JaSST'18 Tohoku

HAYST法による テスト設計の考え方

因子・水準の抽出方法まで

2018年5月25日（金）

富士ゼロックス株式会社
SWI事業本部
秋山　浩一（@akiyama924）

≪SNS注意≫

全てOKです。
ただし、秋山がいましばらく。業界で生きていけるように、無意識で口走ってしまった社名のツイートなどには、ご配慮をいただけますと幸いです。

またご質問をメンションしていただければ後日回答します。

「おもしれー」、「最高です！」、「来てよかった！！」も大好物です。
無理強いはしません。

自己紹介：秋山浩一（今年はとてか、SS、SQiPシンポへ出没予定）

- 1985年4月　富士ゼロックス入社
  - HAYST法のコンサルティング業務に従事（テストの上流が中心）
  - D-Caseを用いた「機能安全」、「テスト戦略」等の合意形成
- NPO ソフトウェアテスト技術振興協会（ASTER）　理事
  - JaSST東京実行委員（2003～、今はアドバイザ）
    日本最大のテストシンポジウム1600名の動員
  - JSTQBステアリング委員（2006年～）
    テスト技術者資格認定を行う国際組織（ISTQB）の日本支部
- 日科技連　SW品質管理（SQiP）研究会　委員長（2011～2013年）
  - テスト分科会主査・副主査・アドバイザ（2008～）
- ISO/IEC JTC 1/SC7 WG26委員（2009年～）
  ソフトウェアテストの国際標準「ISO/IEC/IEEE 29119」の策定
- 共著書『ソフトウェアテストHAYST法入門』（2007年）
  日経品質管理文献賞受賞
  著書『ソフトウェアテスト技法ドリル』（2010年）
  『事例とツールで学ぶHAYST法』（2014年）※いずれも日科技連出版社
- 博士（工学）：信頼性情報システム工学専攻、日本品質管理学会代議員

FUJI Xerox

提言：

わたしがソフトウェアテストの仕事を始めてから20数年経ちました。私の活動は『プリンターのテストに毎年“数トン”の紙を使って大変なことになっている』という問題から始まりました。その問題に対して、プリント操作と出力結果確認を自動化するツールを作り、紙とテスト工数の削減を実現しました。

しばらくすると、別の問題に悩まされました。工数等ではなく『プロッターへの出力をプリントしたときに引出線が消える』といった問題です。この問題は、CADアプリが引出線に対して“太さ指定をゼロ”としていたことが原因でした。プリンターは、太さがゼロなので何も描きませんでした。しかし、“太さ指定がゼロの線”とは“（かすれてもよいので）描画可能な最も細い線”という意味だったのです。

“太さ指定がゼロのときには最も細い線で描く”という仕様は、納得できるものです。もしもこの指示をプリンターではなく人間が受けていたら違和感を覚えて仕様を確認したことでしょう。しかし、プリンターは愚直に太さをゼロとしてしまいました。この体験から、実装の問題に加えて仕様の問題も発生しないようにする必要があると感じました。

上記は昔話ですが、近年、IoTで様々な機器がインターネットにつながってそれらが組み合わさって一つのサービスを提供する世界が始まっています。例えば、『帰宅時に車が家に近づいてきたら自動的に部屋のエアコンのスイッチが入る』といったサービスです。このような**『つながる世界』を実現するためには、様々な機器間の相互運用についてテストする必要があります**。20年前にプリンターで起こったような問題が発生する可能性があるからです。

さて、品質向上が売上、利益に直結していると実感していた昔は、黙っていても品質の重要性をみなが認識していました。しかし、現在は、品質を軽視しているわけではないものの、『形式的にハンコをもらうことが品質保証活動』という風潮があるように思います。IoTやAI、さらにIoTを人のつながりまで拡張したIoH(Internet of Human)といった新しいテクノロジーによって新しい問題が生まれることが懸念される今、品質そのものについて、もう一度深く議論する必要がありそうです。

「叶えたい」より「か・ナーえる」 私なんだ!
100%(ヒャクパー)負けない 止まらない
よくバリバリな夢
フレフレ☆ススメ　♪

※　JSQCニューズ 2018年3月号より

http://www.jsqc.org/ja/kankoubutsu/news/articles/2018_03/index.html#c2

本日のアジェンダ

◆ 組合せテストについて

- 組合せテストってどんなの？
- 組合せテストを支える3つの技術

◆ HAYST法が目指していること（特徴）

- HAYST法が取り扱うもの、取り扱わないもの
- HAYST法の考え方

◆ HAYST法のステップの概要
（詳細はワークで理解します。ここでは概要説明とサンプルの解説をします）

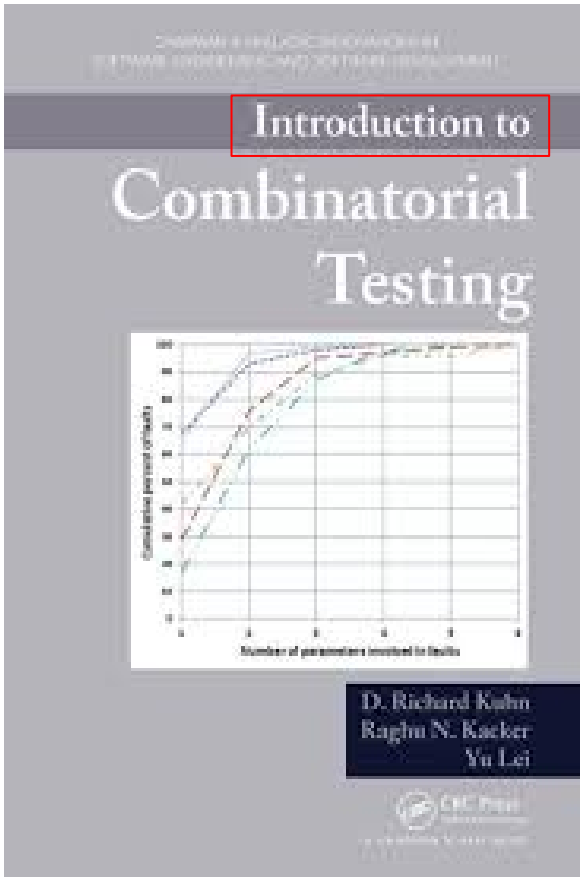
① 6W2H
② 3W～ユーザーストーリー～FV表
③ ラルフチャート
④ FL表

◆ 事例紹介

◆ 質疑（事前アンケートでいただいたものを中心に）

（知識ゼロからの）
組合せテスト

- 組合せテストってどんなの？
- 組合せテストを支える3つの技術



単行本: 341ページ
　でも“Introduction to”＝“入門”
出版社: Chapman (2013/7/31)
著者: David Richard Kuhn
　（IEEE Fellow）

組合せテストってどんなの？：　3ステップ ①→②→③

①　組み合わせる条件を決定する

例）ブラウザ｛IE，　Google Chrome｝
　　OS｛Windows，　iOS｝
　　端末｛デスクトップ，　モバイル，　タブレット｝
　　テスト対象アプリのバージョン｛2.0，　3.0｝

因子：ブラウザ、OS、端末、バージョン

水準：IE、Chrome等

②　組合せ表を作成してテストを実施する

No.	ブラウザ	OS	端末	バージョン	結果
1	IE	Windows	デスクトップ	2.0	pass
2	Chrome	iOS	デスクトップ	3.0	pass
3	IE	Windows	モバイル	3.0	pass
4	Chrome	iOS	モバイル	2.0	pass
5	IE	iOS	タブレット	2.0	fail
6	Chrome	Windows	タブレット	3.0	pass
7	IE	iOS	モバイル	3.0	pass
8	Chrome	Windows	モバイル	2.0	pass

③　テスト結果を分析する

5番目でバグを見つけた！
テスト原因分析した結果、
　A) IE×タブレット
　B) iOS×タブレット
　C) タブレット×2.0
のどれか！
（IE×iOSではあり得ない）

※　因子が増えると分析が大変になるから手法が必要

　上記A，B，Cの3個のテストを追加実施する？？

組合せテストってどんなの？：　補足

有則のテストも
もちろん大切！
9/12のSQiPシンポのHTでやる！？

◆ 組合せテストには大きく2種類ある

- 仕様書に書いている組合せのテスト（有則）
  - 例：　仕様：宿泊キャンセル料金：　当日：100%,　前日：50%,　一週間前：20%
    - ★　上記仕様なのに前々日にキャンセルしたら50%キャンセル料が取られた！
    - ※　開発者談：　「開発ミスとユニットテスト漏れです。すみません。」
  - 原因：　ロジックの実装ミスが現れる等
  - 対応：　デシジョンテーブルを作ってテストを行う
  - 技法：　クラシフィケーションツリー、デシジョンテーブル、原因結果グラフ、CFD法

本日はこちら

- 仕様書に書いていない組合せのテスト（無則）
  - 例：　仕様：用紙サイズと用紙の厚さの組合せ仕様はない（自由に組み合わせることが可能）
    - ★　A3サイズで薄紙時のみプリントに10分かかる
    - ★　「A3×普通紙」、「B4×薄紙」では問題なし
    - ※　開発者談：　「サイズと厚さに関係性があったとは！？　想定外でした。」
  - 原因：　仕様化漏れ（別の関数の仕様変更情報が伝わっていない）、もしくは、状態変数汚染
  - 対応：　全組合せは無理なので、2機能間、3機能間をテストする
  - 技法：　直交表、HAYST法、オールペア

◆ 組合せテストを開始するための３つの前提条件

1. 仕様書がある（無ければ期待結果がわからないし、単機能の問題が出てテストが進まないので、組合せテストを始めるべきでない。　→　仕様書を書くほうが良い）
2. 単機能（有則の組合せを含む）がテスト済み
   （無則の組合せテストができないわけではないが、回帰テストが増え、テスト効率が悪い）
3. 工数は、単機能テストと同量を見積り、用意している

FUJI Xerox

組合せテストを支える3つの技術の広がり

① 組合せ表を作成する技術（1990年頃～）

- 直交表や、オールペア（ペアワイズ）など組合せ表を作る技術（**テスト効率**）
- 禁則（制約）を正しく、かつ、メンテナンスしやすく入力・更新する技術

② 組み合わせる条件を見つける技術（2000年頃～ ）

- 因子｛水準1， 水準2， 水準3， ……｝を見つける技術（**品質を決める**）

例）ブラウザ｛IE， Google Chrome， Safari｝
OS｛Windows， iOS｝
端末｛デスクトップ， モバイル， タブレット｝

本日は②を
マスターします

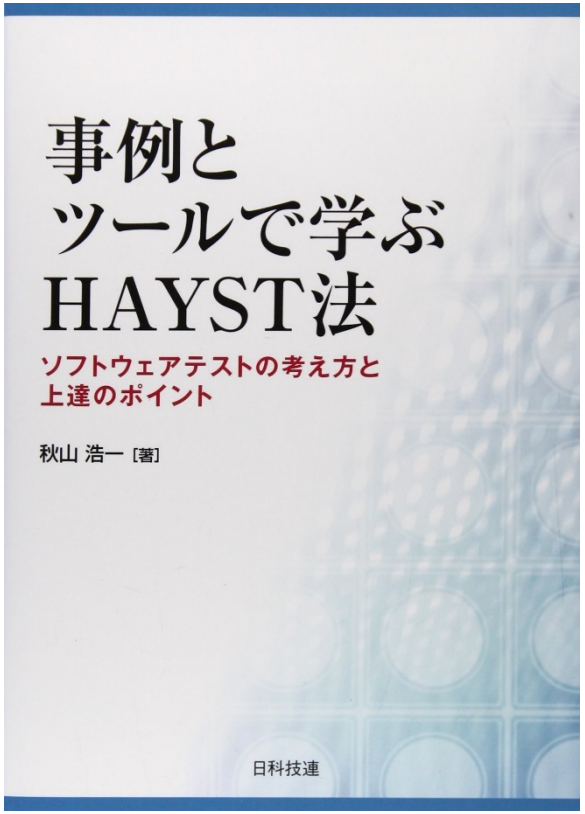
※ 因子・水準のことをパラメータ・バリュー、あるいは、P・Vともいう

③ テストした結果を分析する技術（2010年頃～）

- 因子がk個あると原因となる組合せは2つの因子の組合せとしても${}_kC_2$
  （**デバッグ効率を向上する技術**）
  例）因子が10個なら${}_{10}C_2$＝45個の組合せ
  因子が20個なら${}_{20}C_2$＝190個の組合せ
  因子が100個なら${}_{100}C_2$＝4950個の組合せ
- 再現条件を見つける技術、追加テストを減らす技術

HAYST法の特徴

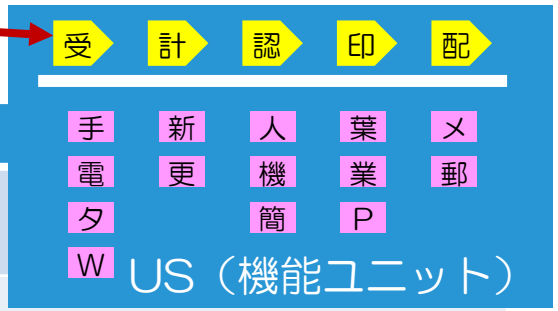
HAYST法が取り扱うもの、取り扱わないもの
HAYST法の考え方



単行本: 197ページ
出版社: 日科技連出版社（2014/7/1）

HAYST法が取り扱うもの（青表）、取り扱わないもの（赤表）

ナラティブフロー（黄色）
例）　受付→計算→確認→印刷→配送

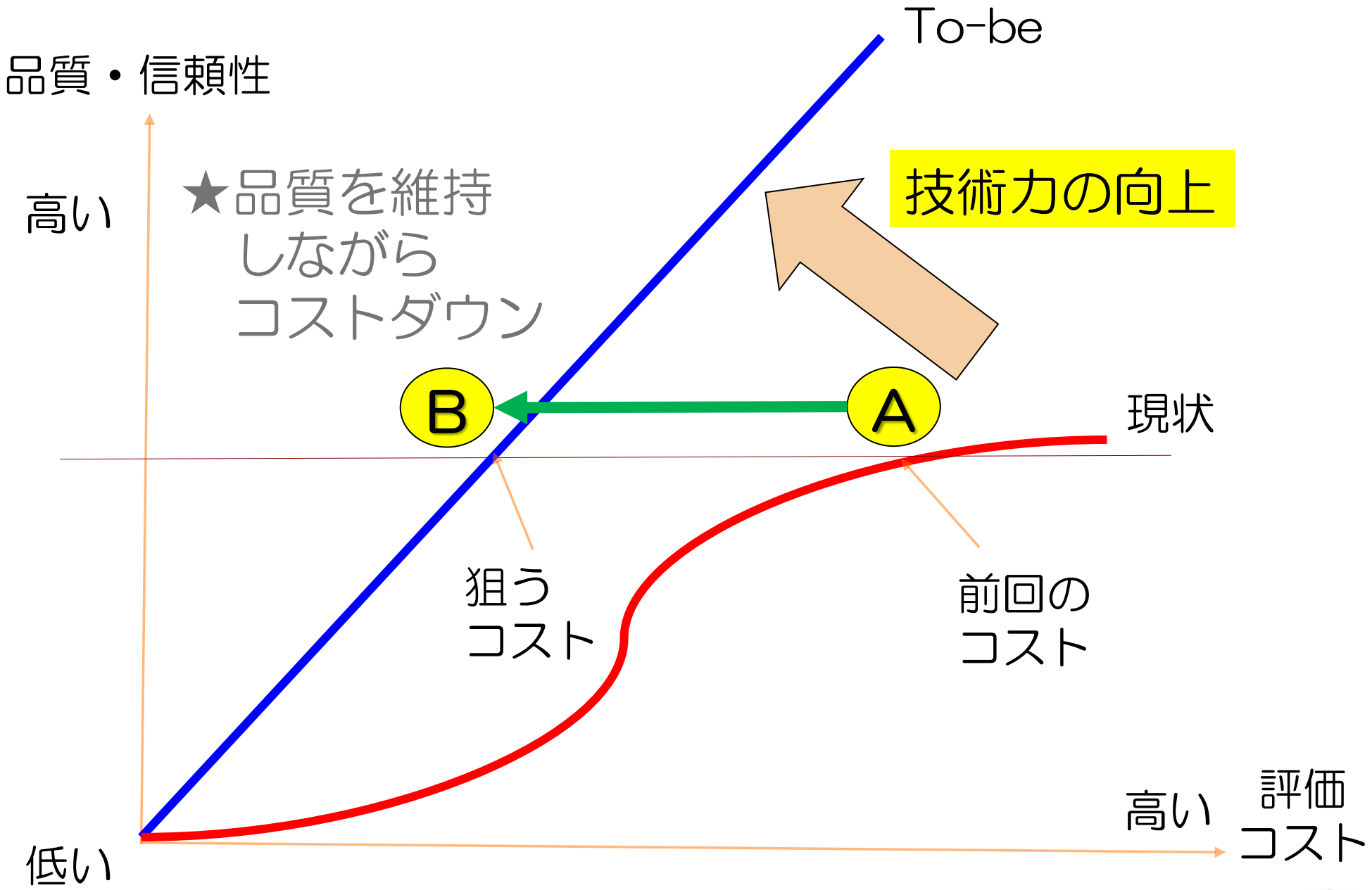


No.	項目	理由（ポイント）
1	アジャイル	1. ユーザーストーリー（US）ごとに組合せテスト 2. ナラティブフローでUSを組み合わせたテスト
2	仕様書 ※　箇条書き部分は望ましい仕様書	• 仕様書が無ければテストしている場合ではない ✓ テスト設計時に考慮する範囲を限定（定義）できる ✓ 仕様書が常に最新情報に更新されていて信用できる ✓ 現行の仕様について、その是非（要求を果たすこと）を説明できる
3	ユニットテスト	• プログラマがプログラミング中にテストを行う（一番大切で数も多い）
4	運用と保守	• 売った後のことを考える（ライフエンド時でも目的を果たすことをテスト）
5	非機能要件	• 機能を目的単位で取り扱うことで非機能のテストを同時に行う（FV表）

No.	項目	理由（ポイント）
1	商品企画	• ビジネスの分析は取り扱わない（企画品質はコンジョイント分析で評価）
2	要求	• ビジネス要求と業務要求は取り扱わない • システム要求は【仕様書】と同じもの（曖昧か、具体的か、の違いのみ） • 「みんな持ってる！」、「みんなって誰よ？」←要求はあてにならない
3	ペルソナ	• 利用者を高々十数名想定しても、広大な市場条件はカバーできない
4	内部構造（設計）	• 効率化のため6W2Hでアーキテクチャ、FL表で境界値はコードで確認するが、それ以外はプログラマの自由裁量（いつ変えても対応できるテストとする）
5	受け入れテスト	• 「業務に使えるか」を確認する作業であり業務シナリオテストを行う

FUJI Xerox

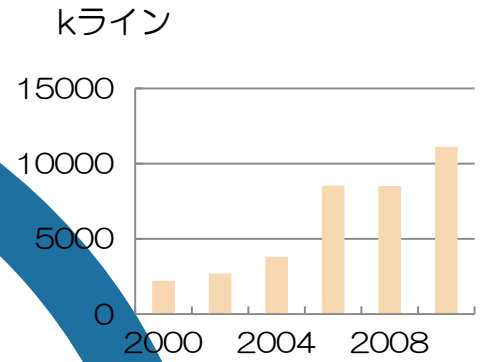
経営者のテストへの期待



FUJI Xerox

ソフトウェア開発/テストを妨げる五悪→HAYST法による対策

① 大規模・複雑化
コード、テストとも
→6W2Hで全体を把握
→FV表で小さくしてから直交表で組合せテスト



② （外）変化を予測しない・想定できない

→ 3W+直交表で保証範囲のキワをねらう

③（内）ソースコード読まない・読めない

→ ラルフチャートでテストの情報をモデル化

④ テスト戦略なし、テストアーキテクチャなし

→ D-Caseでテスト戦略/全体構造の合意を取る

⑤ 個人/組織ノウハウを蓄積・活用しない

→ FL表でノウハウを蓄積する（High / Low）

因果ループ図

①→（②③）→④→⑤

原因→結果

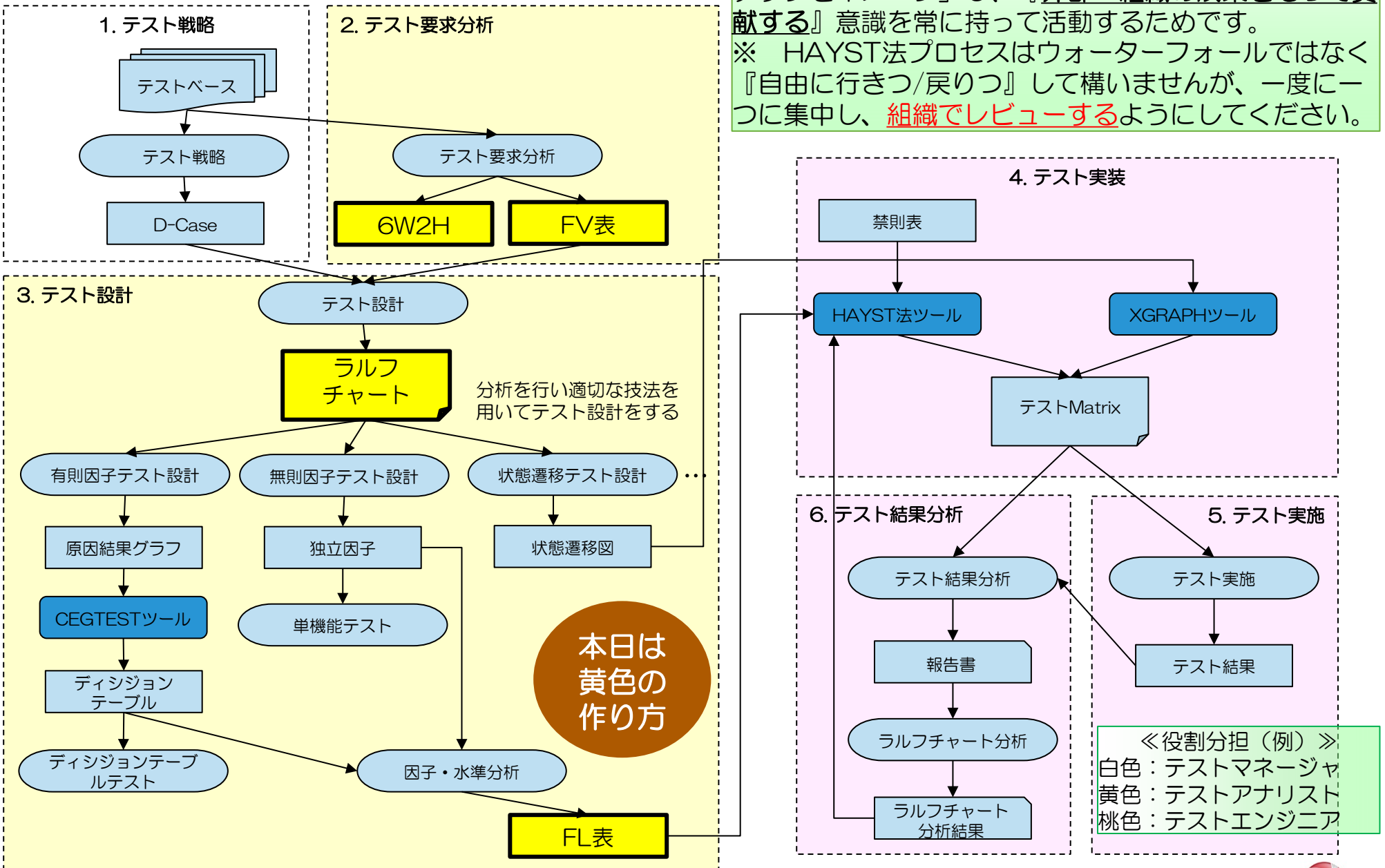
FUJI Xerox

HAYST法のステップ

詳細はワークでHAYST法の考え方

HAYST法プロセス

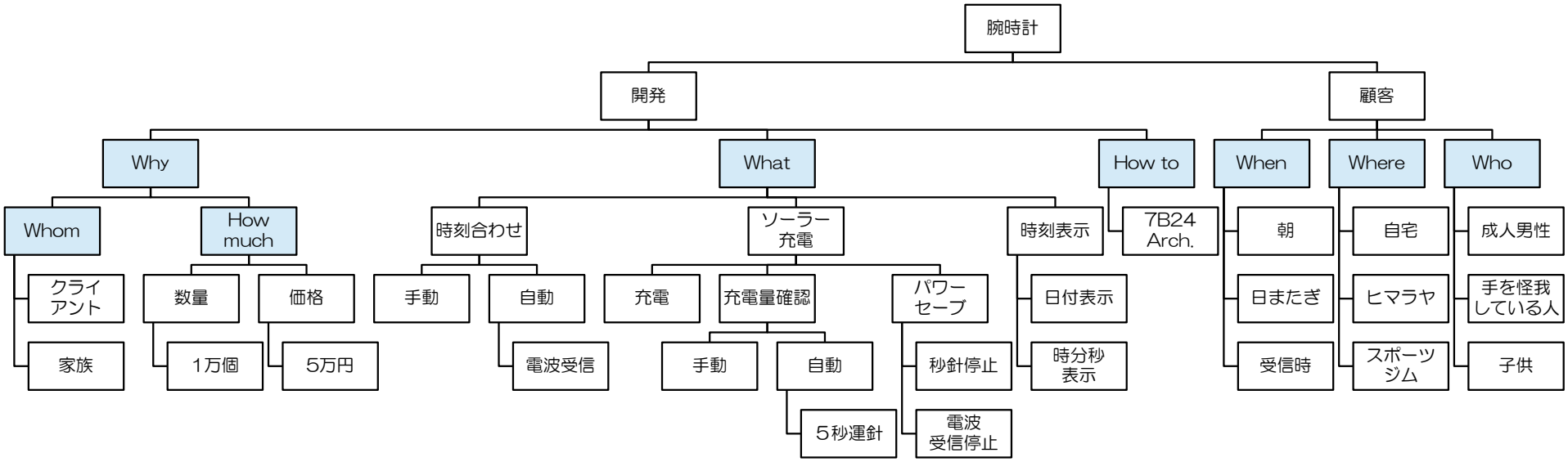
プロセスを作る目的は「未来の抽象的な目的」（十分な品質を提供する等）を「現在の具体的な目標と行動」に変換していくために「目的を分解・具体化」し、「ステップをイメージ」し、『外部へ組織の成果をもって貢献する』意識を常に持って活動するためです。
※　HAYST法プロセスはウォーターフォールではなく『自由に行きつ/戻りつ』して構いませんが、一度に一つに集中し、組織でレビューするようにしてください。



6W2H

- 目的
  - テスト対象の理解
  - テストに対する要求の理解
- 方法
  - 3つの視座で考えられる人が作る　　（誰が知っているか？）
    （3つの視座：　開発者、ユーザー、お客様のお客様）
  - それぞれの視野
    - 開発者｛Why（要求）, What（仕様）, How to（設計）｝
    - ユーザー｛When（いつ）, Where（どこで）, Who（だれが）｝
    - お客様のお客様｛Whom（誰のために）, How much（販売数×単価＝売上）｝
  - 樹構造で分解していく
- どこまでやるか
  - 因子レベル（気にしない）
  - 最長で半日
  - レビューは必須

6W2H



Whom枝はお客様のお客様

※　価値を見つける手がかりとなる

How much枝から売上→品質コスト→評価コストを求める

例）
売上：1万×5万＝5億円
4%は2千万
その半分でテスト

What枝は仕様書を書き写す

① 使う色は黒1色
② MECEについては気にしない
③ とにかく素早く描く
④ ひと通り書いてから気になる点を追加する
※ 手書きでもツールでもよい

How to枝はアーキテクチャなどを書きテストの削減に使う
→グレイボックステストで使う（本当は使いたくないけど）

各３W枝には、

標準１つ
異常（キワ）２つ

を書く。
異常と言ってもライフエンド時に予測される保証範囲のギリギリの【キワ】

3W ～ ユーザーストーリー ～ FV表

目的

- テスト対象を分解してFV表を作る
  （大きいテストは作れないので小さくする）

方法

- 3W： ユーザーのコンテキストをWhen×Where×Whoで作る
  - 勝手に考えたコンテキストで上手くいっても市場環境は複雑
  - 極端な条件で上手くいけば、ゆるい条件なら当然上手くいく
  - 直交表を活用してコンテキストを作る
- ユーザーストーリー：機能ユニット
  - 動きではなく、価値を持った機能のまとまり
- FV表のF：目的機能（なんのためにある機能かとことん考える）
  - 難しい場合はユーザーストーリーで代替する

どこまでやるか

- 仕様書の全て（≒6W2HツリーのWhat枝の全てのノードをV列に挙げた因子で消し込む）

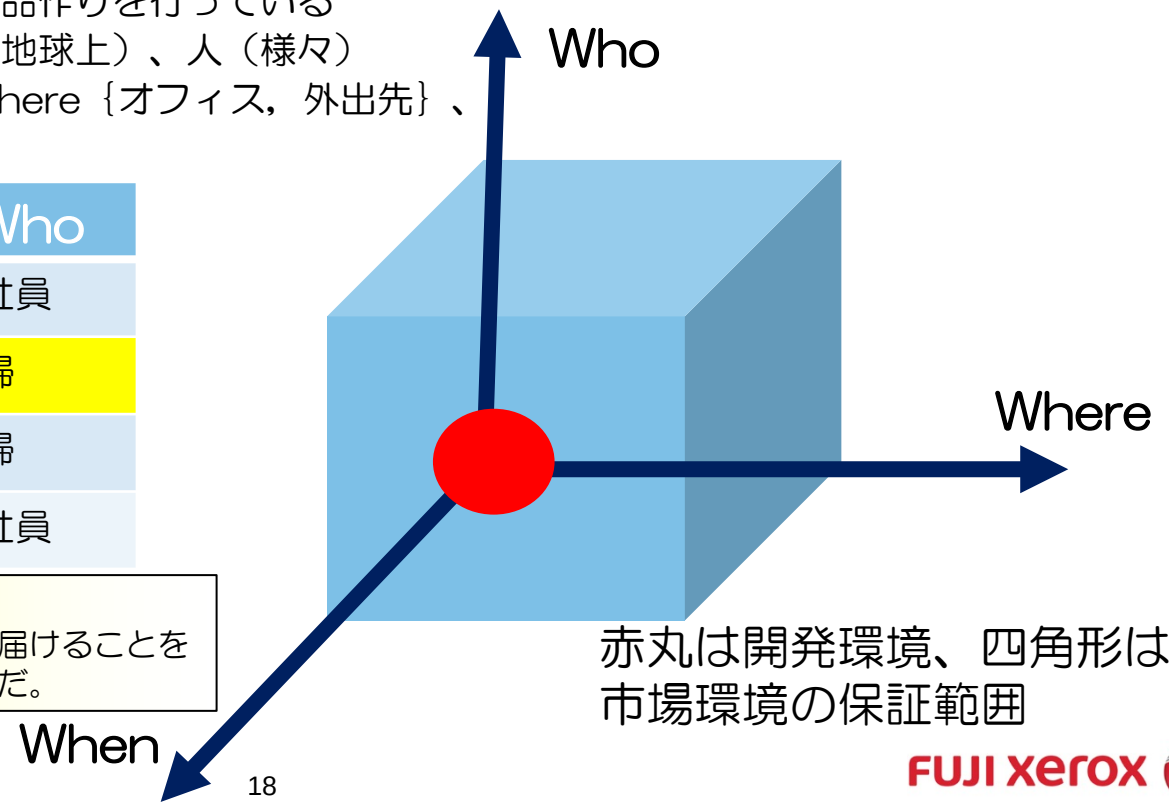
3W（製品の利用コンテキスト）

When	Where	Who
1	1	1
1	2	2
2	1	2
2	2	1

- テスト対象のライフを考える
  - ✓ コピー機はレンタル切り替え時が勝負
    - ライフが切れるときまでずっと喜ばれていることをテスト
    - 顧客＝環境条件と利用者の特定（When：いつ、Where：どこで、Who：誰が）
      - どこまで考えるかが大切（赤丸は開発環境、立方体は市場環境）
      - 例）　富士ゼロックスのSWIでは、例えば、世界中の人がIssueを手書きで書き、それを複合機のボタン一つでクラウド上のサーバーに置き、OCR・翻訳・オントロジー技術で解析し、どのようなテスト状況かわかるようになる市場を想定した製品作りを行っている
        → 時間（いつでも）、場所（地球上）、人（様々）
        → When｛昼間，深夜｝、Where｛オフィス，外出先｝、Who｛会社員、主婦｝

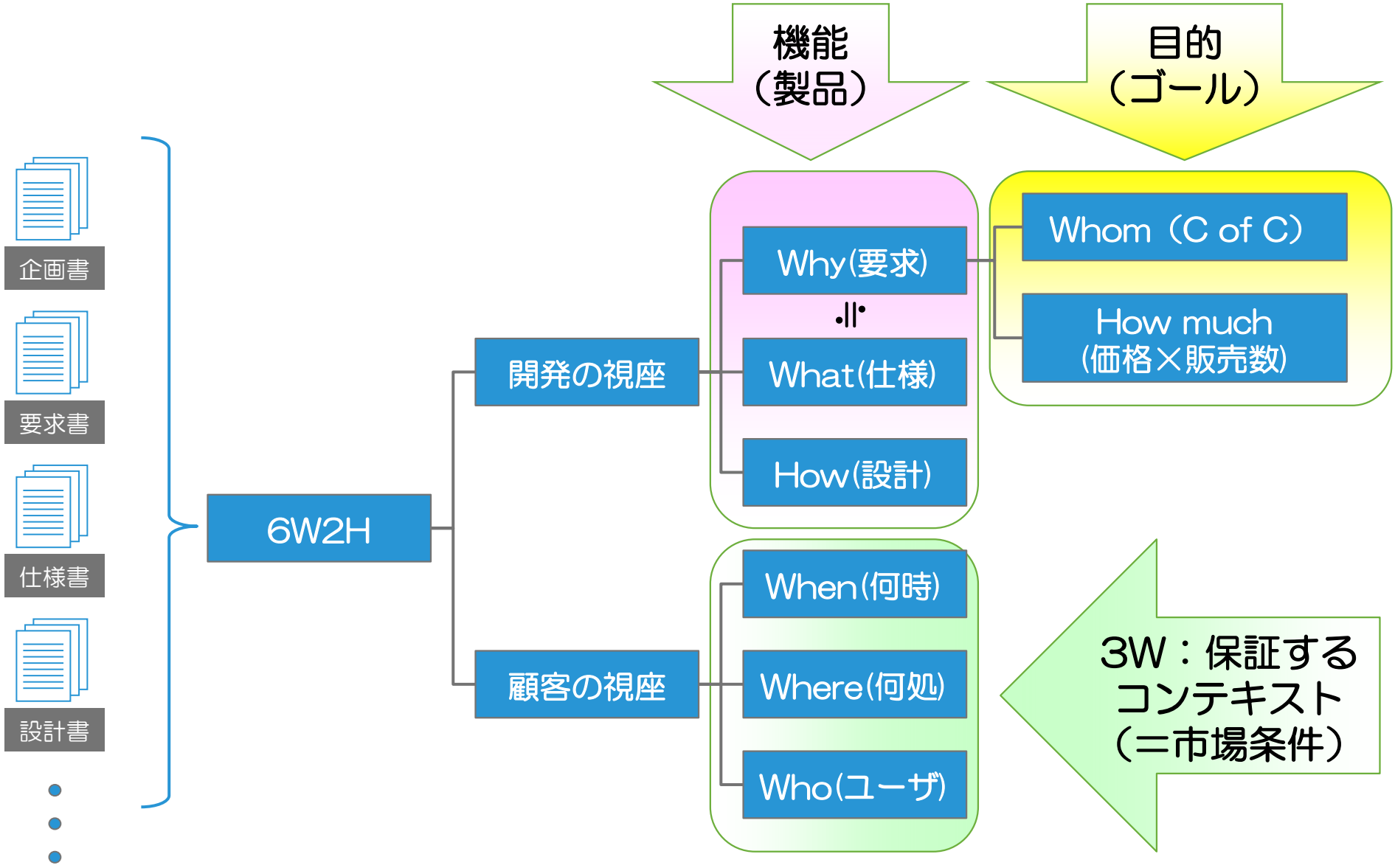
When	Where	Who
昼間	オフィス	会社員
昼間	外出先	主婦
深夜	オフィス	主婦
深夜	外出先	会社員

ユーザーストーリーの例
主婦の役割として、会社の重要書類を夫に届けることを達成したい。家に置き忘れて出社したからだ。



赤丸は開発環境、四角形は市場環境の保証範囲

6W2HからFV表へ



FUJI Xerox

ユーザーの視野（3W）からコンテキストを明らかにして、ユーザーストーリーを作成するヒント

- 6W2Hのなかの3Wからユーザーストーリーを作成する（●は考えるヒント）
  - ✓ When
    - ● 周期（年・季節（四半期）・月・週・日・時・分・秒）
    - ● 順番（インストール直後、操作順番）
    - ● タイミング（充電切れ、動き始め、ネット切断＆再接続）
  - ✓ Where
    - ● 天候環境（雨、風、気温、気圧）
    - ● 国（日本、米国、中国、航海中、国際線の中、宇宙）
    - ● 抽象的な場所（移動中、狭い・広い、劣悪環境、屋内、屋外）
  - ✓ Who
    - ● 性別・年齢・知見の有無（男、女、子供、大人、老人、初心者、熟練者）
    - ● 身体障害の有無（色覚、聴覚、手足、脳）
    - ● コンピュータ（AI、遠方からVR、他のシステム、同時使用）
    - ※　テスト対象にとって意地悪な条件となるキワを考える

FV表（大きいテストは作れないので小さくする）

■ FV表で大きなテスト対象を適切に分割する

No.	Fr	V	T
1.3	目的機能1	検証条件1	方法1
2.1、3.2	目的機能2	検証条件2	方法2
1.1、2.2	目的機能3	検証条件3	方法3
Covering	Goal	Question	Metric

- ✓ テスト対象にある機能を目的の粒度でまとめる
  （アジャイル界隈でいうユーザーストーリーでもよい）
  - V列を書きながら6W2HのWhatのノードを消しこむ
  - Whatの枝にある全てのノードが消し込めたら完了
- ✓ FV表の作成方法
  - ① 仕様書をアプリケーションを目的機能で分割しFr列に目的機能を文章で記述する
  - ② No. 列には該当機能が記載されている箇所を記載する←仕様書の網羅を担保するため
  - ③ 目的機能を満たす条件（因子）をV列に記載する（因子はゴールを満たす条件）
    ※　因子はラルフチャートで出し尽くすのでここでは重要なもののみでよい
  - ④ V列に記載した条件を確認する方法（テスト技法、テストタイプなど）をT列に記載する
    確認する方法　＝　テストによるエビデンスの取得方法

■ 何故FV表が因子の発見に効果的なのか

- ✓ 「ゴールを果たす」は曖昧なので、「これこれの条件（因子）で目的を達する働きをする」に言い換える
- ✓ ユーザー要求を超えてシステムが果たすべき目的にまで考えが及ぶことでブレイクスルーができる（要求と仕様は曖昧か具体的かが違うだけで、同じもの。どちらも過去を起点とする）

目的機能とユーザーストーリー（US）

「質」について考えるためには、何を（製品）誰に（顧客）提供するのかを明らかにし、その上で、顧客の視点・価値観でのその製品に対する評価こそが「質」と考えるのがよいと思います。（飯塚悦功）

- 仕様書の「機能」で整理することの問題点
  - 「正しく動作」しているかの確認しか出来ない
    - 仕様書には「機能」の動きしか書いていない
      - コンテキスト（使用の文脈）の理解が重要
      - 仕様書に暗黙の仕様（前バージョン仕様、開発者の常識）は書かれない
    - 本来は「正しい動作」をしているかの確認がテストの目的
  - 要素還元的な見方しか出来ない
    - 分解した小さな機能が全て動けば全体が問題なく動くと思ってしまう
- 「機能」から導き出した「目的機能」で整理する意義
  - 必ずユーザの使用目的や市場条件を考えることになる（動的）
    - US：＜利用者の役割＞として＜ゴールｘｘ＞を達成したい［＜理由＞のためだ］
  - 非機能要件のテストが楽になる
    - 目的機能単位に非機能要件のメトリクスを計測する
    - 性能、信頼性、セキュリティ要件は目的ごとに異なる（カプセル化）

補足）ユーザーストーリーとは

ユーザーストーリーとは

フィーチャーや機能の価値をユーザーに伝え、会話を引き出すためにKent Beckによって、2000年にその概念が作られた

ユーザーストーリーのテンプレート

Mike Cohn：2004年

＜ユーザーの役割＞として、＜ゴール＞を達成したい。[それは、＜理由＞のためだ。]

- ユーザーの役割：ゴールの達成により恩恵を受けるユーザーの役割
- ゴール：ユーザーストーリーのゴールであるフィーチャーや機能
- 理由：ゴールを達成した時にユーザーが得られる利益や利点（自明であれば書かなくても良い）

ユーザーストーリーの評価軸（INVEST）

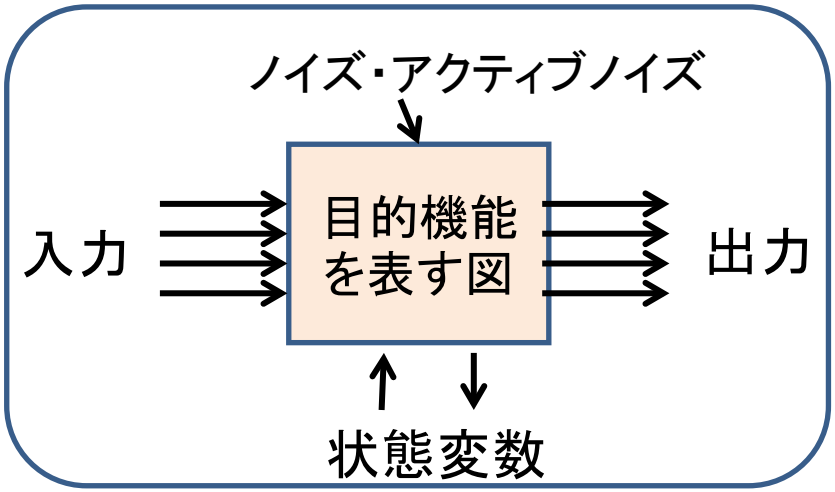
独立していること（Independent）、交渉可能であること（Negotiable）、価値があること（Valuable）、見積もり可能であること（Estimable）、小さいこと、もしくは、適切なサイズであること（Small、もしくは、Sized appropriately）、テスト可能であること（Testable）

- ユーザーに直接接点を持つ目的機能はユーザーストーリーで表現できる
- ログや保守の機能などはユーザーストーリーでは表しにくい
  （そのような場合は、直接、目的機能を書く）
- INVは「目的が明確で要求獲得・分析ができていること」
- ESTは「要求の仕様化ができて、機能仕様が固まっていること」

ラルフチャートによるテスト設計

■ 複雑な目的機能から因子を探す

- ✓ 品質工学・タグチメソッドのKH
  （P-チャート、システムチャートともいう）
- ✓ 因子が見つかる
  - 信号因子：入力
  - 誤差因子：ノイズ
  - 状態因子：状態変数
- ◆ 例えば、IE11
  - ✓ 信号因子：画面上のメニューなどのアイテム
  - ✓ 誤差因子：負荷やクラッキング
  - ✓ 状態因子：DBのレコード



■ 何故ラルフチャートが効くのか

- ✓ システムの最適なモデル
- ✓ ラルフチャートを複数結合することによって状態遷移や変数共有等ねらいどころのウィークポイントが見つかる

ラルフチャートとテスト技法

入力	状態変数	ノイズ	アクティブノイズ	出力	テスト技法
少ない	–	–	–	少ない	同値分割・境界値分析
論理組合せ（有則）	–	–	–	正常系 異常系	原因結果グラフ CFD法・DT
組合せ（無則）	Read中心	市場環境の考慮	いたずら	正常系	HAYST法・直交表 Pairwise
–	Read/Write	–	–	–	状態遷移
論理組合せ	Read/Write	–	–	–	機能図式
–	Read/Write	市場環境	–	例外系	シナリオテスト
–	–	極限環境	–	多大	負荷テスト
–	–	–	犯罪	–	セキュリティテスト
–	–	ターゲット市場	–	振舞い	非機能テスト
異常値	–	–	–	エラー	エラー処理テスト
複数のラルフチャート					並行・並列テスト技法

FL表の作成

FL表による因子・水準の確定

FV表にて、HAYST法などの組み合わせテスト技法を使用すると決めた項目について、入力に当たる因子・水準を整理するもの。水準１はデフォルト値とし、異常値は入れない。

因子名	水準1	水準2	…	水準k	…	水準n	…

FL表の例(メールソフト)

因子名	水準1	水準2	水準3	水準4	水準5
プロトコル	POP	APOP	IMAP	WebMail	
セキュリティ	なし	SSL	TSL		
メールデータ	JIS	UTF-8	HTML	英語添付	日本語添付
送信元のソフト	Outlook	Thunderbird	Becky!		
OS	WindowsXP	Vista	Windows7		

補足）ノウハウを蓄えるFL表

FL表によってノウハウを蓄える

ハイレベルな水準も残すことによって、「なぜその水準を選択したのか」の意図が残るようにする。合わせて管理することで、テスト資産となる。

FL表の例(メールソフト)

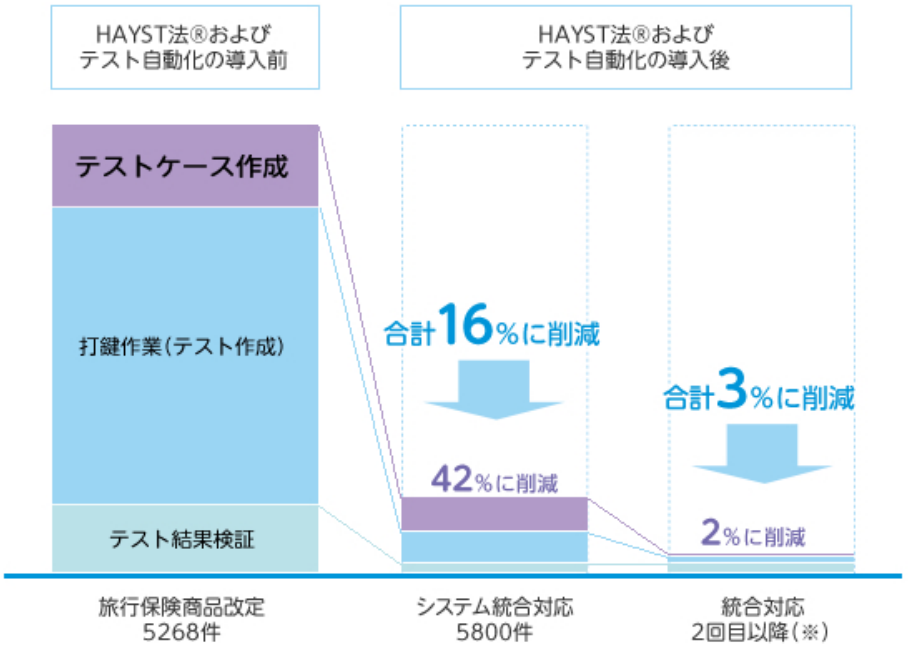
因子名	水準1	水準2	水準3	水準4	水準5
意図（High）	標準	パスワード暗号化	標準2	標準3	
プロトコル	POP	APOP	IMAP	WebMail	
意図（High）	多くのケース	まだ使用あり	今はこちら		
セキュリティ	なし	SSL	TSL		
意図（High）	標準	漢字・合字	形式	添付ファイル	ファイル名
メールデータ	JIS	UTF-8	HTML	英語添付	日本語添付
意図（High）	利用者多い	家庭で使用	プロトコル厳格		
送信元のソフト	Outlook	Thunderbird	Mew		
意図（High）	最新	癖がある	利用者多い		
OS	Windows 10	Vista	Windows7		

事例紹介

目指すところ

社外コンサルティング実績（2004年～2017年：秋山）

業種	数
電気・電子（重電・家電など）	13社
製造（自動車・精密機器など）	11社
金融・保険	10社
SIer	5社
医療機器製造	3社
通信インフラ	2社
合計	44社



導入事例：http://www.fujixerox.co.jp/solution/jirei/sompo01.html

SOMPOシステムズ株式会社様（2009年から導入：右上図）

事前アンケート　Q&A

本日の参加者の申し込み時のアンケートにあったご質問に回答します。

ご質問は2行以下に要約しています。質問の意図が違う場合は別途ご質問ください。

事前アンケート　Q&A

ご質問	回答
機械学習やAIを使ったシステムでは学習する毎に出力値が変わるため、期待結果が一点にならない。	機械学習した状況を可視化する機構を追加しましょう。そして、状況に対する期待結果の対応を作るようにします。
1人しかがいない現場でも有益につかえそうですか？	はい。使えます。そういうケースでスタートするほうが多いです。
Web系でも取り入れるメリットはあるのでしょうか？〃	HAYST法は、ジャンルを選びません。
複合機の品質保証業務で大変だったこと、失敗した事例などがあれば知りたい。	「提言」のスライドをご参照ください。あと、SWIのほうも。
効果的効率的に探索的テストを行いたい。	FV表を作って、その行をチャータにしたら良かったです。
設計時の詳細仕様書の書かれていない部分をどうやって見分けてテストに持っていけるか。	USDM形式で仕様をまとめ、あとはテスト設計者自身も開発方法の勉強をするしかないと思います。（私は開発経験有でした）
建設業の積算ソフトの受け入れテストを行っているが、HAYST法を取り入れるメリット。	受け入れテストはHAYST法対象外です。
目的機能とは何なのか。要求とは異なるものなのか。 あと目的機能の導出方法。	今日のワークでマスターしましょう。座額だけでは限界があるため、実業務に適用した後に本日の内容を復習をすると良いです。
（組み込み系でのHAYST法の活用事例やメリット、）HAYST法を適用するに当たり課題となる部分があれば教えてほしい。	「やるぞ」という強い思いのあるリーダー（役職と言う意味ではなく）がいることが大切です。最初は小さくても構いません。
「こんなHAYST法は嫌だ。」どんなHAYST法？	「目的機能」に「使用場面と機能」を書くHAYST法。
評価対象物によって、HAYST法の向き/不向きはあるの？ ソフトだけでなく、ハードウェア評価への応用への留意点とは？	ルールベースを使った自動生成コードには向きませんでした。 ハード＋ソフトで評価を実施します。HWはシミュレータ命。
開発サイクルが短い状況で、HAYST法を取り入れたいです。 指針がございましたらご教授ください。	各ILでUS単位でHAYST法を適用し、リリース直前に「それから」でつなぐナラティブフローでUSを水準として適用します。
テストを行う時間に制限をかけないと際限なくテストを行ってしまう。テストの止め時みたいなのを教えて欲しい。	一般には納期を前提に最善のテストを作りますので恵まれた悩みのように思いました。月並みですが、テスト結果の予測を立てると共に、テスト結果分析をテスト中に行うことが効果的です。

HAYST法プロセスと手法（まとめの表）

HAYST法のスコープ

標準テストプロセス	手法（How）	目的（What）	理由（Why）
テスト戦略（テスト計画）	D-Case	テスト戦略の合意	テスト全体の戦略の合意を取るため ● テストの前提条件とゴールを明確にする ● 残すエビデンスを決定する
テスト要求分析	6W2H	テスト対象の理解	顧客視点：テスト対象が果たすべきことを知る ● 製品仕様：何を提供するのかを理解する(What) ● 市場条件：誰に提供するのかを理解する(3W)
本日	FV表	テスト対象の分割	大きなテスト対象を小さく分割する ● ユーザーストーリーを作る ● 機能が果たすべき目的を考えて分割する ● その目的機能の検証条件を明らかにする
テストアーキテクチャ設計	ラルフチャート	分割したテスト対象のテスト条件（因子）の発見	分割後のテスト対象をモデリングする ● 入力・出力・状態の因子 ● ノイズ・アクティブノイズの因子
テスト設計	FL表	テストする因子・水準の確定	テストすべき因子・水準を決める ● 抽象度の高い水準：テスト意図の伝承 ● 具体度の高い水準：テストを正しく実施
テスト実装	HAYST法ツール XGRAPHツール	組合せテストマトリクスの作成	直交表に因子・水準を割り付ける ● 高い網羅率 ● 少ないテスト件数
テスト実施	データ駆動自動化 キーワード駆動自動化	テスト実施工数の削減	テストを実行し不具合を検出する ● 自動実行でテスト実施工数の削減 ● テスト結果分析で不具合原因の解明

FUJI Xerox

Appendix

良く聞かれること

HAYST法とオールペアとの違い

No.	比較ポイント	HAYST法	オールペア（例：PICT/PictMaster等）
1	ツールの品質保証	あり	なし（困ったことやおかしな現象が起こっても、どちらのバグかすら、相談できない）
2	ツール	有償	無償
3	テスト件数	コントロール可能（網羅率を犠牲にすることで、テスト工数に入るテスト件数に制御可能）	コントロール不可能（網羅率１００%になるまでテストが作られる。出来上がったテストから適切に間引くことは出来ない）
4	禁則	・簡単な入力（テストや開発の経験なしでも可能） ・複雑な禁則でもマトリクス生成可能	・複雑な入力（入力ミスが発生するとテスト漏れとなる） ・複雑な（実際の製品仕様で多少複雑な）禁則条件があるとマトリクスができない
5	スコープ	・テスト戦略（計画） ・テスト要求分析 ・テストアーキテクチャ設計 ・テスト詳細設計 ・テスト実装/最適化 ・テスト実装の評価 ・テスト結果分析	・テスト実装（因子/水準が明らかになった状況でマトリクスを生成する）