてしまう
  - このような状況になると，いずれ参加を拒否されるようになってしまう

- 開発レビューに有効に参画するには
  - 相手の土俵に乗る
  - 開発やドメインの知識を仕入れておく
  - 仕様や設計方式にも（テストの知識を生かして）コメントする
  - 開発者と同等のレビューの技術を持っておく
  - また，開発レビューに参画する場合，テスト担当者も工数を確保しておく必要がある

© Akira Ikeda

そのほか，テストベースとして忘れがちな情報

- マスターテスト計画
  - マスターテスト計画にテスト設計に関連する情報あり
- プロジェクト計画
  - プロジェクト計画に，テストの方針として必要な情報が書かれていることが多い
- 上位の設計ドキュメントや要求
  - 仕様の内容によっては，上位の情報が必要になる場合もある
    - 各クラス仕様の他に，全体クラス図が必要，等
- 用語集，規約等
  - 当該プロジェクトで利用される用語や規約などはテストベースを読み解くために必要となる
- 打ち合わせメモ，QAシート
  - 直接仕様書等に記載されないテストに関す情報が存在する場合がある
- 先行・後行テストレベルでの情報
  - 先行するテスト設計や実行結果の情報によって，テスト設計に影響がある場合がある
  - 後行するテストレベルで実行するテストが，納品のために確認テストとして必要な場合がある

© Akira Ikeda

余談：雇用形態によるテストベースの入手の難易度

- 社員
  - 基本的には全ての情報にアクセス可能
- 派遣社員
  - 派遣先の機微な情報については非開示の場合が多い
- 請負社員
  - 契約範囲の情報のみに限定される

- 契約以上の情報入手や相手先へのアクセスは法律にふれる場合がある
- 前もって，入手できるように手配しておこう
  - 請負契約の場合は特に重要になる
    - 事前に請負契約の開示情報に盛り込んでおく必要がある
    - このためには日頃から営業部門や調達部門等とのコミュニケーションが必要

© Akira Ikeda

第1部のまとめ

© Akira Ikeda

第１部のまとめ

テスト設計技法を適用する前の準備として，テスト分析とテスト設計がある

テスト分析設計のやり方のひとつにマインドマップを活用する方法がある

テスト分析の前にもさらに必要な準備もある

© Akira Ikeda

第２部
Buildup

～意識を高めよう，鍛錬しよう～

テスト設計技法を使えるようになるための
ガイドとなる情報をお話しします

見つめ直そうテストの意義

もう一度，「なぜ我々はテストしなければならないのか」「なぜテスト設計技法をうまく使わなければならないのか」を考えてみよう

© Akira Ikeda

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

© Akira Ikeda

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，ソフトウェアをお客様が安心して利用することができるようになる．」（Akira Ikeda, Mikio Suzuki, 2019）

- 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

© Akira Ikeda

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
  - 電波の受信レベルは最高なのに，メッセージが送信できない
  - メニューやメッセージが文字化けする
  - アドレス帳に登録ができない，勝手にデータが消える
  - 目覚ましを設定しているのに，時間通りにアラームが鳴らない
  - 操作中に無反応になったり，勝手に電源が落ちる

- 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

© Akira Ikeda

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

© Akira Ikeda

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！

© Akira Ikeda

(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

© Akira Ikeda

(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。

© Akira Ikeda
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

© Akira Ikeda

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，ソフトウェアをお客様が安心して利用することができるようになる．」（Akira Ikeda, Mikio Suzuki, 2019）

はて？
なんだか意義が足りないような･･･？

© Akira Ikeda

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，ソフトウェアをお客様が安心して利用することができるようになる。
　また，リリース・出荷後にバグが出ないことで，ソフトウェアの回収や修正などに必要なコストを抑制し，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, 2019）

- 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

このための武器の1つとして，テスト設計技法やテスト分析設計技法があるということ

© Akira Ikeda

テスト設計技法をもう一度学ぼう

知識が古くなっていたり，誤った理解だったりするかもしれません

知識を更新しましょう

© Akira Ikeda

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！

© Akira Ikeda

SQuBOKガイド V2に見るテスト設計技法

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

© Akira Ikeda

SQuBOKガイド V2に見るテスト設計技法

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

- -

© Akira Ikeda

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）

© Akira Ikeda

テスト設計技法は組み合わせる

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

テスト設計技法を学ぶための書籍

はじめて学ぶソフトウェアテスト技法

- Lee Copeland 著／宗 雅彦 訳／日経BP社／2005年
- テスト技法と言えば，境界値テストしか思いつかない人が残念ながらいます。テストにはさまざまな技法が存在します。この本は多くの技法を取り上げ，演習問題を通して学ぶことができます。

ソフトウェアテスト実践ワークブック

- Rex Black 著／成田光彰 訳／日経BP社／2005年
- この本もテスト技法を演習を通して学ぶことができます。特長は，仮想のプロジェクト事例を使って，さまざまなテスト技法を紹介していることです。テスト技法を実際のプロジェクトに適用するヒントが得られます。

ソフトウェアテスト技法ドリル　テスト設計の考え方と実際

- 秋山浩一 著／日科技連出版社／2010年
- ソフトウェアテストを点・線・面・立体でとらえ，例題を交えながら丁寧に解説しています。テスト設計技法をガッチリと学びたいときに最適です。

「[改訂新版]マインドマップ]から始めるソフトウェアテスト」から引用

© Akira Ikeda

テストプロセス改善モデルや資格を利用しよう

実務で活用できるように技術力を向上するにあたっては，ガイドを利用するのもよいでしょう

© Akira Ikeda

TPI NEXT を利用してテスト設計技術を向上する

テストプロセス改善のための技術としてTPI NEXT がある

TPI NEXT はテストプロセスを16個のキーエリアに分け，それぞれに設定されているチェックポイントにより判定によって，成熟度を計測する

TPI NEXT はクラスタという概念を持ち，テストプロセスを改善・成熟させていくためのガイドとして利用できる

© Akira Ikeda

TPI NEXT のテスト成熟度マトリクス

#	キーエリア	グループ	初期レベル	コントロールされたレベル				効率化されたレベル				最適化したレベル		
K01	利害関係者のコミットメント	SR		1	2	3	4	1	2	3		1	2	3
K02	関与の度合い	SR		1	2	3	4	1	2	3		1	2	
K03	テスト戦略	SR		1	2	3	4	1	2	3		1	2	
K04	テスト組織	SR		1	2	3	4	1	2	3	4	1	2	3
K05	コミュニケーション	SR		1	2	3	4	1	2	3		1	2	
K06	報告	SR		1	2	3		1	2	3		1	2	
K07	テストプロセス管理	TM		1	2	3	4	1	2	3		1	2	
K08	見積もりと計画	TM		1	2	3	4	1	2	3	4	1	2	3
K09	メトリクス	TM		1	2	3		1	2	3	4	1	2	
K10	欠陥管理	TM		1	2	3	4	1	2	3	4	1	2	3
K11	テストウェア管理	TM		1	2	3	4	1	2	3		1	2	3
K12	手法の実践	TP		1	2	3		1	2	3	4	1	2	
K13	テスト担当者のプロ意識	TP		1	2	3	4	1	2	3	4	1	2	3
K14	テストケース設計	TP		1	2	3		1	2	3	4	1	2	3
K15	テストツール	TP		1	2	3		1	2	3	4	1	2	3
K16	テスト環境	TP		1	2	3	4	1	2	3	4	1	2	3

© Akira Ikeda

テストケース設計のチェックポイント

- コントロールレベル
  1. テストケースを論理レベルで記録している。
  2. テストケースには以下の説明項目を含む。
     a. 開始時の状況
     b. 変更プロセス＝実施するテストアクション
     c. 予測される結果
  3. テストケースにシステムの詳細な振る舞いを記述することで，テストベースのどの箇所がテストの対象であるかが把握できる。

© Akira Ikeda

テストケース設計のチェックポイント

- 効率化レベル
  1. テストケースは，テスト組織の同僚が見ても理解でき，保守できるようなものになっている。
  2. テストケースによって達成できるテストベースのカバレッジレベルが明確である。
  3. **テストケース設計に正式な技法を用いている。**
  4. テストケースが設計できないような品質特性のテスト作業には，チェックリストを用いている。

© Akira Ikeda

テストケース設計のチェックポイント

- 最適化レベル
  1. 次のフェーズ（次のテストレベルや本番）で発生した欠陥を分析し，テストケースの正確性や有効性の向上につなげている。
  2. テストケースそれぞれの妥当性と保守性についてチェックし，評価している。
  3. テスト設計技法を，将来さらに再利用するために評価し，調整をしている。

© Akira Ikeda

TPI NEXT のクラスタ

#	キーエリア	グループ
K01	利害関係者のコミットメント	SR
K02	関与の度合い	SR
K03	テスト戦略	SR
K04	テスト組織	SR
K05	コミュニケーション	SR
K06	報告	SR
K07	テストプロセス管理	TM
K08	見積もりと計画	TM
K09	メトリクス	TM
K10	欠陥管理	TM
K11	テストウェア管理	TM
K12	手法の実践	TP
K13	テスト担当者のプロ意識	TP
K14	テストケース設計	TP
K15	テストツール	TP
K16	テスト環境	TP

初期レベル	コントロールされたレベル				効率化されたレベル				最適化したレベル		
	A	B	B	C	F	H	H		K	M	M
	A	B	C	E	H	H	J		L	L	
	A	A	B	E	F	F	H		K	L	
	A	D	D	E	I	I	J	J	K	L	L
	B	C	C	D	F	F	J		M	M	
	A	C	C		F	G	G		K	K	
	A	A	B	B	G	H	J		K	M	
	B	B	C	C	G	H	I	I	K	L	L
	C	C	D		G	H	H	I	K	K	
	A	A	B	D	F	F	H	J	K	L	L
	B	B	D	E	I	I	J		K	L	L
	C	D	E		F	H	J	J	M	M	
	D	D	E	E	G	G	I	I	K	K	M
	A	A	E		F	I	I	J	K	K	M
	E	E	E		F	G	G	I	L	M	M
	C	D	D	E	G	H	J	J	L	M	M

© Akira Ikeda

TPI NEXT のクラスタによる，テストケース設計技法を使う前にやるべきこと

#	キーエリア	グループ
K01	利害関係者のコミットメント	SR
K02	関与の度合い	SR
K03	テスト戦略	SR
K04	テスト組織	SR
K05	コミュニケーション	SR
K06	報告	SR
K07	テストプロセス管理	TM
K08	見積もりと計画	TM
K09	メトリクス	TM
K10	欠陥管理	TM
K11	テストウェア管理	TM
K12	手法の実践	TP
K13	テスト担当者のプロ意識	TP
K14	テストケース設計	TP
K15	テストツール	TP
K16	テスト環境	TP

初期レベル	コントロールされたレベル				効率化されたレベル				最適化したレベル		
	A	B	B	C	F	H	H		K	M	M
	A	B	C	E	H	H	J		L	L	
	A	A	B	E	F	F	H		K	L	
	A	D	D	E	I	I	J	J	K	L	L
	B	C	C	D	F	F	J		M	M	
	A	C	C		F	G	G		K	K	
	A	A	B	B	G	H	J		K	M	
	B	B	C	C	G	H	I	I	K	L	L
	C	C	D		G	H	H	I	K	K	
	A	A	B	D	F	F	H	J	K	L	L
	B	B	D	E	I	I	J		K	L	L
	C	D	E		F	H	J	J	M	M	
	D	D	E	E	G	G	I	I			
	A	A	E		F	I	I	J			
	E	E	E		F	G	G	I	L	M	M
	C	D	D	E	G	H	J	J	L	M	M

テストケース設計に正式な技法を用いている。

テストケース設計だけではなく，他のキーエリアも事前に成熟度を上げる必要あり

© Akira Ikeda

（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

© Akira Ikeda

ISTQB・JSTQBを利用してテスト設計技術を向上する

テスト技術者の国際認定資格としてISTQBがある

ISTQB はテスト技術者のスキルレベルを三段階に設定している

ISTQBではテスト設計技法について，特に扱っているものに，CERTIFIED TESTER，TEST ANALYST，TECHUNICAL TEST ANALYSTがあり，技術力向上のためのガイドとして利用できる

© Akira Ikeda

ISTQBにおけるスキルアップモデル

関心がある方は，JSTQBの関係者に聞いてみよう！

© Akira Ikeda

ベンダーのトレーニングを活用しよう

早急な技術立ち上げが必要な場合は，トレーニングを積極活用しよう

© Akira Ikeda

専門会社のセミナーを活用して具体例を得る

- 第三者検証会社はテスト専門会社ならではの実践的な内容である
  - これまでの多数のプロジェクト経験やノウハウからカリキュラムやテキストが作られている
  - 第三者検証会社社内の教育コンテンツと共用されている場合，より専門的な内容である（実務目線で検討されているから）
- ツールベンダーのトレーニングは，ツールベンダーのツールを念頭に置いた内容である
  - 学習したことを，現在ツールを使っているプロジェクトに適用しやすい
  - ツールの特性を踏まえた解説となっていることが多い
  - やりたいテスト，実行したいテストケースを，ツールではどのように設定すれば良いのかを答えてくれる

関心がある方は，
本日出展しているスポンサー企業に質問してみよう！

© Akira Ikeda

コミュニティを活用しよう

テスト技術者が集まるコミュニテイに参加して，
テストに関する高度な議論を行なおう

© Akira Ikeda

JaSST（ジャスト：ソフトウェアテストシンポジウム）

- ソフトウェアテスト技術を扱うシンポジウム
- 全国各地で開催されている
  - 北海道，東北，新潟，東京，東海，北陸，関西，四国，九州
  - レビューに特化したJaSSTレビューも昨年から開催
- テストの様々な手法などの発表に加え，チュートリアルで学ぶことができる
- 次回は2019年10月4日にJaSST’19 Tokaiが開催される

関心がある方は，JaSST実行委員に聞いてみよう！

WACATE
（ワカテ：ソフトウェアテストワークショップ）

- ソフトウェアテスト技術を扱う合宿型勉強会
- 毎年二回（夏期と冬期）に神奈川県三浦海岸で開催されている
- 夏は狭く深く，冬は広く浅く
- テストに関する手法などを議論と演習にて実践的に学ぶ
- 次回は2019年12月14～15日

関心がある方は，WACATE実行委員に聞いてみよう！

TEF-DO
（テフドウ：TEF道）

- 北海道で活動している，ソフトウェアテスト技術者のコミュニティ
- 不定期に勉強会を開催
- これまで，技術を検討し，論文発表などにも取り組んでいる

関心がある方は，TEF-DO関係者に聞いてみよう！

国内で参加できる主なイベントやコミュニティ

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/
- WACATE　http://wacate.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/

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（北海道）

- TEF-Do（ＴＥＦ北海道）　https://ameblo.jp/tef-do/

© Akira Ikeda

テスト設計コンテストを活用しよう

本気でテスト設計技法やテスト分析設計に取り組みたいならば，コンテストの活用は必須！

© Akira Ikeda

テスト設計コンテストとは

- テスト設計に焦点を当てたコンテスト
- テスト設計コンテストの目的
  - ソフトウェアテストを分析設計から行うことを周知し，ソフトウェアテストエンジニアに対する教育の機会を提供する
  - コンテストという形式をとることにより，ソフトウェアテストが創造的な作業であり，楽しいということを経験してもらい，若年層及び初級テストエンジニアからベテランテストエンジニアまでテストへの興味を高める
  - ソフトウェアテスト業界における技術開発を競技を通じ，促進する

© Akira Ikeda

テスト設計コンテストの成果物を活用しよう

- 2011年度以降のコンテストの成果物が無償で入手可能
- 多数の技術者が考え抜いた，様々な手法が提案されている
- コンテキストが近い手法を見つけて，勉強したり取り入れたりすることでテスト設計力をこうじょうすることができる
- また，自社とそれ以外のベンチマークを取ることができる
- テストカタマリーは本日体験が可能！

© Akira Ikeda

テスト設計コンテストに出場しよう

- テスト設計コンテストの出場をきっかけに自社の技術を向上する
- 実務の実態に対応するテスト設計手法を確立できる
- 審査員による，専門的コメントにより，手法をさらに磨ける
- テスト設計力を外部に示すことで，自社をアピールできる

関心がある方は，テスト設計コンテスト関係者に聞いてみよう！

© Akira Ikeda

社内でテスト技術者を育成しよう

技術者のスキル向上は，その企業や管理者の責任のひとつです

強いチームを作るためには，充実した教育支援が必須

© Akira Ikeda

社内でどうテスト技術者を育成するか

- 全社教育部門による，専門教育の企画と実施
  - ただし，研修会社に丸投げは良くない
    - 自社の状況に合わせて，共創してくれる研修会社を選ぼう
  - 一番よいのは自社講師を育てること
    - 自社内の生の事例が使える，講師はそのまま技術リーダーとなってくれる
- 社内勉強会の支援
  - 現場から勉強会実施の機運が高まった場合，開催を支援する
    - モチベーションが高いときが一番効果が高い
    - 工数やお金の面で支援が必要，自己啓発だと長続きしにくい
    - **安心して技術力向上に取り組んでもらうための制度作りは会社の責任**
- 社内イベントの実施
  - 定期的に社外の情報をインプットする
    - 実務者は忙しいので，新たなことを業務時間内に勉強する時間がとれないことが多い

© Akira Ikeda

社内でどうテスト技術者を育成するか

- 社外イベントへの参加支援
  - 社外で他社の技術者との交流は得るものが多い
    - 技術情報，事例，モチベーション，etc
    - 前もって全社で教育予算を抑えておくこと
- 社内スキル認定制度の確立
  - 社内にテスト技術者というロールを根付かせる
  - ISTQBやIVECといった社外の認定制度を活用するのもよい
- ソフトウェアテスト，品質についての文化作り
  - 目先の技術だけを追ってもいけない
  - 全員で品質を高める，テスト技術を活用するという文化がないと廃れてしまう

トレーニングについては，ISO/IEC/IEEE291119のテスト計画でも項目がある
テスト技術者を育成するのは，プロジェクトや企業の責務であることを認識すること

© Akira Ikeda

余談：発注者側も社内の取り組みを注視

- ソフトウェアテスト技術が広く普及していくにつれて，
  発注者側が発注にあたり，候補となる業者がどれくらいテストに取り組んでいるかを見るようになってきている
- 単なる実行ではなくて，テスト分析設計など専門的に実施する能力があるのかどうかを見るようになってきている
- 企業としてどれくらいの品質やテストに関心があるかも見ている
  - JSTQBやIVEC，JCSQEなどの資格取得に取り組んでいるか
  - 勉強会やイベントに技術者を派遣しているか
  - 勉強会やイベントに投稿しているか，発表しているか
  - コミュニティや団体に委員を派遣しているか，スターエンジニアが存在するか
  - イベントにスポンサー参加しているか，自社イベントを行なっているか

先端的な企業は，テストの専門性を求めるようになってきている
テストに取り組み，それを社外にアピールしないと受注できなくなる未来はすぐそこに

© Akira Ikeda

本日を有効活用しよう

シンポジウムには沢山の有識者が参加しています。
積極的に交流しよう！

© Akira Ikeda

沢山の人と交流しよう！

- スポンサー出展されているベンダーの方々
- 講演者
- コミュニティ参加者
- テスト設計コンテスト出場経験者
- Etc…

沢山の人と交流し，刺激を受け，様々な情報を交換し，
マインドを高め合うことが重要である！

© Akira Ikeda

テスト技術者の今後

今後，我々はどうなっていくべきなのか

© Akira Ikeda

今後のテスト技術者に求められるもの マインドアップが重要である

- テスト分析設計力が進むと，モデリング能力が求められる
- 自動化が進むと，プログラミング能力が求められる
- 開発者とテスト技術者の融合が進むと，システム・ソフトウェア開発能力が求められる
- グローバル化が進むと，英語が求められる
- テストをビジネスとしていく上で，経営能力が求められる

...etc

テスト技術者が増えると，技術者同士の競争も激化する
また，テストに関するビジネスが拡大すると，持つべきスキルも拡大する
今後はテスト技術者同士の切磋琢磨がより重要になってくる

そこに“マインドアップ”が必要となる

© Akira Ikeda

第2部のまとめ

© Akira Ikeda

第2部のまとめ

なぜテスト設計技法が必要なのか，テスト分析設計が必要なのか，なぜテストを行なわなければならないのか，といった事を今一度意識し直そう

技術力を高めていくためには，テストプロセス改善モデル，テスト国際認定資格，ベンダのトレーニング，コミュニティ，テスト設計コンテスト，
そして本日を活用しよう

個人だけではなく，企業も真剣に技術力向上に取り組もう

本日ここに集まっている人と知見を最大限に活用しよう

© Akira Ikeda

Mindup!

あらためて，あなたがUPしたいマインドとは

あなたが考えるマインドアップとは？

考えてみよう　→付箋に書いてみよう

- どのようなマインドをアップしたいですか？
- どのようなマインドをアップすることが大切でしょうか？
- などなど，マインドアップについて自分なりのイメージを持ちましょう

共有してみよう

- 同じテーブルの方と共有しましょう

© Akira Ikeda

まとめ

本日お話したこと

- 第０部：まずは足元
- 第１部：Faceup
  - テスト設計技法を使う前にある，テスト分析・設計
  - テスト分析設計方法の一例　～マインドマップを活用する～
  - さらにそのまえに考えておくべき事
- 第２部：Buildup
  - 見つめ直そうテストの意義
  - テスト設計技法をもう一度学ぼう
  - テストプロセス改善モデルや資格を利用しよう
  - ベンダーのトレーニングを活用しよう
  - コミュニティを活用しよう
  - 社内でのテスト技術者育成
  - テスト技術者の今後

© Akira Ikeda

Thank you! Hokkaido!

本講演が皆さんにとりまして
テスト技術を高める情報を
お伝えできていれば幸いです！

是非，一緒に
よりよくしていきましょう！

© Akira Ikeda

ご参考

講演では触れませんが，関連して
ご参考になりそうなことを共有します

テストに必要となる思考

「設計」と「テスト」，「チェック」と「テスト」を分けて考えてみよう

© Akira Ikeda

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

© Akira Ikeda

設計とは問題領域を狭めていく行為

© Akira Ikeda

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

© Akira Ikeda

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

© Akira Ikeda

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

© Akira Ikeda

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わかりやすい
- 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

© Akira Ikeda

マインドマップをコミュニケーションに活用する

マインドマップはコミュニケーションを促進するためにも利用できます

© Akira Ikeda

見える化の効果をコミュニケーションに生かす

- 自分で見る
- 相手の思考を見る（見せる）
- 全員の思考を見る（見せる），合わせる

© Akira Ikeda

一人で利用する場合

- 自分の思考を見える化することで…
  - トレースすることができる
  - セルフレビューが可能
  - あとで参照することができる
  - 自分の思考からさらなる発想を得られる

- 自分の発想を促進し，
  自分の発想を反芻する！

© Akira Ikeda

二人で使う場合（上司と部下の場合）

- 若手（特に新人）は自分の考えを説明できない
  - 自分の考えをまとめられない
  - 人への説明自体になれていない（補助輪が必要）
- 上司は若手の考えを理解出来ない
  - 脈絡のない説明
  - どこでつまづいているのか聞かないとわからない
  - 思考タイプが把握出来ていない

- 自分・相手の思考が見えることで…
  - 相手に自分の思考を順を追って説明できる
  - 相手の意図を理解しやすい

© Akira Ikeda

多人数で使う場合

- 多人数の会議ではなかなかまとまらない
  - 空中戦が多発
  - 質問が質問を呼び，元の話題に戻れない

- 全員の思考が見える，合わせることで…
  - 相手の発言の根拠や背景が理解できる
  - 思考の流れが見えるので脱線しにくい，元に戻りやすい
  - 完成したマインドマップは，思考の合意でもあるため，チームが意思レベルで合意しやすい

- ホワイトボードや模造紙を利用
  - 概ね4人くらいから
  - 一般的なブレーンストーミングの手法も適用する

© Akira Ikeda

マインドマップを適用する場合の課題と注意点

マインドマップは強力ですが万能ではありません

© Akira Ikeda

マインドマップを適用するにあたっての3つの限界

- 描きあがったマップのデータとしての再利用は難しい
  - デジカメやスキャナを使って，画像データとして取り扱う
  - ひとつの解決策として，PC用の描画ツールを使うという手もあるが…
    - データとしての再利用は容易だが，マインドマップ本来の効果は落ちる
      - マインドマップはイメージを多用することで，発散思考が刺激される
    - 発散は手描き，収束はツールというように目的によってどちらかを選択する

- 厳密なモデリングには対応しにくい
  - マインドマップは厳密な記法が定められているわけではないので，厳密なモデルを求める場合はNGTなど，他の記法を使う
    - マインドマップは，発散思考を使った発想促進と，ゆるやかな構造化
    - 他の記法のプリプロセスとして割り切っても良い

- 公式文書としては，理解が得られにくい
  - 思うほどマインドマップは普及していない
  - 受け入れられない状況であれば，マインドマップは中間成果物と位置づけ，内部資料とする
  - したがって，他のフォーマットへの転記作業が必要となる

© Akira Ikeda

テスト設計作業における注意点/ポイント

- マインドマップは思考の発散に重きをおいたツール
  - 発散と収束をひとつのマップで行ってもいいが，**発散のマップと収束のマップはわけて作成**することをオススメ
    - 収束のマップはNGTなど他の記法を使っても良い
    - マインドマップ→NGT，マインドマップ→FV表，などなど

- 思考を際限なく発散させない
  - 無秩序に発散させていくと巨大なマインドマップができあがる
  - **どこかで立ち止まって見直す**必要あり

- マインドマップは客観的には描かれない
  - 完成したマインドマップの内容は，他の人とは確実に異なる
  - 他の人のマインドマップを馬鹿にしない（人格否定に受け取られる）
    - **マインドマップに正解はない**

© Akira Ikeda

テスト設計作業における注意点/ポイント

- マインドマップを描くことを目的化しない
  - 綺麗な絵がかけても**気づきがなければ意味が無い**
  - 汚くても沢山の気付きやテスト観点が描かれていたほうがいいマップ

- 他の記法と連携させる
  - マインドマップの中に**他の図表を“絵”として埋め込む**
    - デシジョンテーブルや状態遷移図・表など
      - これらは絵とみなすことができる

- 保存方法やバージョンのつけ方を決めておく
  - あとで参照しやすいように**ファイリング**すること
    - 写真化して，アルバムソフトで管理しても良い
      - ただし，ある程度高解像度で保存しておくこと
  - 何枚か書き直す場合，バージョンをつけておく
    - さらに，**開発成果物と紐付けられる情報を付加しておく**
      - 日時，工程，担当者など

© Akira Ikeda

プロジェクトへ導入する際の注意点/ポイント（環境面）

- マインドマップを使うんだ！という雰囲気作り
  - マネージャやリーダ，ベテランこそ積極的に利用する
    - 上からの号令だけだと，若い技術者から反発を受け易い
      - 自らお手本を見せる気概
  - プロジェクトルームの装飾
    - 模造紙に書くなどしたマインドマップを壁に張り出しておく
      - プロジェクトの目標やメンバ構成など，描いておくと良いだろう
  - カラーペンや紙を準備
    - その気になればいつでも描けるように，道具を共有エリアに常備しておく
    - 書籍なども置いておくとなおよい

- プロジェクト周辺の理解の獲得
  - マインドマップは知らない人から見ると奇異に見える
    - 遊んでいると受け取られ，攻撃を受ける可能性がある
  - 部門内や同じフロアで作業している人たちへ根回ししておく

© Akira Ikeda

プロジェクトへ導入する際の注意点/ポイント（教育面）

- 公式トレーナによるトレーニング
  - 最低限何人かのキーマンは受けておいたほうが良い
    - 書籍ベースの学習では描き方は覚えられるが，頭の使い方までは覚えられない
    - 概念レベルまでしっかりと学習することが重要
  - リーダークラスの受講を特にオススメ
    - 立場上，一番他人のマインドマップを見て，場合によっては指導するため

- マインドマップを描く機会を意識的に増やす
  - スキルは使わないとさび付くし，成長もしない
  - 朝会や会議の資料のひとつとする
    - （例）日報の進捗や問題点を5分くらいでマインドマップにまとめ，報告させることで，一日最低一枚は描くことになる
      - このときメインブランチのいくつかは前もってテンプレート化しておいても良い

© Akira Ikeda

初級者向けのスキルアップモデルケース

MINDUPしたら，具体的に行動しよう！

© Akira Ikeda

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

© Akira Ikeda

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

© Akira Ikeda

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

© Akira Ikeda

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

© Akira Ikeda

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/

© Akira Ikeda

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/

© Akira Ikeda

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

© Akira Ikeda

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

© Akira Ikeda

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

© Akira Ikeda

その他，無償で参照できる資料

© Akira Ikeda

技術力up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

© Akira Ikeda