

テスコンからの 納得できるテスト設計への挑戦

2018年6月15日

大段智広（おおだんともひろ）

てすにゃん

自己紹介



名前：

大段 智広（おおだん ともひろ）

お仕事：

開発技術導入支援（CI、静的解析等）

テスト関連の研修講師

社外活動：

ソフトウェアテスト系のコミュニティ&イベント
に出没中

- ✓ JaSST Kansai 実行委員
- ✓ テスト設計コンテスト'15~18（てすにゃん）
- ✓ 関西ソフトウェアテスト勉強会(WARAI)
- ✓ バグ票ワーストプラクティス検討プロジェクト

本日お伝えしたいこと

自分のテスト設計が
いまいちくと言われる)



自分もテスト設計に
いまいち納得できない！



そのためには...

本当に解決したい
“テストの意図や目的”

を理解し、押さえる！

テスコンの活動を通じて
これから2つのことを
お話しします

※テスコン：ASTER テスト設計コンテスト

1つめは、

「“テストの目的や意図”
を理解し、押さえる方法」

2つめは、
「実践した
自分たちのテスト設計」

アジェンダ

1. テスコン(テスト設計コンテスト)とは
2. なぜテスコン?
3. 納得できないテスト設計
4. "テスト設計"とは
5. "納得"とは
6. "納得できるテスト設計にするために"
7. 自分たちのテスト設計
8. まとめ

テスコン(テスト設計コンテスト)とは

指定のテストベースに対するテスト設計を行い、優劣を競う。
コンテストは各地域予選を行い、優秀と認められたチームが
決勝戦で争う。



<http://aster.or.jp/business/contest.html#system>

なぜテストコン?

参加当時のテストの悩み

- ✓ 会社のテストプロセスに沿ったテストしかしていない
- ✓ 現状でもテスト設計を行えるがより改善できるのでは?と感じている
- ✓ 開発が楽になる方法やQAの人ともっと連携できる方法があるはず...



納得できないテスト設計をしている
(より良いテスト設計をしたい)

納得できないテスト設計

テスト設計の悩み具体例

- テストの項目自体がなんか足りない
- あるけどロジカルじゃない
- あるテストとあるテストのやり方に整合性や統一性がない
- 網羅度が十分じゃない
- テスト項目、テストケース導出の根拠が不十分
- 項目の洗い出しやテストの実施が俗人的
- そもそも開発製品が何やりたいかさっぱりわからん

etc,...

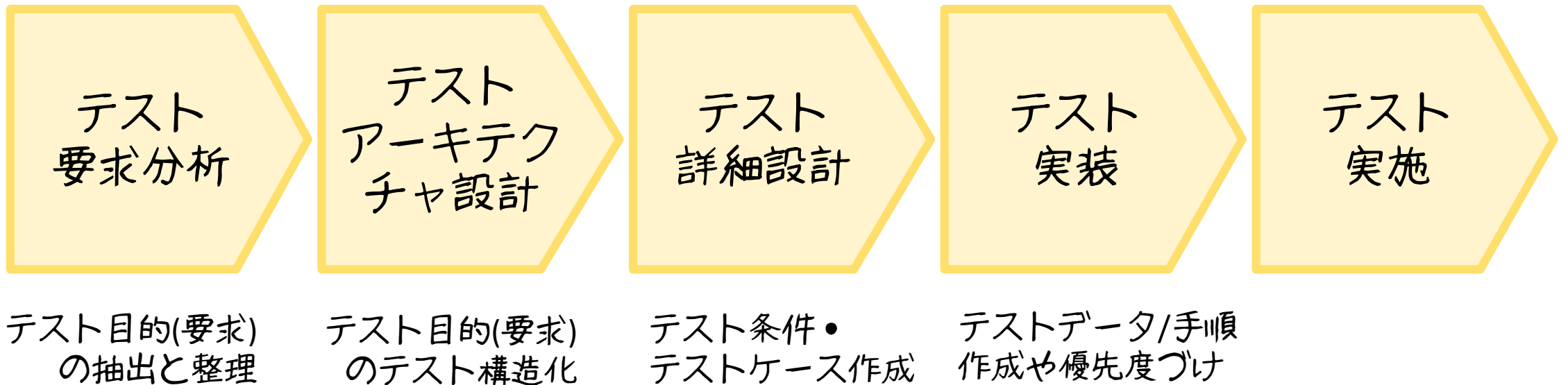


“テスト設計”と“納得”を改めて考える

“テスト設計”ってなんだ

テスト開発プロセス

ソフトウェア開発プロセスと対応したテスト開発プロセスを推奨している。



"テスト設計"とは

テスト設計 (test design) 【JSTQB 引用】

(1) test design specification を参照のこと。

テストアイテムのテスト条件(カバレッジアイテム)、詳細なテストアプローチ、及び、関連する高位レベルテストケースを記述したドキュメント。

(2)

概略的なテスト目的を具体的なテスト条件とテストケースに変換するプロセス。

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

テスト
実施

Input

テスト設計

Output

“納得”ってなんだ

"納得"とは

『他人の考え・行為を理解し、もっともだと認めること』

(デジタル大辞泉引用)

他人



考え



行為

自分



なるほど...
そういうこと

理解
&
認める

"納得"とは

例えば…他人の考え・行為が見えなかったり…

他人



考え



行為

自分



見えない！...

観る

"納得"とは

見えたとしても、理解のための意図や目的は理解できないと…

他人



考え



行為

自分



わからん...

観る

"納得"とは

認められず、無意識に拒否したり、理解できず混乱する。

他人



考え



行為

自分

拒否



混乱

"納得"とは

自分自身の拒否がその相手（他人）に伝わると…

他人



考え



拒否

自分

不満



混乱

“納得”とは

行く先は『争い』…。

他人



自分

拒否



混乱

「納得」するためには…

下記の取り組みが必要になる。

- ① 他人（自分含む）を決める
- ② 他人の行為・考えに触れる
- ③ 理解するための方法を持つ
- ④ 他人と考えを認め合う術を持つ

『他人の考え・行為を理解し、もっともだと認める』

①

②

③

④

『納得できるテスト設計』にするために

『テストに対する他人の考え・行為の目的や意図を理解し、
もっともだと認める』

そのためには

- ① 他人（自分含む）を決める
- ② 他人の行為・考えに触れる
- ③ 理解するための方法を持つ
- ④ 他人と考えを認め合う術を持つ

- ⇒ ① 文脈、場面状況を決める
- ⇒ ② 他人の行為・考えを話し合う
- ⇒ ③ 可視化・整理する
- ⇒ ④ 文脈、場面状況を確認する

テスト設計した結果

自分たちのテスト設計

① 文脈、場面状況を決める

No. 内容

- 1 OSSプロジェクトへのアジャイルQAが分かる
- 2 エンドユーザのニーズに合わせた市場リリースができる
- 3 アジャイルのテストプロセスが知りたい
- 4 テストタイプ教えて
- 5 自動テストってどこをどんだけやればいいの？
- 6 テストケースの作り方が分かる
- 7 テストケースがメンテしやすい
- 8 アジャイルテスト参考文献を探すときに便利
- 9 これやってみよう、うまく行きそうと思える
- 10 だれでもアジャイルテストがわかる
- 11 抽象的にも具体的にテストの内容を説明できる
- 12 アジャイルプロジェクトでのテストについてこれ見てと言える

上から実践

自分たちのテスト設計

① 文脈、場面状況を決める

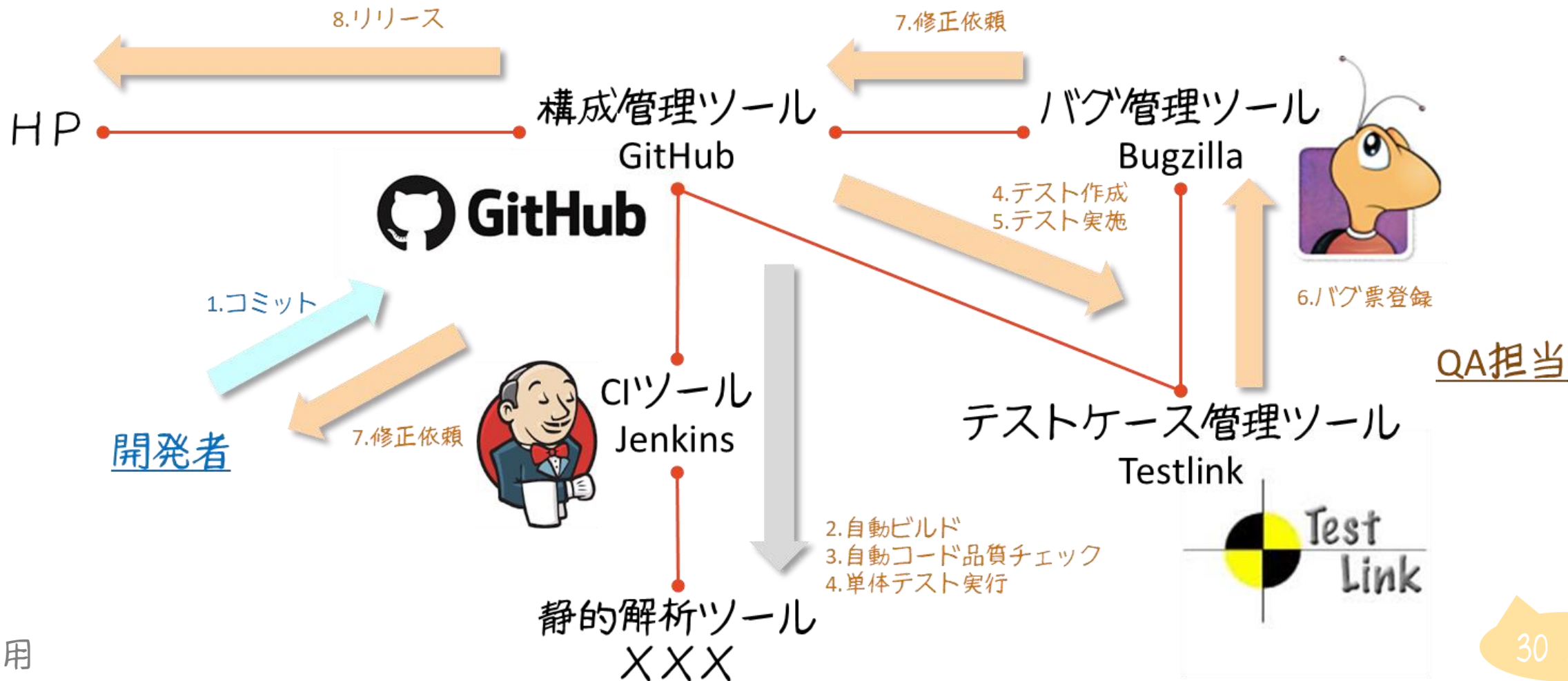
- 開発製品 : オープンソースソフトウェアの通信カウオケシステム
- 立場 : コミュニティメンバ QA担当
- リソース : 2人
- リリース周期 : 2週間(最新版)、6か月(安定板)

製品のフィーチャーはコミュニティ開発者の意思で、適宜実装される。



自分たちのテスト設計

① 文脈、場面状況を決める



自分たちのテスト設計

② 他人の行為・考えを話し合う

「製品のリリースを止めるバグ
を許容範囲まで減らしたい！」

なんで？



自分たちのテスト設計

③ 可視化・整理する 理由を理解できる要素まで分解する。



バグがなかなか治らない

ソフトウェア構造が
分からない

フィードバックが多い
(バグ票が大量に出る)

提起課題の全体像
が分からない

課題の全体を整理
する人がいない

修正箇所の影響範囲が
分からない

コードをいじると
不具合が出る

バグが見過ごされる

ソフトウェア構造の全体像を簡
単に把握できる手段がない

テストのモチベー
ションが低い

テストが十分にでき
ていない

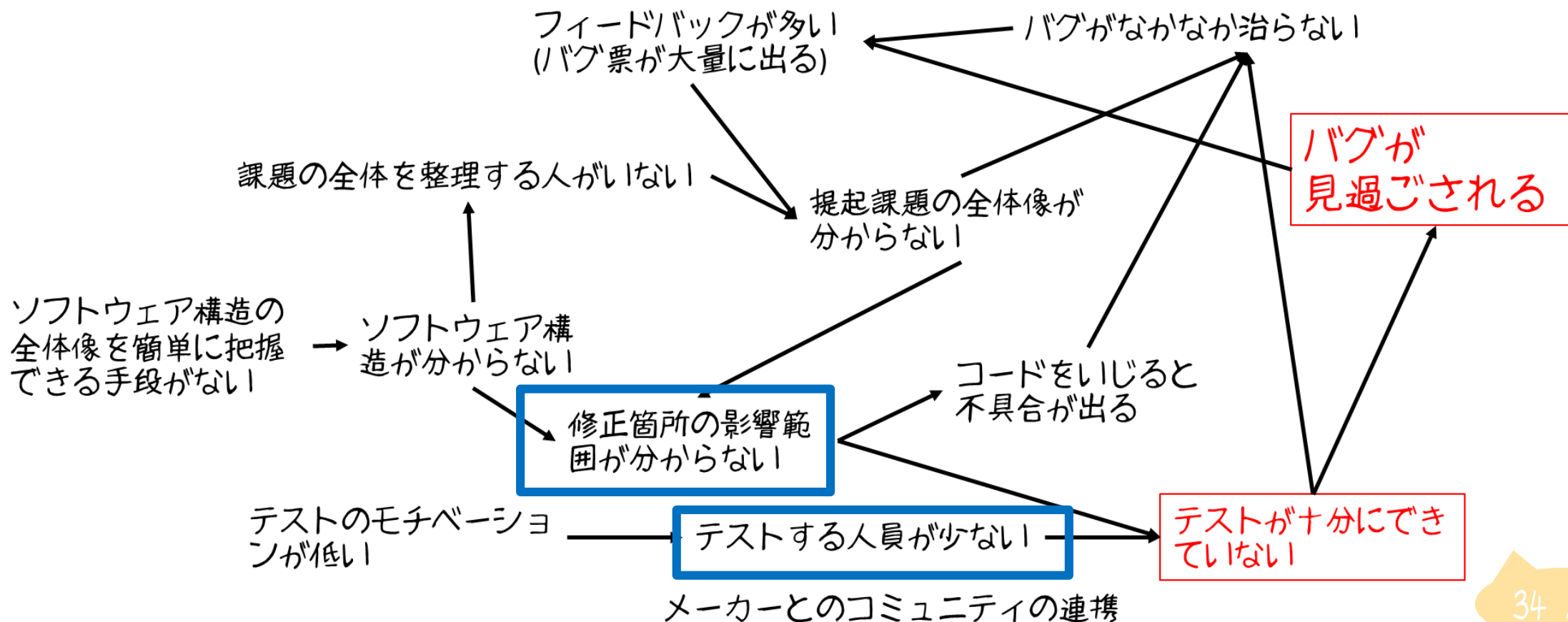
テストする人員が少ない

③ 可視化・整理する
問題の因果関係を整理する。



自分たちのテスト設計

③ 可視化・整理する 解決すべき箇所を特定する。



自分たちのテスト設計

④ 文脈、場面状況を確認する 確認しながら開発しているテスト設計を行う。

自分たちのテスト設計

① 文脈、場面状況を決める

●開発製品 : オープンソースソフトウェアの通信カラオケシステム

自分たちのテスト設計

① 文脈、場面状況を決める

8.リリース

7.修正依頼

される。

自分たちのテスト設計

③ 可視化・整理する

フィードバックが多い
(バグ票が大量に出る)

バグがなかなか治らない

課題の全体を整理する人がいない

提起課題の全体像が
分からない

バグが
見過ごされる

ソフトウェア構造の
全体像を簡単に把握
できる手段がない

→

ソフトウェア構
造が分からない

修正箇所の影響範
囲が分からない

コードをいじると
不具合が出る

テストのモチベーショ
ンが低い

テストする人員が少ない

テストが十分にでき
ていない

解決すべき箇所を特定する

メーカーとのコミュニティの連携

38 / 64

リリースについての課題

1. 製品のリリースを止める事象およびバグをどう特定するか?
2. テストで対処する/しない事象はどのように決めるか?
3. 対処するためにどんなテストをどれくらい行うか?

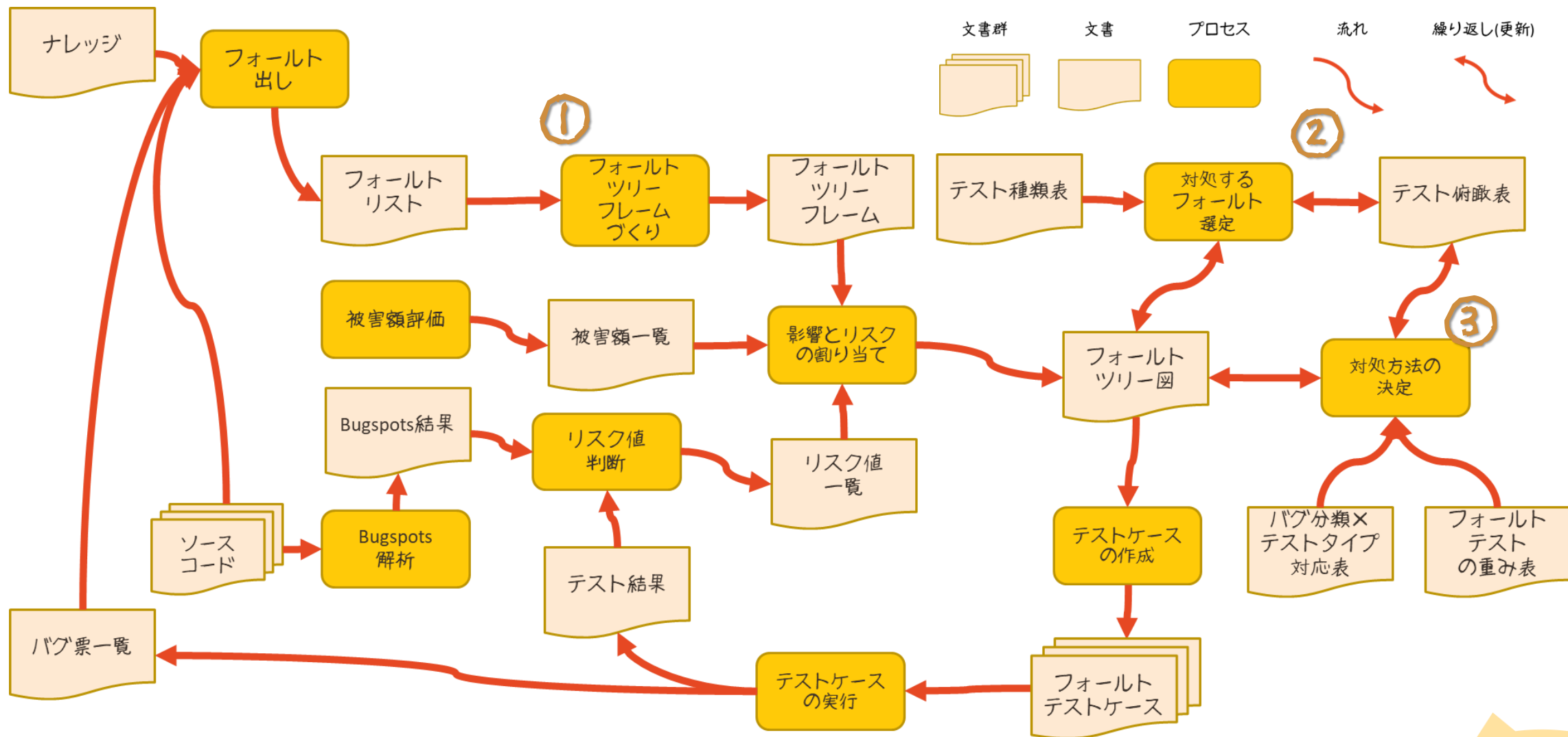


リリースについての課題

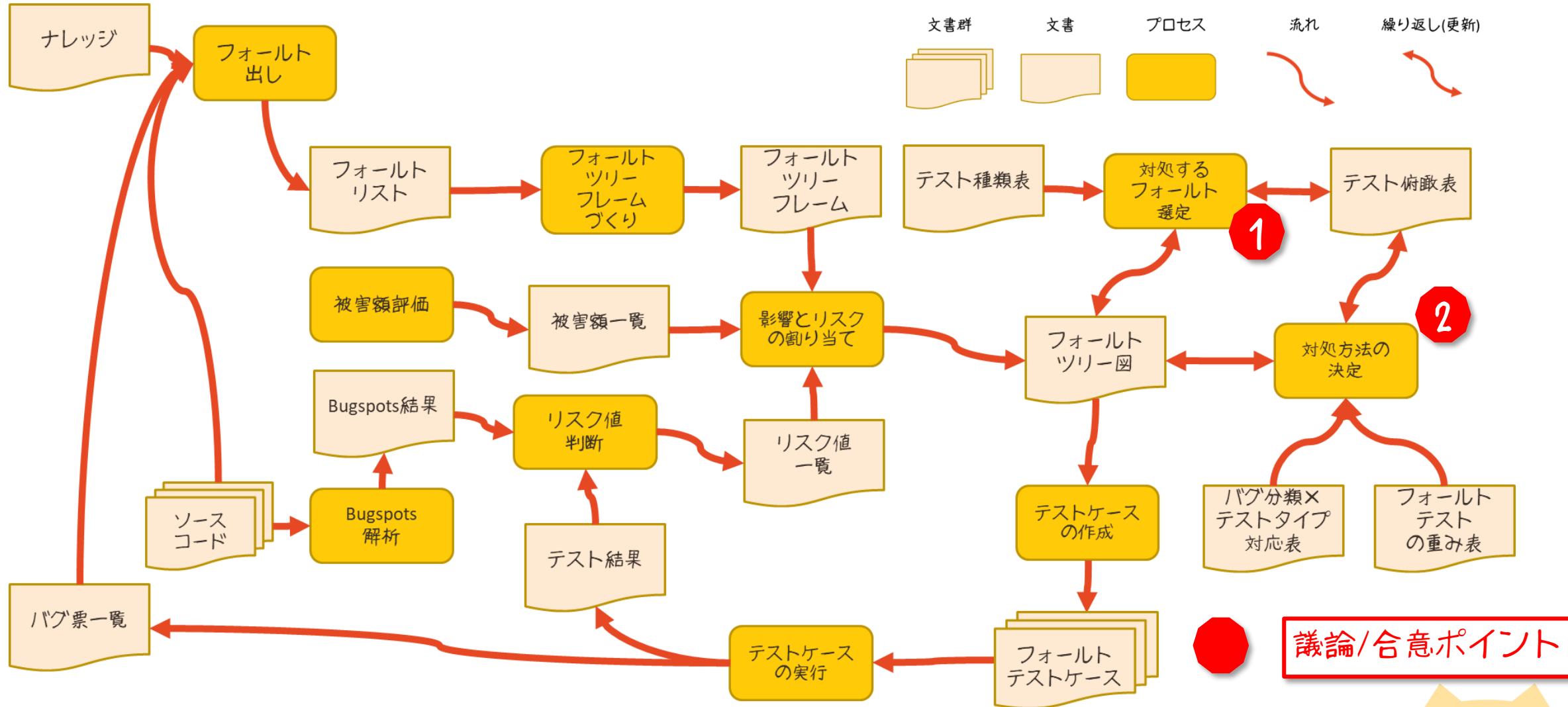
- ① 製品のリリースを止める事象およびバグをどう特定するか？
- ② テストで対処する/しない事象はどのように決めるか？
- ③ 対処するためにどんなテストをどれくらい行うか？



テストの全体像



テストの全体像



テストアーキテクチャ(フォールト)

1

事象の影響度(被害金額)と発生確率(リスク値)から
対処すべき事象(フォールト)を特定し、対応者について合意
する。

被害金額



コード変更リスク



フォールトの
やばさ

公開用

身体的 リスク	不快な音波
	大音量
	気分が悪くなる映像が流れる
	チカチカ(光の点滅)
金銭的 リスク	課金ができない
	課金の計算が正しくない
	課金データが改ざんされる
社会的 要請	クラッキングされる
	踏み台にされる
	操作データが外部にもれる
	楽曲データがとられる
物理的破壊 リスク	ハードウェアが壊れる
	バックアップデータが壊れる
	サーバーのデータを破壊する
	大量にデータをDL

身体的リスク	1000万円
金銭的リスク	500万
社会的要請	5億円
物理的破壊 リスク	10万円

```
graph LR; subgraph Layer1 [ ]; direction TB; T[Top事象]; end; subgraph Layer2 [ ]; direction TB; S1[サブ事象]; S2[サブ事象]; end; subgraph Layer3 [ ]; direction TB; C1[原因事象]; C2[原因事象]; end; T --> S1; T --> S2; S1 --> C1; S2 --> C2;
```

ユーザ事象
(トップ階層)

システム事象
(ミドル階層)

原因事象
(ボトム階層)

Top事象

サブ事象

サブ事象

原因事象

原因事象

ユーザ事象 (トップ階層)	システム事象 (ミドル階層)	原因事象 (ボトム階層)
課金の計算が正しくない or	連曲モードで、単曲モード課金されてしまう	課金形態設定で単曲モードが設定されない 15-1
	単曲モードで、連曲モード課金されてしまう	課金形態設定で連曲モードが設定されない 15-2
	キャンセルしても課金されてしまう(返金されない)	キャンセル信号が送信されない 15-3
	設定した割引率が適用されない	課金リストの参照条件がずれている 15-4

The diagram illustrates the development workflow for a game, showing the relationship between code changes, bugfixes, and releases.

Terminal Window:

```

$ ./code/pythonmain.py git bugfixes -s
[stamping] 2020-07-20 12:00:00
$ git checkout -b bugfixes
$ git pull
Found 32 bugfix commits, with 34 hotspots:

Example:
- Fix window position to top-left for new player windows.
- Fix end-cpu-cycling in the CPU player
- v1.4 written by Matt Ferrell.
- Fix cosmetic typo in "may be corrupt" message
- Request hardware accelerated and double-buffered display fr
... 50 more

```

Hotspots:

```

Hotspots:
  #13 - pckdb.py
  #14 - pckdb.py
  #15 - pckdb.py
  #16 - pckdb.py
  #17 - pckdb.py
  #18 - pckdb.py
  #19 - pckdb.py
  #20 - pckdb.py
  #21 - pckdb.py
  #22 - pckdb.py
  #23 - pckdb.py
  #24 - pckdb.py
  #25 - pckdb.py
  #26 - pckdb.py
  #27 - pckdb.py
  #28 - pckdb.py
  #29 - pckdb.py
  #30 - pckdb.py
  #31 - pckdb.py
  #32 - pckdb.py
  #33 - pckdb.py
  #34 - pckdb.py
  #35 - pckdb.py
  #36 - pckdb.py
  #37 - pckdb.py
  #38 - pckdb.py
  #39 - pckdb.py
  #40 - pckdb.py
  #41 - pckdb.py
  #42 - pckdb.py
  #43 - pckdb.py
  #44 - pckdb.py
  #45 - pckdb.py
  #46 - pckdb.py
  #47 - pckdb.py
  #48 - pckdb.py
  #49 - pckdb.py
  #50 - pckdb.py
  #51 - pckdb.py
  #52 - pckdb.py
  #53 - pckdb.py
  #54 - pckdb.py
  #55 - pckdb.py
  #56 - pckdb.py
  #57 - pckdb.py
  #58 - pckdb.py
  #59 - pckdb.py
  #60 - pckdb.py
  #61 - pckdb.py
  #62 - pckdb.py
  #63 - pckdb.py
  #64 - pckdb.py
  #65 - pckdb.py
  #66 - pckdb.py
  #67 - pckdb.py
  #68 - pckdb.py
  #69 - pckdb.py
  #70 - pckdb.py
  #71 - pckdb.py
  #72 - pckdb.py
  #73 - pckdb.py
  #74 - pckdb.py
  #75 - pckdb.py
  #76 - pckdb.py
  #77 - pckdb.py
  #78 - pckdb.py
  #79 - pckdb.py
  #80 - pckdb.py
  #81 - pckdb.py
  #82 - pckdb.py
  #83 - pckdb.py
  #84 - pckdb.py
  #85 - pckdb.py
  #86 - pckdb.py
  #87 - pckdb.py
  #88 - pckdb.py
  #89 - pckdb.py
  #90 - pckdb.py
  #91 - pckdb.py
  #92 - pckdb.py
  #93 - pckdb.py
  #94 - pckdb.py
  #95 - pckdb.py
  #96 - pckdb.py
  #97 - pckdb.py
  #98 - pckdb.py
  #99 - pckdb.py
  #100 - pckdb.py
  #101 - pckdb.py
  #102 - pckdb.py
  #103 - pckdb.py
  #104 - pckdb.py
  #105 - pckdb.py
  #106 - pckdb.py
  #107 - pckdb.py
  #108 - pckdb.py
  #109 - pckdb.py
  #110 - pckdb.py
  #111 - pckdb.py
  #112 - pckdb.py
  #113 - pckdb.py
  #114 - pckdb.py
  #115 - pckdb.py
  #116 - pckdb.py
  #117 - pckdb.py
  #118 - pckdb.py
  #119 - pckdb.py
  #120 - pckdb.py
  #121 - pckdb.py
  #122 - pckdb.py
  #123 - pckdb.py
  #124 - pckdb.py
  #125 - pckdb.py
  #126 - pckdb.py
  #127 - pckdb.py
  #128 - pckdb.py
  #129 - pckdb.py
  #130 - pckdb.py
  #131 - pckdb.py
  #132 - pckdb.py
  #133 - pckdb.py
  #134 - pckdb.py
  #135 - pckdb.py
  #136 - pckdb.py
  #137 - pckdb.py
  #138 - pckdb.py
  #139 - pckdb.py
  #140 - pckdb.py
  #141 - pckdb.py
  #142 - pckdb.py
  #143 - pckdb.py
  #144 - pckdb.py
  #145 - pckdb.py
  #146 - pckdb.py
  #147 - pckdb.py
  #148 - pckdb.py
  #149 - pckdb.py
  #150 - pckdb.py
  #151 - pckdb.py
  #152 - pckdb.py
  #153 - pckdb.py
  #154 - pckdb.py
  #155 - pckdb.py
  #156 - pckdb.py
  #157 - pckdb.py
  #158 - pckdb.py
  #159 - pckdb.py
  #160 - pckdb.py
  #161 - pckdb.py
  #162 - pckdb.py
  #163 - pckdb.py
  #164 - pckdb.py
  #165 - pckdb.py
  #166 - pckdb.py
  #167 - pckdb.py
  #168 - pckdb.py
  #169 - pckdb.py
  #170 - pckdb.py
  #171 - pckdb.py
  #172 - pckdb.py
  #173 - pckdb.py
  #174 - pckdb.py
  #175 - pckdb.py
  #176 - pckdb.py
  #177 - pckdb.py
  #178 - pckdb.py
  #179 - pckdb.py
  #180 - pckdb.py
  #181 - pckdb.py
  #182 - pckdb.py
  #183 - pckdb.py
  #184 - pckdb.py
  #185 - pckdb.py
  #186 - pckdb.py
  #187 - pckdb.py
  #188 - pckdb.py
  #189 - pckdb.py
  #190 - pckdb.py
  #191 - pckdb.py
  #192 - pckdb.py
  #193 - pckdb.py
  #194 - pckdb.py
  #195 - pckdb.py
  #196 - pckdb.py
  #197 - pckdb.py
  #198 - pckdb.py
  #199 - pckdb.py
  #200 - pckdb.py
  #201 - pckdb.py
  #202 - pckdb.py
  #203 - pckdb.py
  #204 - pckdb.py
  #205 - pckdb.py
  #206 - pckdb.py
  #207 - pckdb.py
  #208 - pckdb.py
  #209 - pckdb.py
  #210 - pckdb.py
  #211 - pckdb.py
  #212 - pckdb.py
  #213 - pckdb.py
  #214 - pckdb.py
  #215 - pckdb.py
  #216 - pckdb.py
  #217 - pckdb.py
  #218 - pckdb.py
  #219 - pckdb.py
  #220 - pckdb.py
  #221 - pckdb.py
  #222 - pckdb.py
  #223 - pckdb.py
  #224 - pckdb.py
  #225 - pckdb.py
  #226 - pckdb.py
  #227 - pckdb.py
  #228 - pckdb.py
  #229 - pckdb.py
  #230 - pckdb.py
  #231 - pckdb.py
  #232 - pckdb.py
  #233 - pckdb.py
  #234 - pckdb.py
  #235 - pckdb.py
  #236 - pckdb.py
  #237 - pckdb.py
  #238 - pckdb.py
  #239 - pckdb.py
  #240 - pckdb.py
  #241 - pckdb.py
  #242 - pckdb.py
  #243 - pckdb.py
  #244 - pckdb.py
  #245 - pckdb.py
  #246 - pckdb.py
  #247 - pckdb.py
  #248 - pckdb.py
  #249 - pckdb.py
  #250 - pckdb.py
  #251 - pckdb.py
  #252 - pckdb.py
  #253 - pckdb.py
  #254 - pckdb.py
  #255 - pckdb.py
  #256 - pckdb.py
  #257 - pckdb.py
  #258 - pckdb.py
  #259 - pckdb.py
  #260 - pckdb.py
  #261 - pckdb.py
  #262 - pckdb.py
  #263 - pckdb.py
  #264 - pckdb.py
  #265 - pckdb.py
  #266 - pckdb.py
  #267 - pckdb.py
  #268 - pckdb.py
  #269 - pckdb.py
  #270 - pckdb.py
  #271 - pckdb.py
  #272 - pckdb.py
  #273 - pckdb.py
  #274 - pckdb.py
  #275 - pckdb.py
  #276 - pckdb.py
  #277 - pckdb.py
  #278 - pckdb.py
  #279 - pckdb.py
  #280 - pckdb.py
  #281 - pckdb.py
  #282 - pckdb.py
  #283 - pckdb.py
  #284 - pckdb.py
  #285 - pckdb.py
  #286 - pckdb.py
  #287 - pckdb.py
  #288 - pckdb.py
  #289 - pckdb.py
  #290 - pckdb.py
  #291 - pckdb.py
  #292 - pckdb.py
  #293 - pckdb.py
  #294 - pckdb.py
  #295 - pckdb.py
  #296 - pckdb.py
  #297 - pckdb.py
  #298 - pckdb.py
  #299 - pckdb.py
  #300 - pckdb.py
  #301 - pckdb.py
  #302 - pckdb.py
  #303 - pckdb.py
  #304 - pckdb.py
  #305 - pckdb.py
  #306 - pckdb.py
  #307 - pckdb.py
  #308 - pckdb.py
  #309 - pckdb.py
  #310 - pckdb.py
  #311 - pckdb.py
  #312 - pckdb.py
  #313 - pckdb.py
  #314 - pckdb.py
  #315 - pckdb.py
  #316 - pckdb.py
  #317 - pckdb.py
  #318 - pckdb.py
  #319 - pckdb.py
  #320 - pckdb.py
  #321 - pckdb.py
  #322 - pckdb.py
  #323 - pckdb.py
  #324 - pckdb.py
  #325 - pckdb.py
  #326 - pckdb.py
  #327 - pckdb.py
  #328 - pckdb.py
  #329 - pckdb.py
  #330 - pckdb.py
  #331 - pckdb.py
  #332 - pckdb.py
  #333 - pckdb.py
  #334 - pckdb.py
  #335 - pckdb.py
  #336 - pckdb.py
  #337 - pckdb.py
  #338 - pckdb.py
  #339 - pckdb.py
  #340 - pckdb.py
  #341 - pckdb.py
  #342 - pckdb.py
  #343 - pckdb.py
  #344 - pckdb.py
  #345 - pckdb.py
  #346 - pckdb.py
  #347 - pckdb.py
  #348 - pckdb.py
  #349 - pckdb.py
  #350 - pckdb.py
  #351 - pckdb.py
  #352 - pckdb.py
  #353 - pckdb.py
  #354 - pckdb.py
  #355 - pckdb.py
  #356 - pckdb.py
  #357 - pckdb.py
  #358 - pckdb.py
  #359 - pckdb.py
  #360 - pckdb.py
  #361 - pckdb.py
  #362 - pckdb.py
  #36
```


フォールトツリー作成方法

まず、三層に分けてフォールトツリーを考える。



フォールトツリー作成方法

ステークホルダーの要求からTop事象を出す。



より導出するため、要求に対して
ガイドワードを適用し、Top事象を洗い出す。

要求からのTop事象(フォールト)だし

要求+HAZOPガイドワードにより、Top事象を効率よく導出する。

ユーザ



要求



ガイドワード



フォールト

起きてほしくない事象

オーナー

お客さんが利用
してくれ、儲かる
(課金ができる)



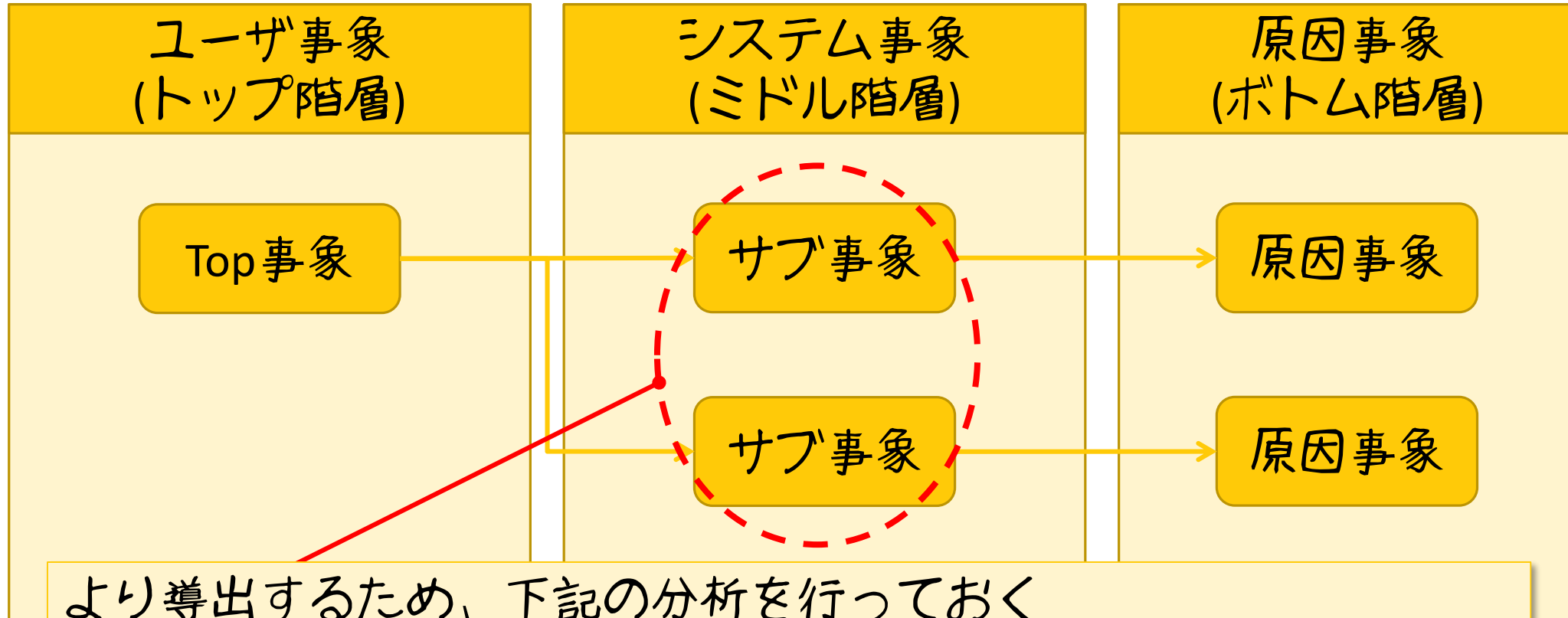
種類：違う



課金額を誤る

フォールトツリー作成方法

また他にハードとソフトの分析を行ってから作成する。



より導出するため、下記の分析を行っておく

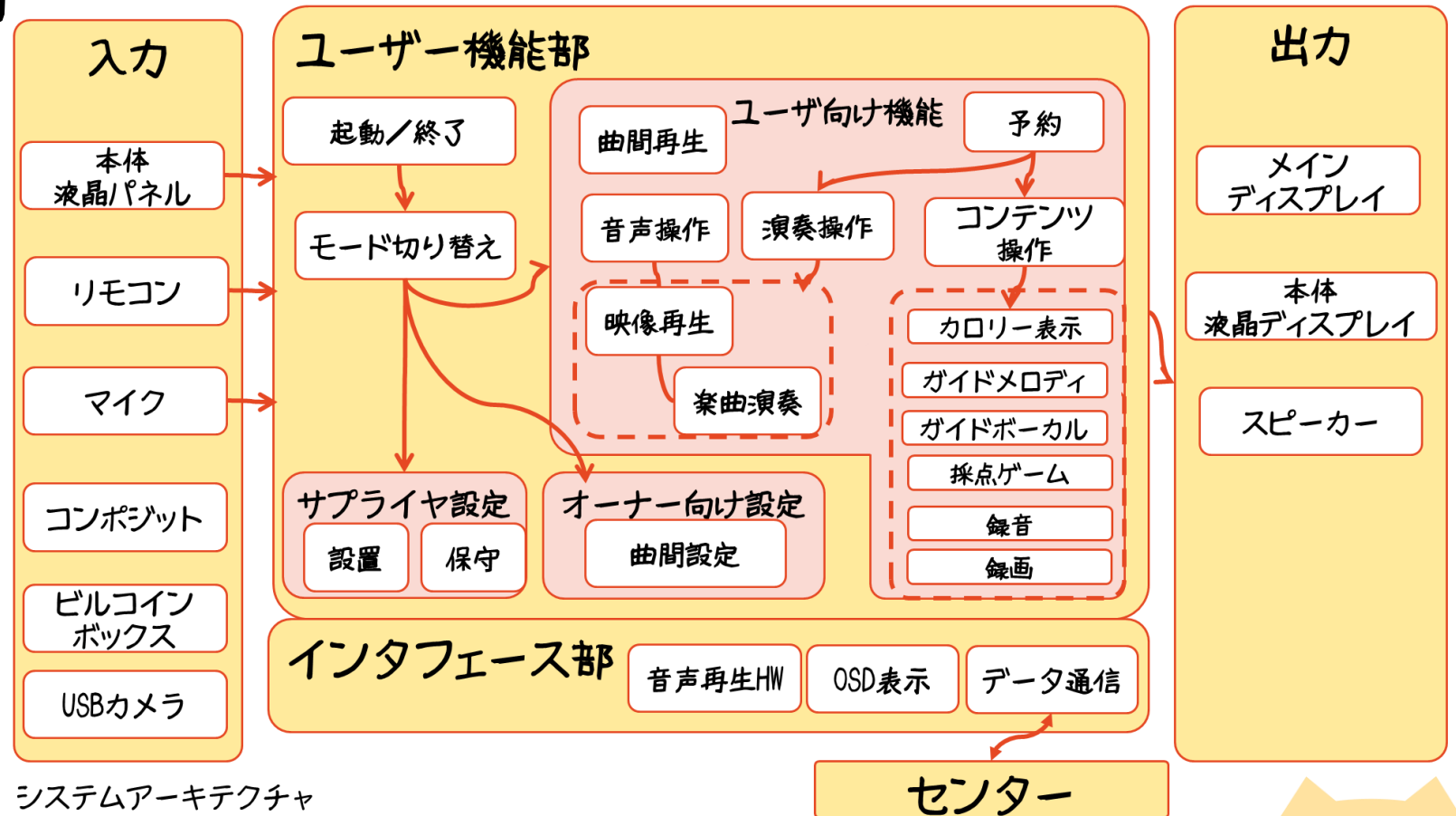
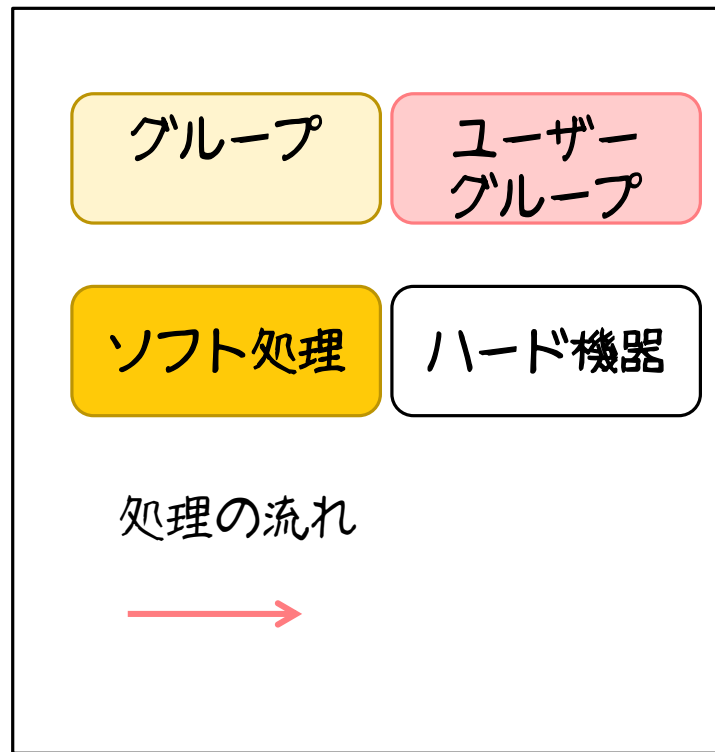
1. システムのハード構成: 機器の種類、機器の構成
2. システムのソフト構成: 処理の種類、処理実装のファイル構成

※想定でもよいので出す

ハードの分析:機器の構成

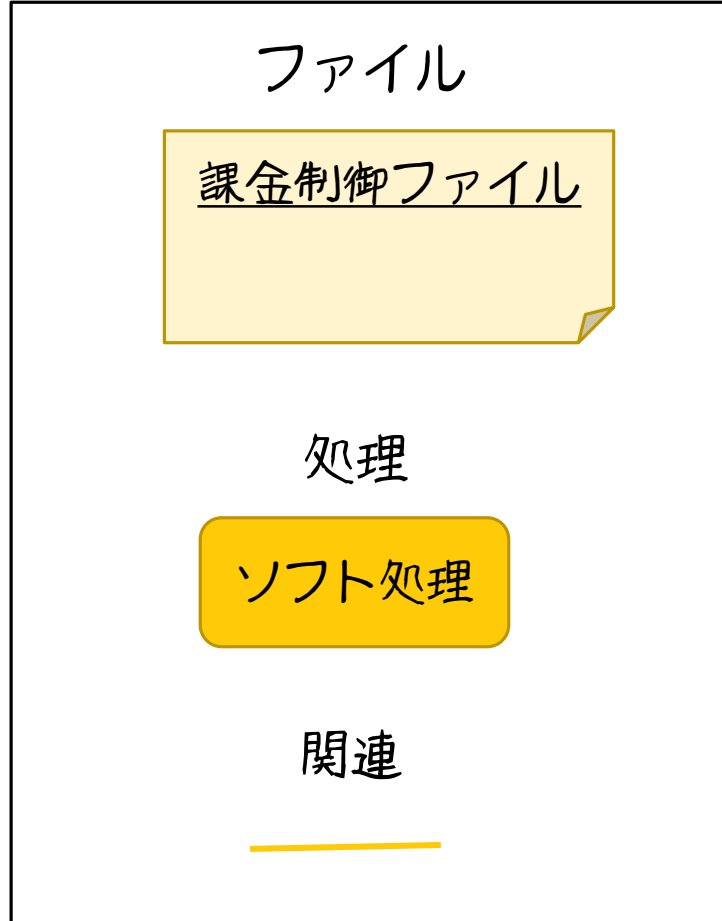
ハードウェア機器の構成を考える。

例

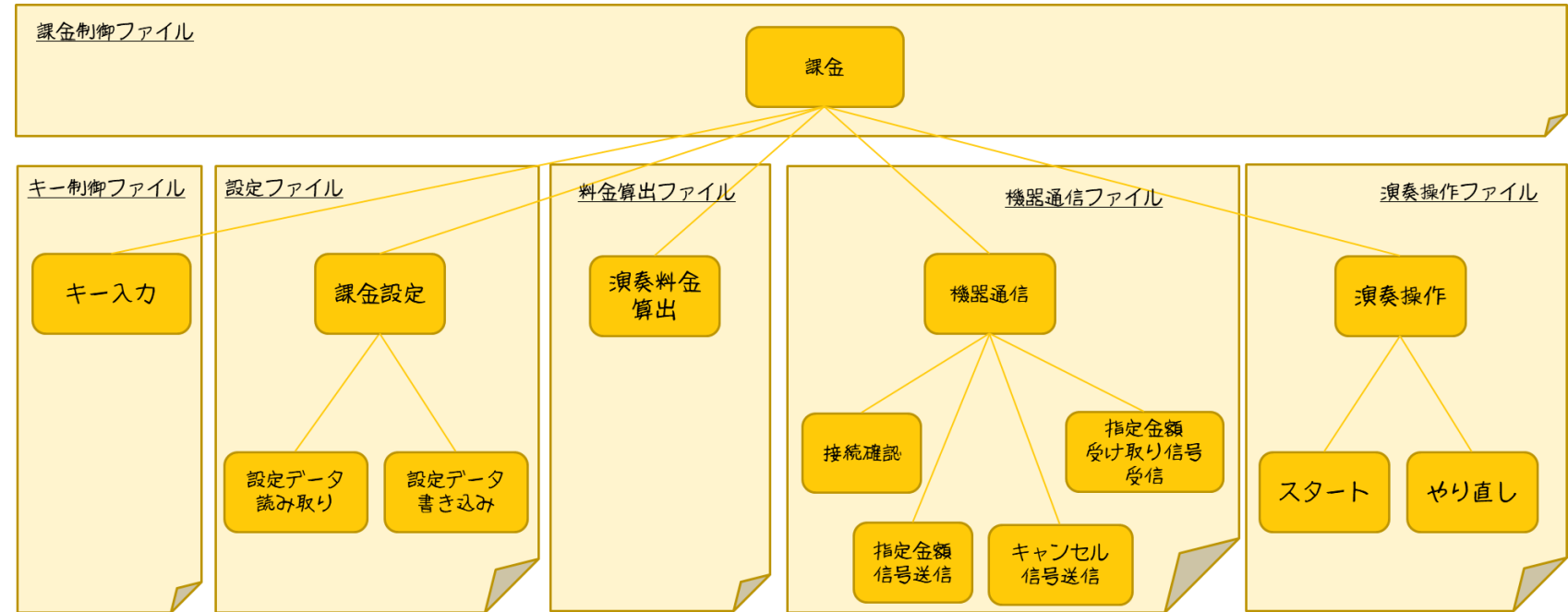


ソフトの分析:処理実装のファイル構成

ソフトウェア処理実装のファイル構成を考える。



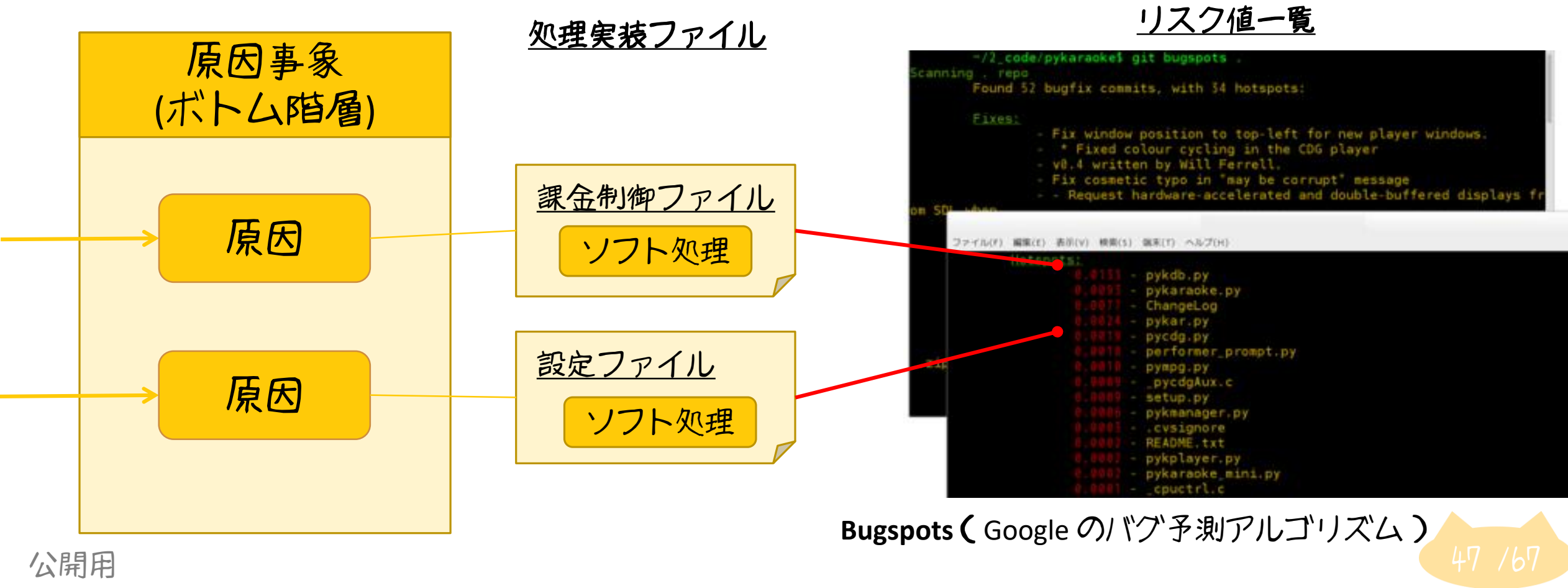
例



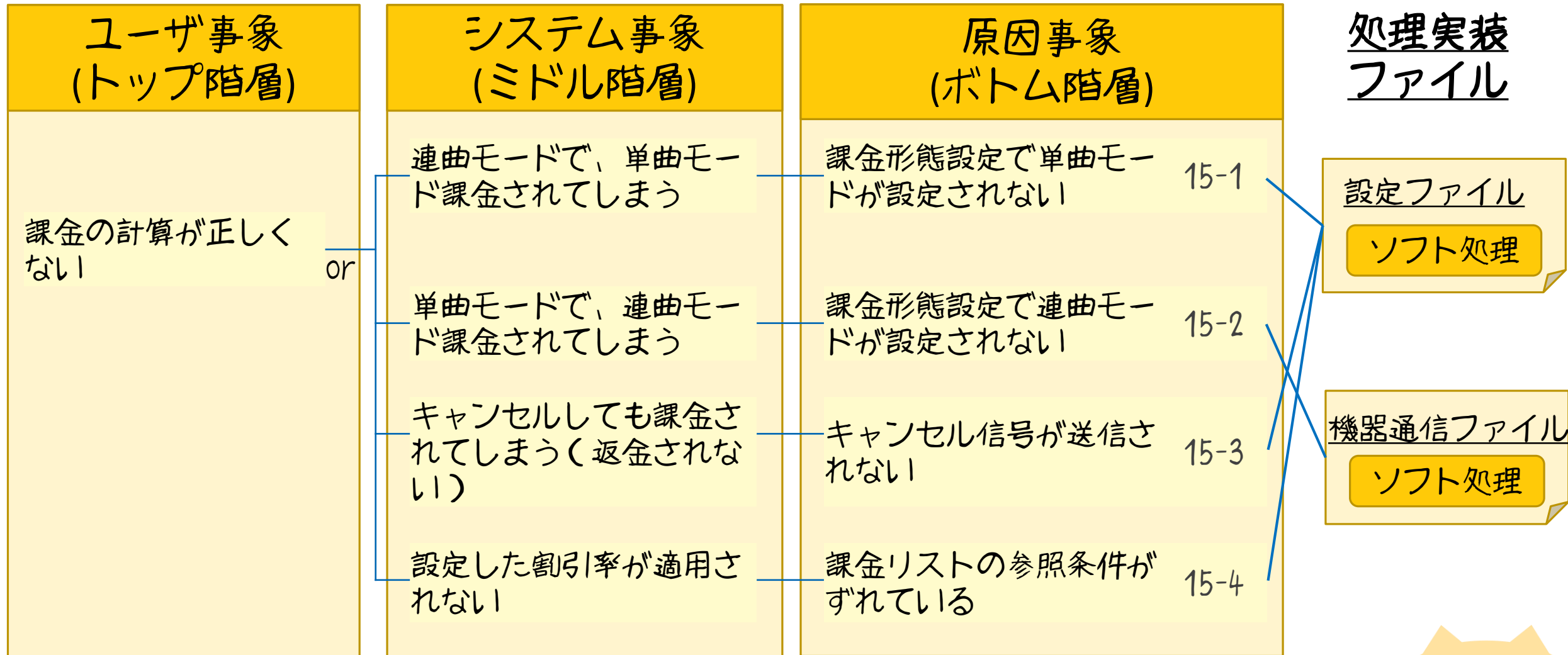
※可能であれば、処理の流れも書く。

リスク値の算出と割り当て

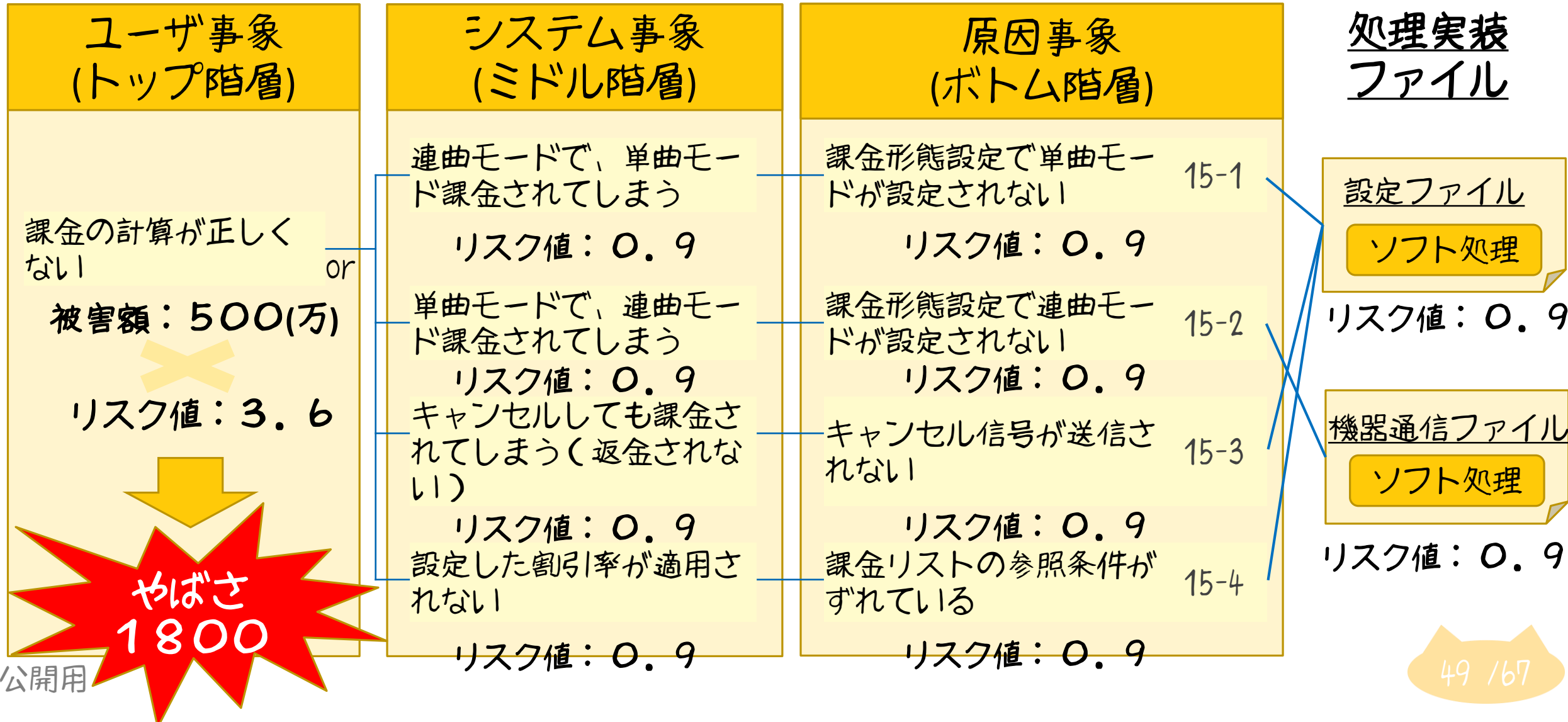
原因に処理実装ファイルを紐づけ、リスク値を算出・割り当てる。



フォールトツリーフレームサンプル



フォールトツリーフレームサンプル



テストアーキテクチャ(テスト設計方針)

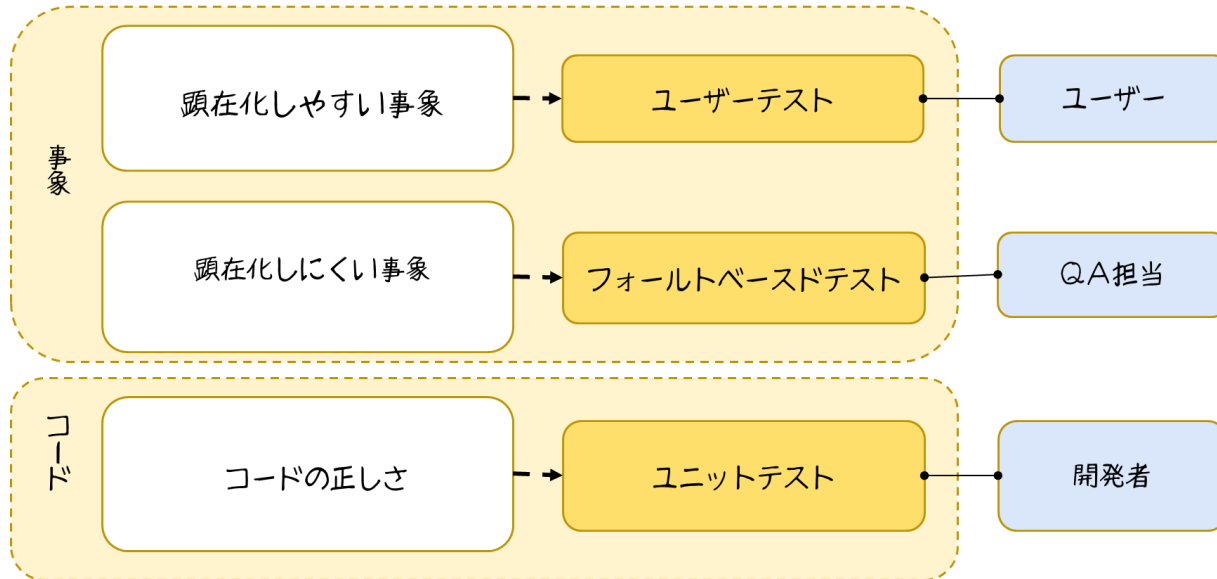
テストで対処する起こってほしくない事象のために、テスト設計方針を合意する。

2

テスト対象

テスト設計方針

テスト実施者



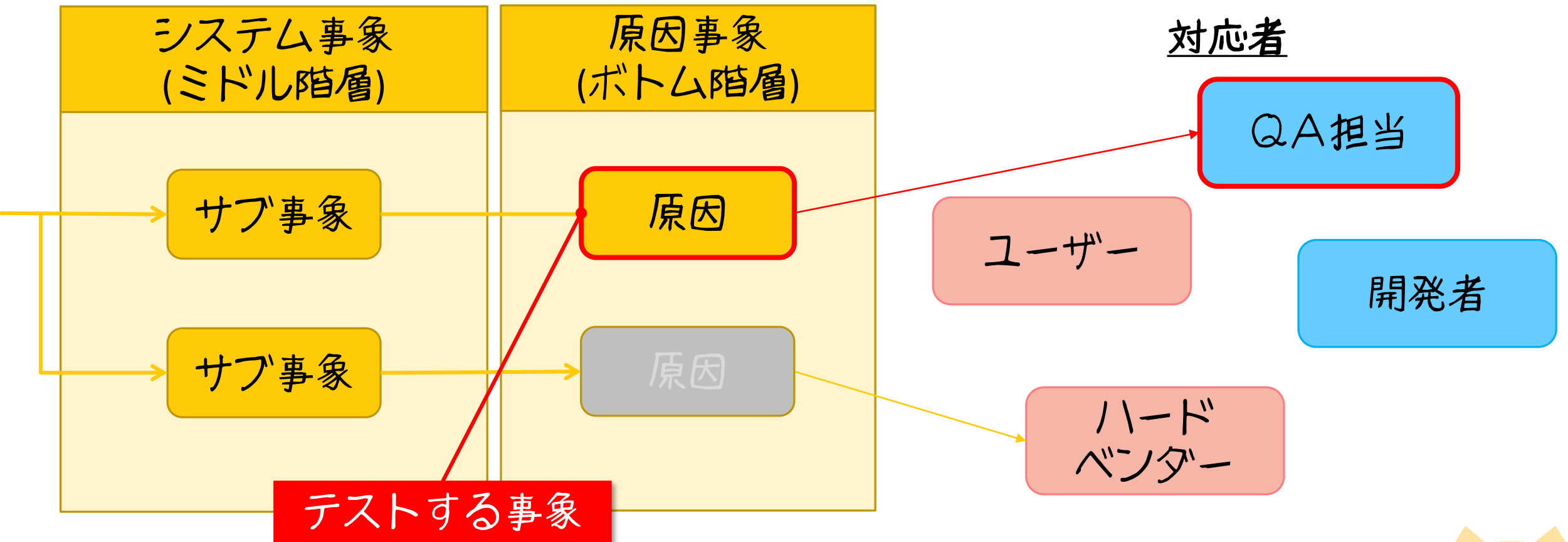
フォールトテストの重み表

		テストタイプ※1				
		データ	条件	シナリオ	イベント / タイミング	構成
テスト の 重 み	軽 最低限のテスト	有効/無効	重複結果なし	C0	0-switch	1-wise
	中	+同値クラス	無効なし	C1	セル網羅	2-wise
	重	+境界値	完全	C2	1-switch	3-wise

		テストタイプ				
		データ	条件	シナリオ	イベント / タイミング	構成
テスト の 重 み	軽 最低限のテスト	31.00%	25.00%	56.76%	61.67%	23.57%
	中	71.00%	45.00%	77.35%	83.89%	80.71%
	重	95.00%	95.00%	95.00%	95.00%	95.00%

対処する事象(フォールト)の選定と対処方法割り当て

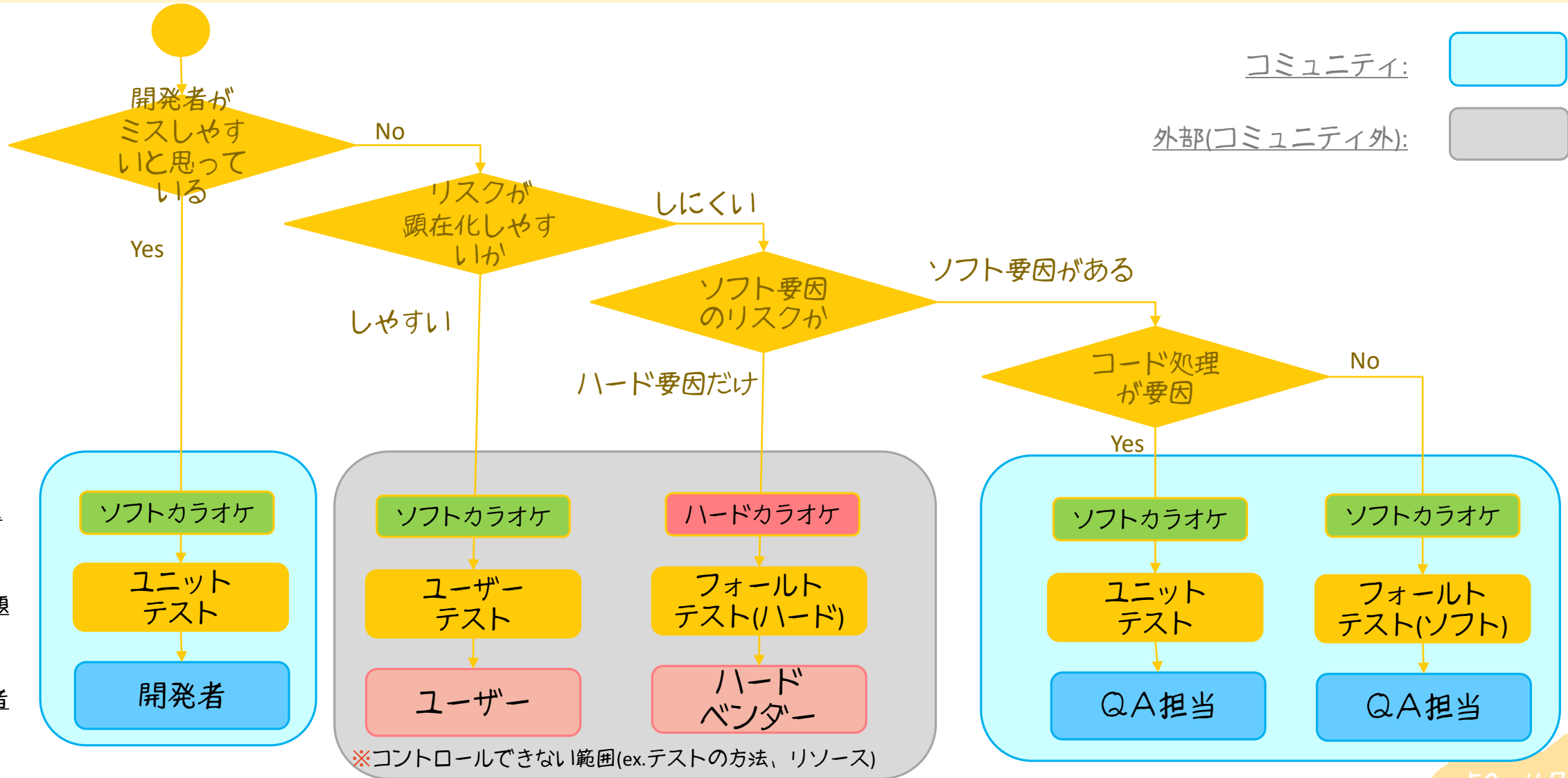
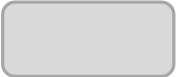
QAが対処すべきかを合意する。



事象ごとのテスト担当決めフロー

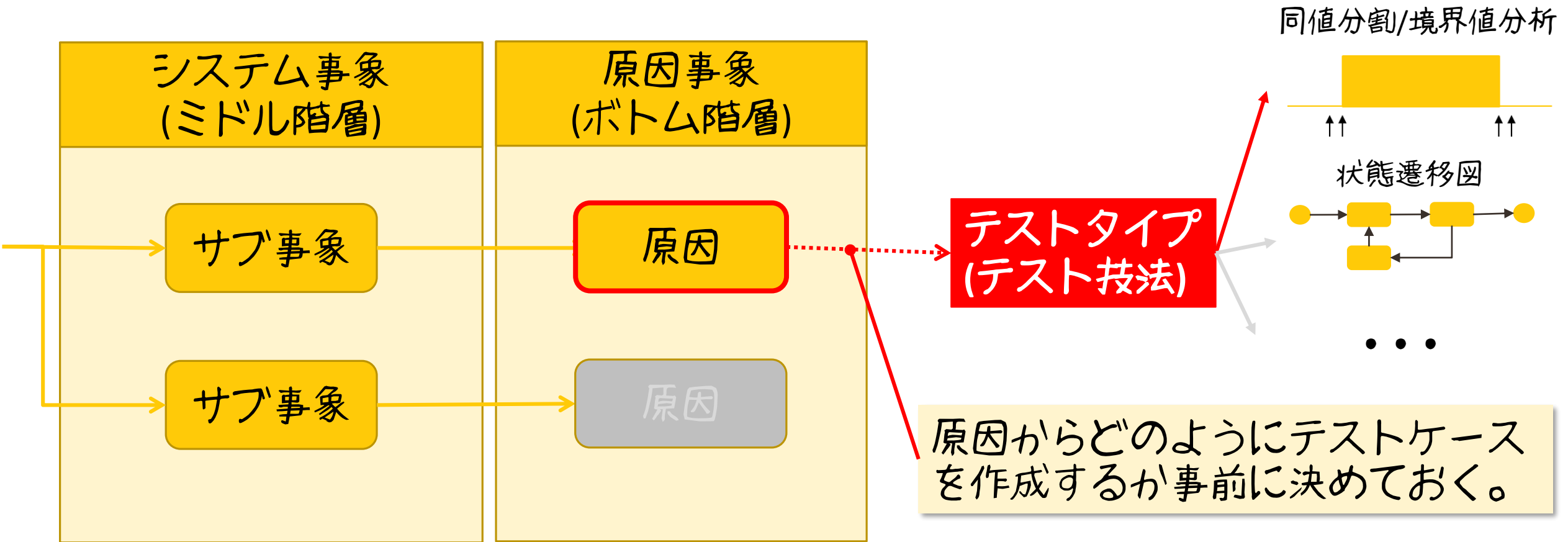
コミュニティ:

外部(コミュニティ外):



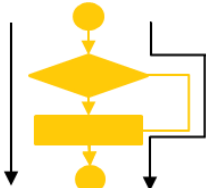
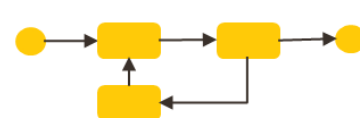
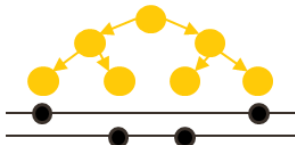
対処する事象(フォールト)の選定と対処方法割り当て

QAが対処すべき事象に対処方法(テストタイプ)を割り当てる。



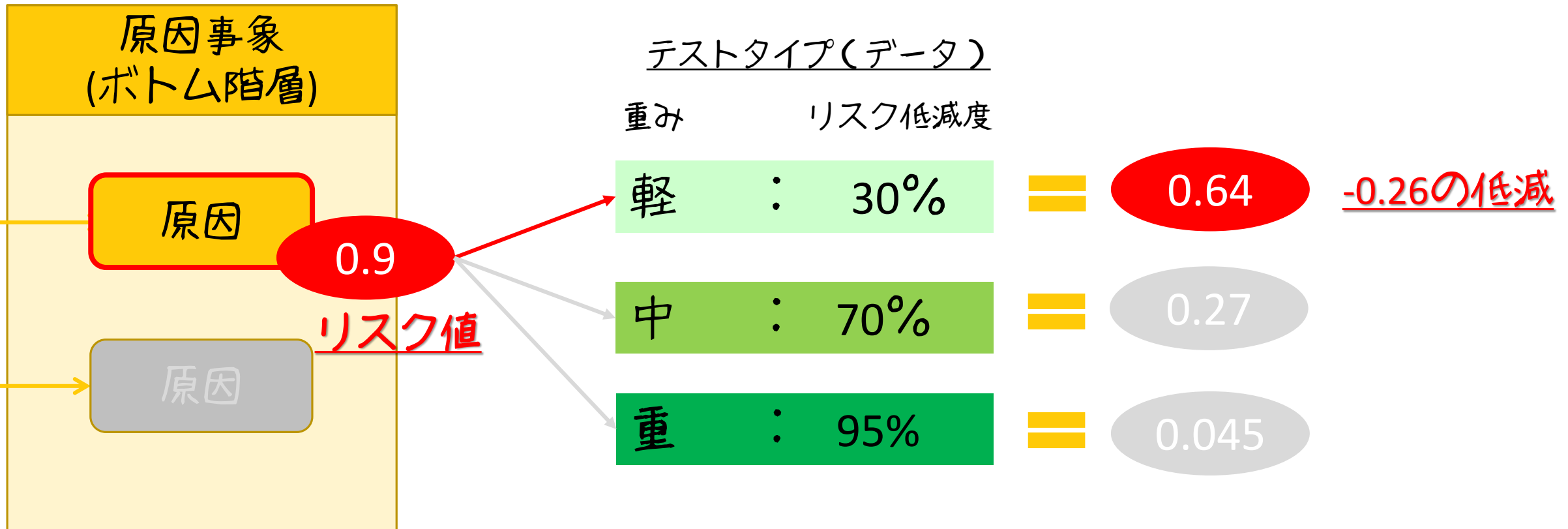
テストタイプ⇔テスト技法対応表

原因からどのようにテストケースを作成するか決めておく。

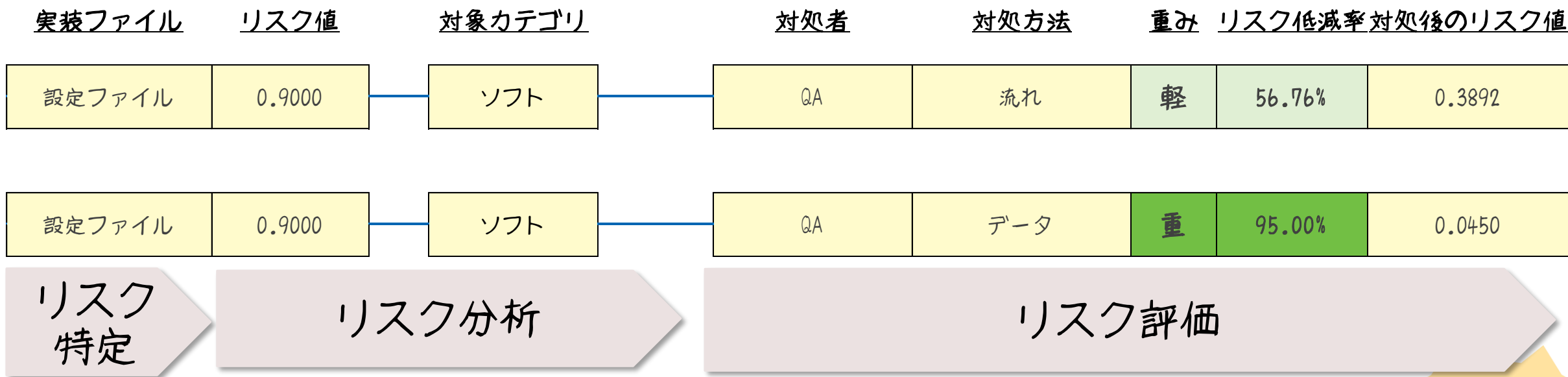
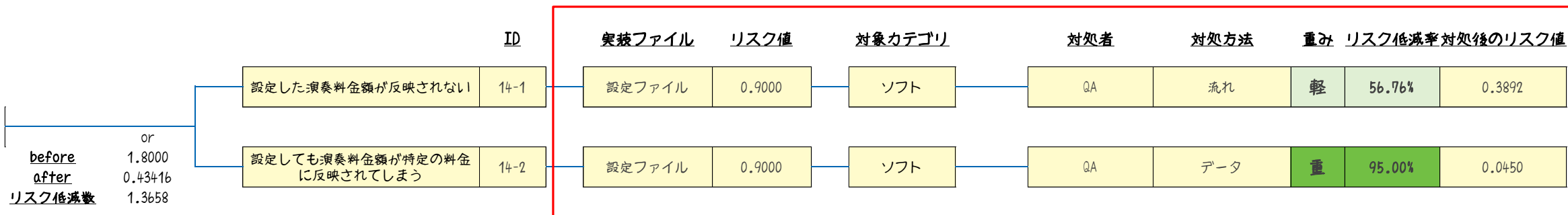
テストタイプ	(キーワード)	テスト技法	(テストケース導出モデル)																				
- データ (領域、境界、サイズ) - 性能	テストタイプ+ 抜け/漏れ ズレ/跳び 過剰/不足 矛盾/衝突 未反映/初期値/なし	同値分割 境界値分析	例 																				
流れ (シナリオ、処理)	同上	フローチャート	例 																				
イベント/タイミング (状態、モード、遷移)	同上	状態遷移図/表	例 																				
構成 (環境、機器、モジュール)	同上	分類ツリー法	例 																				
条件 (入出力、制約)	同上	デシジョンテーブル	例 <table border="1" data-bbox="1755 1253 2150 1389"><thead><tr><th></th><th>1</th><th>2</th><th>3</th><th>4</th></tr></thead><tbody><tr><td>A</td><td>F</td><td>F</td><td>T</td><td>T</td></tr><tr><td>B</td><td>F</td><td>T</td><td>F</td><td>T</td></tr><tr><td>Result</td><td></td><td>○</td><td>○</td><td>○</td></tr></tbody></table>		1	2	3	4	A	F	F	T	T	B	F	T	F	T	Result		○	○	○
	1	2	3	4																			
A	F	F	T	T																			
B	F	T	F	T																			
Result		○	○	○																			

対処する事象(フォールト)の選定と対処方法割り当て

テストタイプの重みを割り当て、事象全体の「やばさ」を下げる。



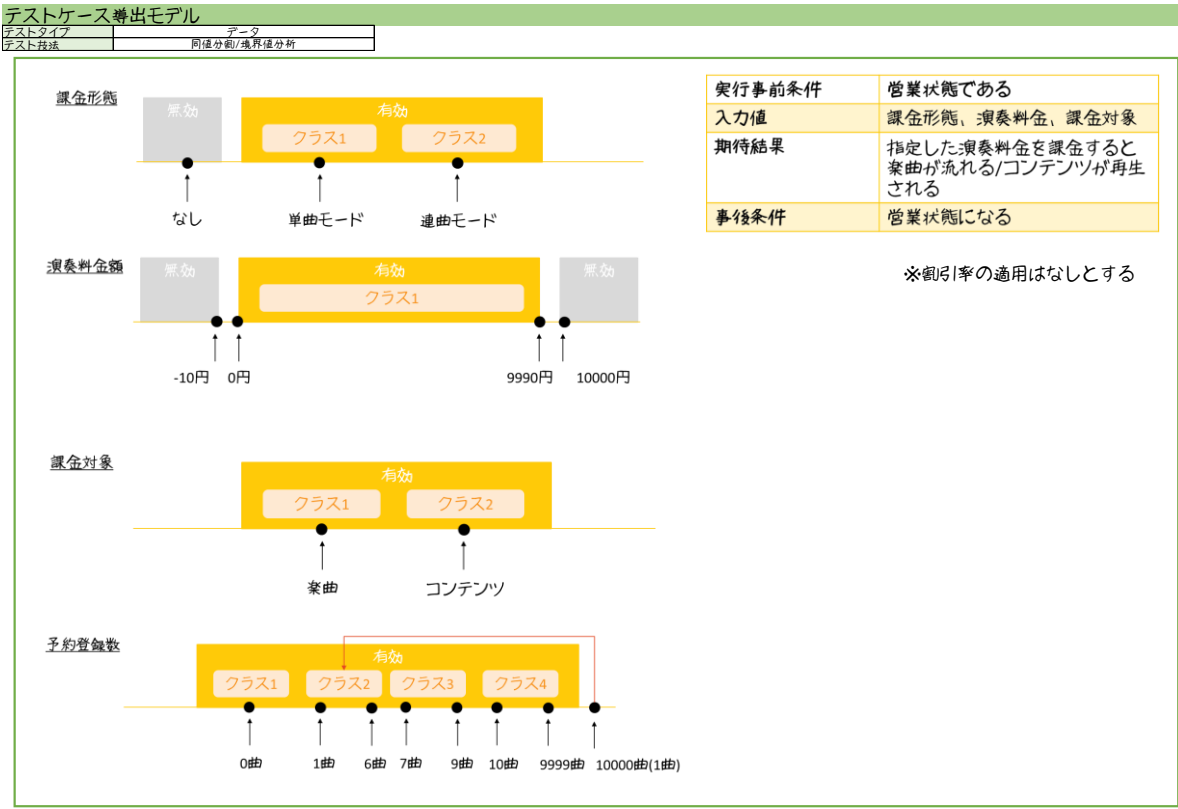
フォールトツリー図サンプル



テストケースサンプル

テストケース導出モデルからテストケースを作成する。

テストケース導出モデル



テストケース

テストケース						
重み		重み				
カバレッジ		有効/無効+同値クラス+境界値				
実装テストケース数		168				
テストケースNo.	入力条件(事前状態)			期待結果	事後条件	事象ID
	課金形態	演奏料金額	課金対象	予約登録数		
1	連曲モード	-10円	コンテンツ	1曲	設定できない	営業状態になる
2	単曲モード	-10円	コンテンツ	1曲	設定できない	同上
3	連曲モード	-10円	コンテンツ	10000曲	設定できない	同上
4	なし	10000円	楽曲	9999曲	設定できない	同上
5	連曲モード	10000円	コンテンツ	10000曲	設定できない	同上
6	なし	0円	コンテンツ	6曲	課金されない	同上
7	単曲モード	10000円	楽曲	9曲	設定できない	同上
8	単曲モード	0円	コンテンツ	7曲	指定金額で課金される	同上
9	単曲モード	0円	楽曲	6曲	指定金額で課金される	同上
10	連曲モード	0円	コンテンツ	7曲	指定金額で課金される	同上
11	単曲モード	9990円	楽曲	0曲	指定金額で課金される	同上
12	連曲モード	9990円	コンテンツ	10000曲	指定金額で課金される	同上
13	なし	9990円	楽曲	6曲	課金されない	同上
14	単曲モード	0円	コンテンツ	9999曲	指定金額で課金される	同上
15	なし	-10円	コンテンツ	6曲	設定できない	同上
16	単曲モード	10000円	楽曲	0曲	設定できない	同上
17	単曲モード	10000円	コンテンツ	9999曲	設定できない	同上
18	単曲モード	-10円	コンテンツ	0曲	設定できない	同上
19	単曲モード	9990円	コンテンツ	10000曲	指定金額で課金される	同上
20	なし	0円	コンテンツ	7曲	課金されない	同上
21	単曲モード	-10円	楽曲	9999曲	設定できない	同上
22	なし	-10円	コンテンツ	10000曲	設定できない	同上
23	連曲モード	0円	楽曲	6曲	指定金額で課金される	同上
24	連曲モード	-10円	コンテンツ	9曲	設定できない	同上
25	単曲モード	0円	楽曲	1曲	指定金額で課金される	同上
26	なし	9990円	コンテンツ	1曲	課金されない	同上
27	単曲モード	9990円	コンテンツ	6曲	指定金額で課金される	同上
28	なし	0円	楽曲	10000曲	課金されない	同上
29	なし	9990円	楽曲	7曲	課金されない	同上

リスク・テスト全体を俯瞰

「やばさ」に対して、許容範囲になっているか、何をテストするかを俯瞰する。

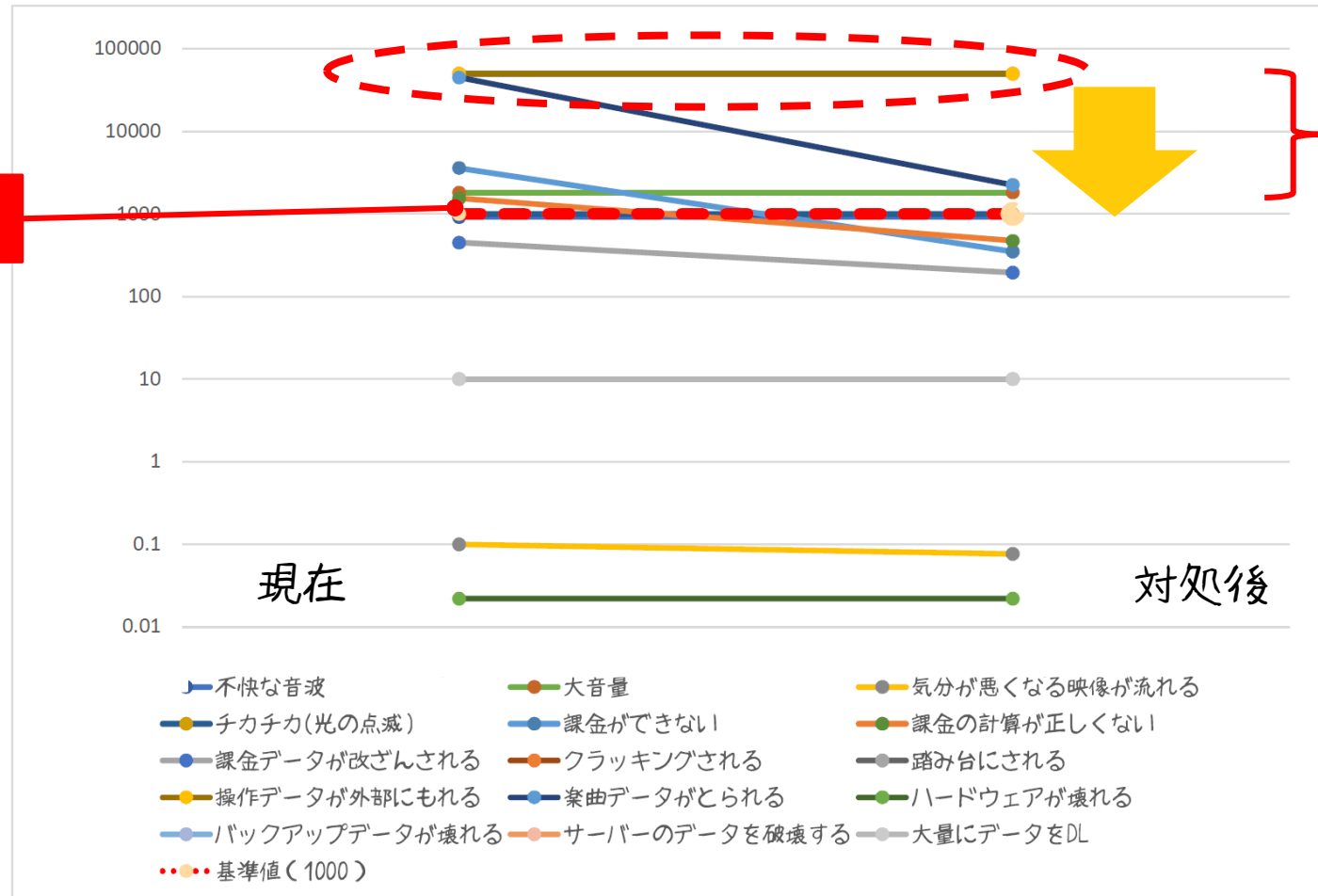
テスト俯瞰表

				やばさ						テストタイプ					
カテゴリ	リスクカテゴリ	No.	フォールト項目	Rank	現在	⇒	Rank		低減数	低減率	データ	流れ	条件	イベント/タイミング	構成
顕在化しにくい	身体的リスク	1	不快な音波	9	917.8	⇒	7	917.8	0	0.00%	—	—	—	—	—
		2	大音量	6	1809.32	⇒	5	1809.32	0	0.00%	—	—	—	—	—
		3	気分が悪くなる映像が流れる	14	0.10	⇒	14	0.10	0	0.00%	—	—	—	—	—
		4	チカチカ(光の点滅)	8	1000	⇒	6	1000	0	0.00%	—	—	—	—	—
	金銭的リスク(課金)	14	課金ができない	5	3600.00	⇒	9	352.08	3247.92	90.22%	○	○	○	○	—
		15	課金の計算が正しくない	7	1544.58	⇒	8	472.5	1072.08	69.41%	—	—	—	○	○
		16	課金データが改ざんされる	10	450.00	⇒	10	194.58	255.42	56.76%	—	○	—	—	—
	社会的要請	17	クラッキングされる	1	50000	⇒	1	50000	0	0.00%	—	—	—	—	—
		18	踏み台にされる	1	50000	⇒	1	50000	0	0.00%	—	—	—	—	—
		19	操作データが外部にもれる	1	50000	⇒	1	50000	0	0.00%	—	—	—	—	—
		20	楽曲データがとられる	4	45000	⇒	4	2250	42750	95.00%	—	—	—	—	—
	物理的破壊リスク	21	ハードウェアが壊れる	15	0.022	⇒	15	0.022	0	0.00%	—	—	—	—	—
22		バックアップデータが壊れる	11	10	⇒	11	10	0	0.00%	—	—	—	—	—	
23		サーバーのデータを破壊する	11	10	⇒	11	10	0	0.00%	—	—	—	—	—	
24		大量にデータをDL	11	10	⇒	11	10	0	0.00%	—	—	—	—	—	
顕在化しやすい	うまく使えないリスク	5	ハウリング	—	—	⇒	—	—	—	—	—	—	—	—	—
		6	音が出ない	—	—	⇒	—	—	—	—	—	—	—	—	—
		7	映像が出ない	—	—	⇒	—	—	—	—	—	—	—	—	—
		8	字幕がずれる	—	—	⇒	—	—	—	—	—	—	—	—	—
		9	途中で止まる	—	—	⇒	—	—	—	—	—	—	—	—	—
		10	テンポが変わる	—	—	⇒	—	—	—	—	—	—	—	—	—
		11	字幕が切れる	—	—	⇒	—	—	—	—	—	—	—	—	—
		12	となりの音が割り込む	—	—	⇒	—	—	—	—	—	—	—	—	—
		13	起動しない	—	—	⇒	—	—	—	—	—	—	—	—	—
					204351.82	⇒		157026.4	47325.4	23.16%					

リスク・テスト全体を俯瞰

「やばさ」がどうなるかをそのリリース毎に確認していく。

基準値:1000



対応必須なもの

実現できそうか？

テスト開発コンセプト

「製品のリリースを止めるバグを許容範囲まで減らす！」

課題

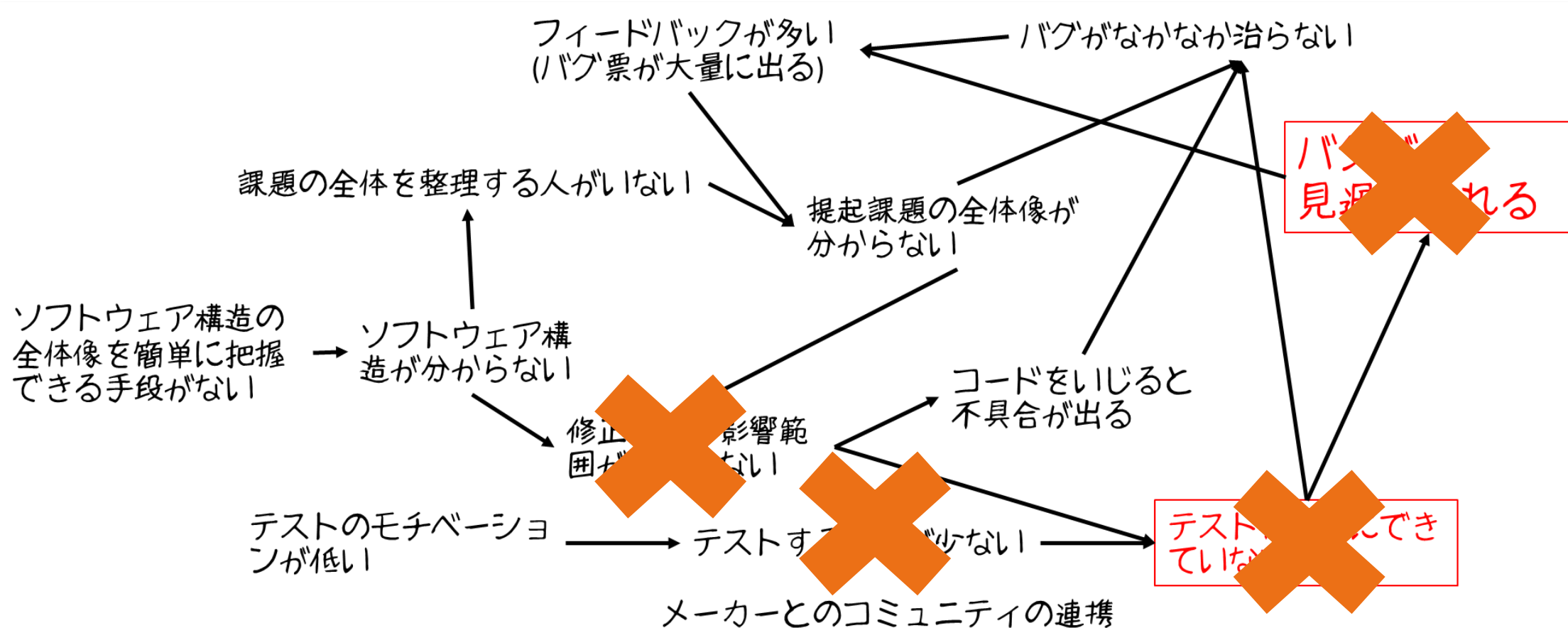
1. 製品のリリースを止める事象およびバグをどう特定するか？
⇒フォールトツリー図を用いた事象の分解
2. テストで対処する/しない事象はどのように決めるか？
⇒被害金額とコード変更履歴からのリスク値による優先順位
3. 対処するためにどんなテストをどれくらい行うか？
⇒テストタイプのリスク低減表による事象の許容範囲への調節

実現できそうか？

テストする人員が少なくても、
修正箇所から影響範囲を見極めて
テストを十分できる

テスト開発コンセプト

「製品のリリースを止めるバグを許容範囲まで減らす！」



開発したテスト設計

納得できている/できていないところが今後の課題。

◆ 納得できていること

- 文脈や場面状況にあったテスト設計になった
- 実践できるテスト技術を習得・開発して適用できた

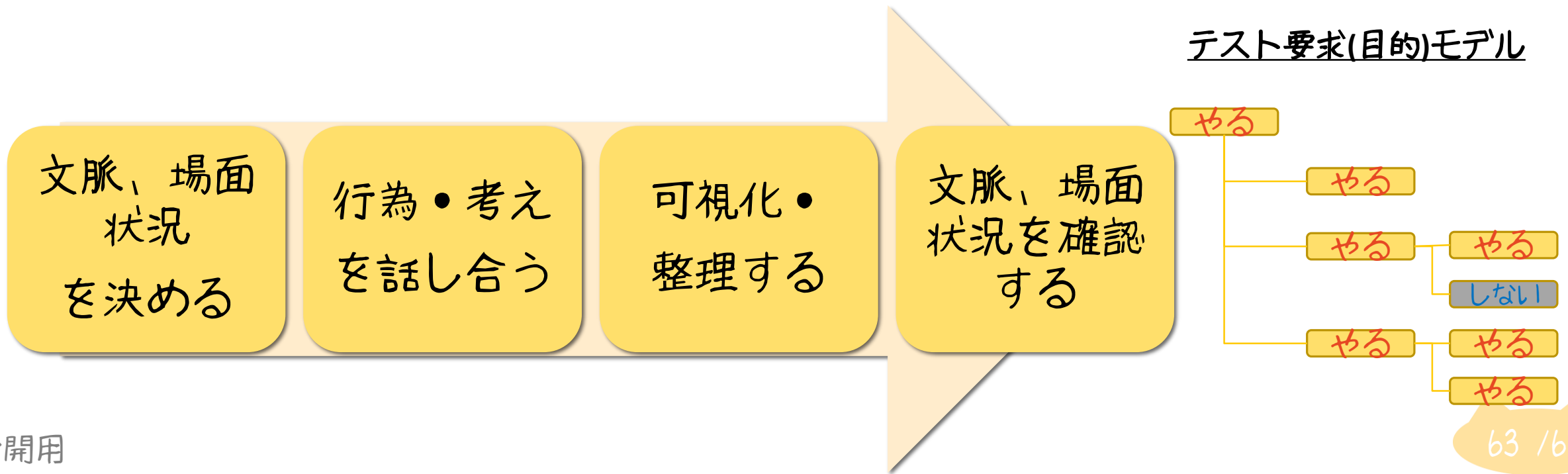
◆ 納得できていない(未消化な)こと

- テスト設計コンテスト審査員からのフィードバック
 - 数値化したやばさ、被害金額の妥当性
 - テストケースを十分性
- 文脈や場面状況が同じ、実テーマへの実践とフィードバック
- やりきれない場面が出てくるはず…その際の課題と解決方法の検討

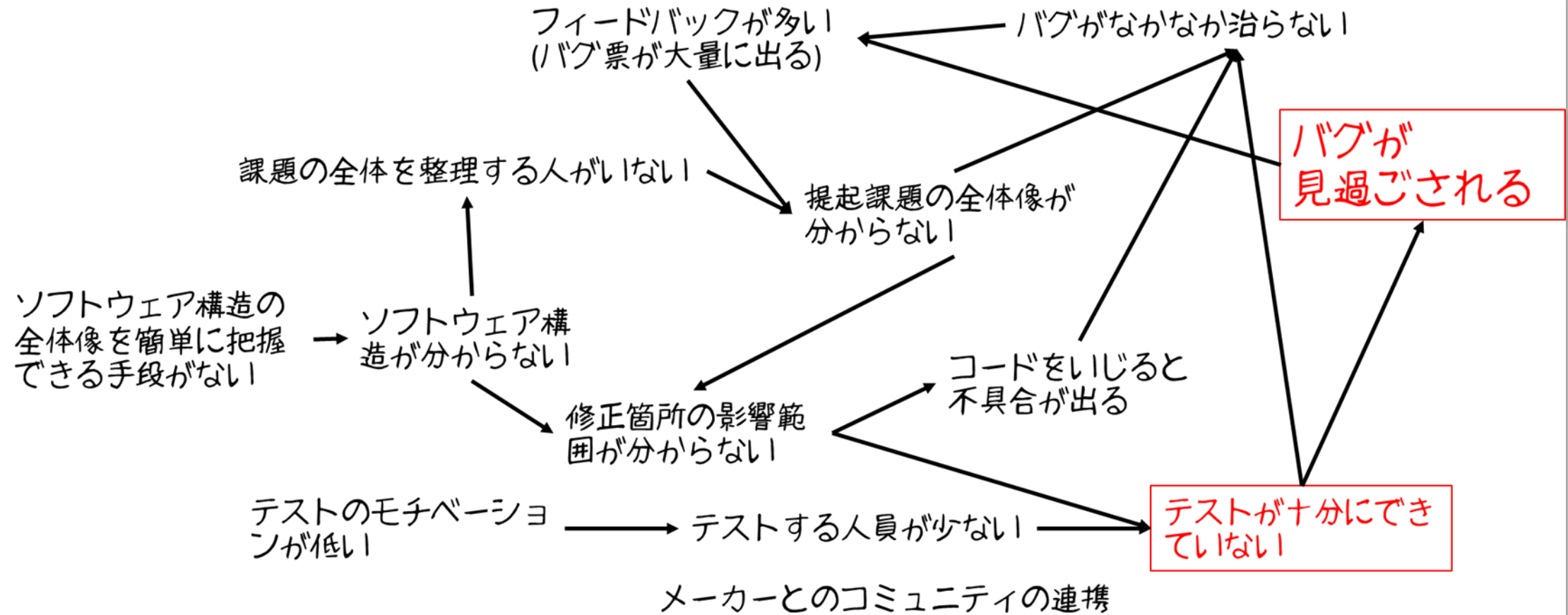
まとめ

◆ 納得するために

- 関連する他人の考えや行為を観察範囲を増やし、整理する
- 整理したものを通して、関連する他者と議論する
- 納得できている/できていないものを明確にして取り組む



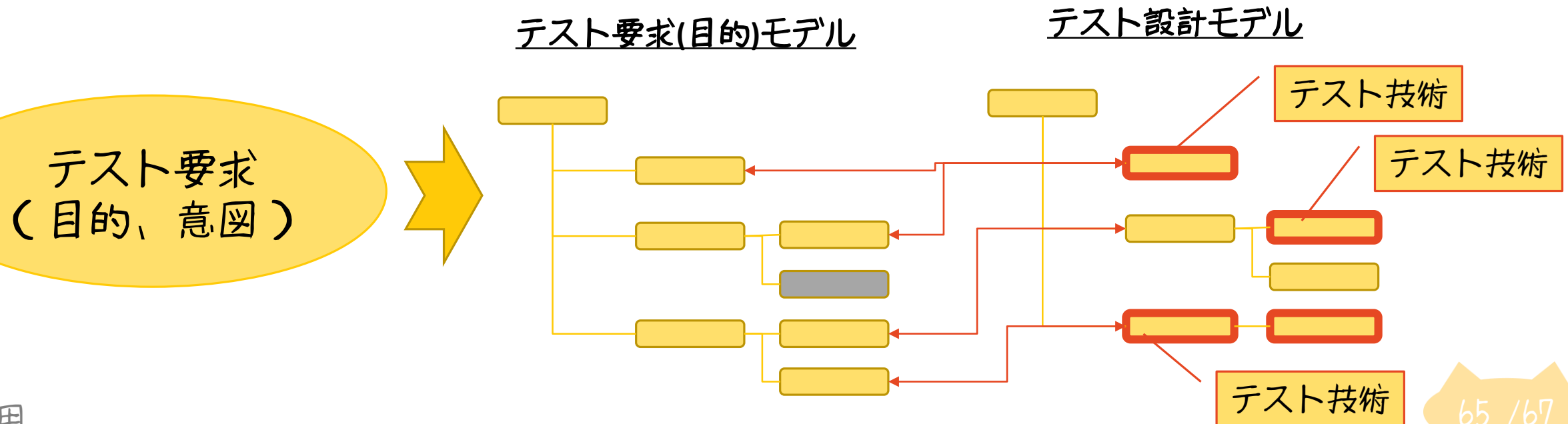
まとめ



自分がやりやすいやり方でやってみる

まとめ

- ◆ ワンステップするために
 - 関係する自分以外の押さえるべき考えや行為をないがしろにしない
 - 考えや行為の観察範囲を整理して成長、洗練させていく
 - 合わせてテスト技術を取り込んでいく（無暗に使わない）



まとめ

- ◆ ワンステップするために
 - 関係する自分以外の押
 - 考えや行為の観察範囲
 - 合わせてテスト技術を

テスト要求
(目的、意図)



テスト



```
~/2_code/pykaraoke$ git bugspots .
Scanning . repo
Found 52 bugfix commits, with 34 hotspots:

Fixes:
- Fix window position to top-left for new player windows.
- * Fixed colour cycling in the CDG player
- v0.4 written by Will Ferrell.
- Fix cosmetic typo in "may be corrupt" message
- - Request hardware-accelerated and double-buffered displays fr
om SDL when
```

ファイル(F) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)

Hotspots:

```
0.0133 - pykdb.py
0.0093 - pykaraoke.py
0.0077 - ChangeLog
0.0024 - pykar.py
0.0019 - pycdg.py
0.0018 - performer_prompt.py
0.0010 - pympg.py
0.0009 - _pycdgAux.c
0.0009 - setup.py
0.0006 - pykmanager.py
0.0003 - .cvsignore
0.0002 - README.txt
0.0002 - pykplayer.py
0.0002 - pykaraoke_mini.py
0.0001 - _cpuctrl.c
```

Bugspots (Google のバグ予測アルゴリズム)

一つずつ試していく

まとめ

◆納得するために

- 関連する他者の考えや行為を観察範囲を増やし、整理する
- 整理したものを通して、関連する他者と議論する
- 納得できている/できていないものを明確にして取り組む

◆ワンステップするために

- 関係する自分以外の押さえるべき考えや行為をないがしろにしない
- 考えや行為の観察範囲を整理して成長、洗練させていく
- 合わせてテスト技術を取り込んでいく（無暗に使わない）

”テストの目的や意図”を理解し、押さえる方法確立して
自分なりのテストを開発して、育てていきましょう！

ご清聴
ありがとうございました



参考資料

1. ASTER(2017) 『テスト設計コンテスト'18 チュートリアル資料 Open版』 ASTER.
2. International Software Testing Qualifications Board 用語集作業班(2014) 『ソフトウェアテスト標準用語集(日本語版)Version 2.3.J01』 Japan Software Testing Qualifications Board 技術委員会.
3. JaSST Kansai実行委員会(2016) 『『ソフトウェアテストの価値って?』 ～ てすにゃんくえすと～』 <http://jasst.jp/symposium/jasst16kansai/pdf/S4-1.pdf> (2018/06/11 アクセス)
4. Igrigorik (2011) 『Bugspots - Bug Prediction Heuristic』 <https://github.com/igrigorik/bugspots> (2018/06/11 アクセス)