三銃士モデルを
現場適用してみた！

目次

- はじめに

- 弊社でのテストの課題①
- 課題解決への道①
  - 「フリーデバッグ」実施時期の見直し

- 弊社でのテストの課題②
- 課題解決への道②
  - 三銃士モデルへのチャレンジ

はじめに

はじめに

本資料は、弊社における三銃士モデルの現場適用事例を通じて、
得られた知見・効果について、共有させていただきます。

なお、現場によって状況が異なると思いますので、
適用時には、各現場の方々と十分にコミュニケーションをとって
適用・導入されることをおすすめします。

はじめに　～ 自己紹介 ～

名前　：河内 奈美（かわうち なみ）

あだ名　：お母ちゃん、母、ママ etc

（なぜこんなあだ名かは後述…）

経歴：

- 2012年　～　2016年

  ゲーム主体のテスト専門会社勤務

  → 項目作成、携帯端末でのテストなどを経験。

  → ガラケーからスマホ、そしてブラウザからアプリへと
  主軸が変わっていく激動の時代を生きる。

- 2016年(アカツキ入社) ～　現在にいたる

  → 2017年までテスターとして勤務。(この頃リーダーとして
  皆の尻を叩く姿がお母ちゃんぽいという理由で「検証の母」と呼ばれる)

  → 2018年にPM見習いPMOとして異動。現在に至る。
  (ちなみにこのタイミングで「開発の母」に昇格し、プロジェクト全体の尻叩き役になる)

弊社でのテストの課題①

弊社でのテストの課題①

プロジェクト終盤に・・・
エンジニアレビュー済みのテスト項目も完了し
スケジュールどおりに検証を進め・・・

あとは、**「フリーデバッグ」**をするだけ
というところまで来て・・・

弊社でのテストの課題①

「フリーデバッグ」で重大な不具合を発見し・・・
リリースが伸びることがありました・・・

「リリースが伸びる」という
課題に対して・・・
取り組んだ内容について
ご紹介させていただきます。

弊社でのテストの課題①

弊社での課題解決の取り組みを
ご紹介したいと思いますが・・・

ちょっと、その前に！！

ここまで、何度か出てきた
「フリーデバッグ」という
キーワードについて、
ご説明したいと思います！

フリーデバッグとは？

Akatsuki

突然ですが・・・

みなさんに質問です！

「フリーデバッグ」という

言葉をご存知ですか？

Akatsuki

まず「デバッグ」という

言葉自体、馴染みがない方も

おられるのでないでしょうか？

Akatsuki

そもそも「デバッグ」とは・・・

「de bug」＝「バグを取り除く」であり

プログラマの作業のことですが・・・

ゲーム業界では・・・

「テスト」≒「デバッグ」という意味で

テスターの作業となってる現場が多い

のが現状です！

フリーデバッグとは？

『ゲーム業界において・・・
「フリーデバッグ」という言葉をよく耳にすると
思いますが、「フリーデバッグ」には多くの意味が含まれており、受発注者間でそれぞれ異なる認識で進めている場合があります。

その中でもリリース直前のユーザーテスト・探索的テストをメインに実施されることが多いですが、明確なチェックシートがなく、「お任せ」的な「フリーデバッグ」となることも少なくありません。』

※ 上記は、弊社QAチーム内での説明資料のため、組織によって異なる場合がありますので、ご注意ください

フリーデバッグとは？

フリーテスト（フリーデバッグ）の種類

ISTQB等で定義されている、フリーテストに紐付くテストを以下に列挙します。

- モンキーテスト （ monkey testing ）
- スモークテスト （ smoke test ）
- インテークテスト （ intake test ）
- アドホックテスト （ ad hoc testing ）
- 探索的テスト （ exploratory testing ）
- ユーザーテスト （ user test ）

※ 詳細は、本資料文末のAppendix をご参照ください

課題解決への道①

課題解決への道①

リリース直前に行っていた、**「フリーデバッグ」**
において多くの不具合が発見され・・・
リリース延期が発生していたが・・・

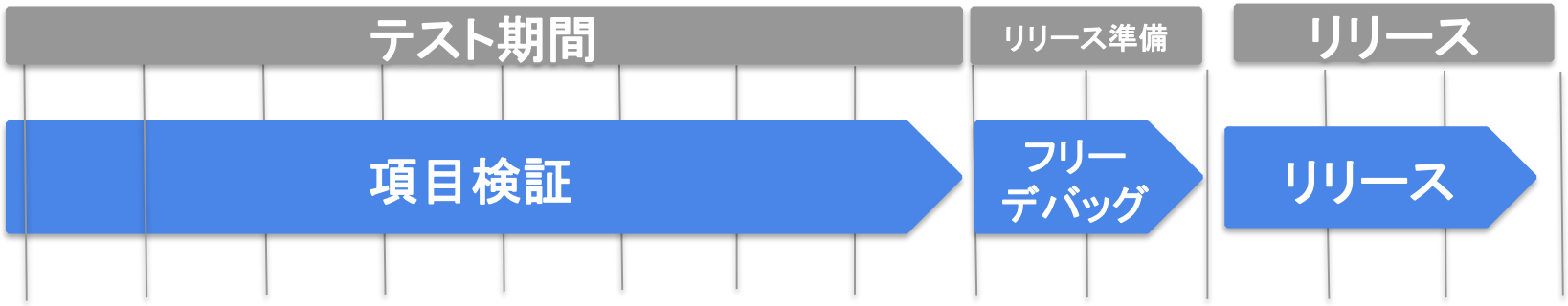
不具合内容から、**「フリーデバッグ」**の有効性が
認められ、実施時期・期間を見直すことで・・・
より安全にリリースできるようになった！

課題解決への道①

課題解決は、主に下記のように3段階で行われた

- 初期
  - 時期：リリース直前の数日
    課題：不具合による、リリース延期
- 中期
  - 時期：リリース前の5営業日（≒1週間）
  - 課題：不具合による、リリース準備の再実施
- 後期
  - 時期：リリース準備前の5営業日（≒1週間）
  - 課題：増員したが、不具合件数は・・・

課題解決への道①　～ 初期 ～



基本的な項目テスト（≒ 機能テスト）が終了した後、

リリースまでのリリース準備期間に品質を高めるために、

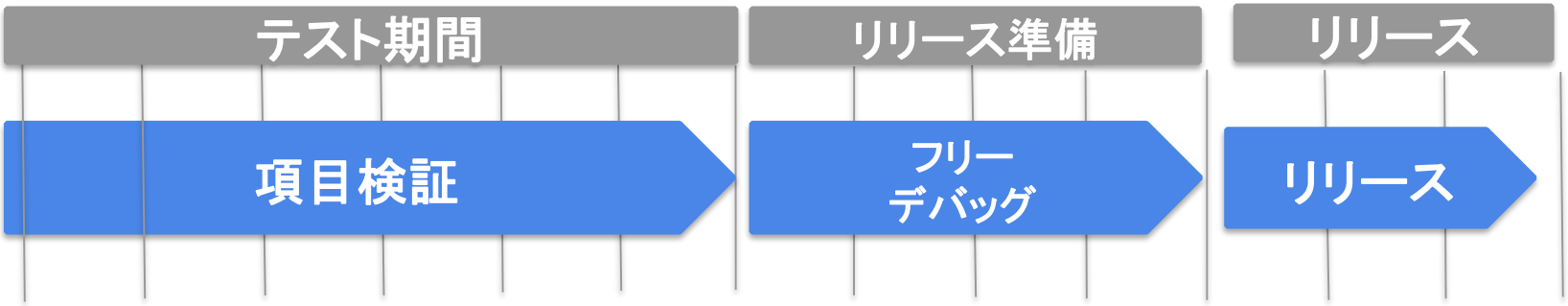
補助的なテストとして「フリーデバッグ」を実施した。

「フリーデバッグ」では、ベテランにより市場インパクトが大きい不具合を事前に発見できたことにより、「フリーデバッグ」期間が重要視されるようになった。

ただし、リリース直前に大きい不具合を発見したために、

★リリース延期が発生した

課題解決への道①　～ 中期 ～

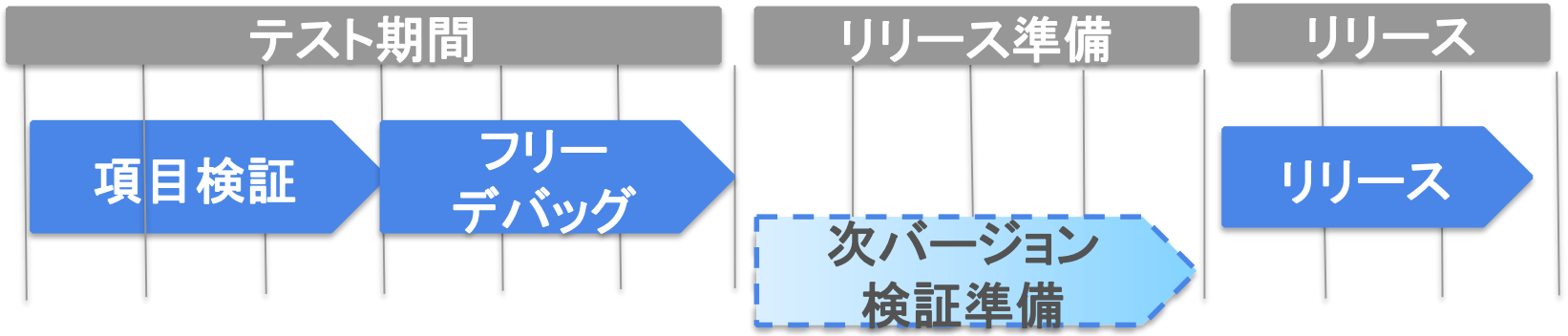


「フリーデバッグ」が重要視されることで、課題解決中期には
「フリーデバッグ」期間が長くなり、重大な不具合が出ても
リリース延期にまでいたらなかったが・・・

★リリース準備の再実施が発生することあった・・・

そんな中、プロダクトの状況は、全体の機能数が増え既存機能
との関連も多く、サービス全体の複雑度が上がり続けた。
そのため基本的な項目テストだけでは複雑な条件での不具合が
見つけられず、「フリーデバッグ」の重要性が高まっていった。

課題解決への道①　～ 後期 ～



課題解決後期になると「フリーデバッグ」期間をリリース準備前に実施することで、リリース準備での手戻りも削減された。

だがしかし・・・「フリーデバッグ」期間が重要視される中・・・
テスターを増員したにもかかわらず・・・

発見される不具合は、いままでどおり、ほとんどベテランテスターによるものだったため、増員した割には効果が発揮できなかった

課題解決への道①　～ 後期 ～

課題解決を進める上で、「期間」「時期」「増員」だけでは解決できない下記のような課題があることに気がついた。

我々は、ベテランテスターがどのような「フリーデバッグ」を実施しているか解明するために、発見された「不具合を分析」することにした・・・・

発見される不具合は、いままでどおり、ほとんどベテランテスターによるものだったため、増員した割には効果が発揮できなかった

「フリーデバッグ」の
不具合を分析してみた

「フリーデバッグ」の不具合を分析してみた

なぜ「フリーデバッグ」まで不具合が見つからなかったのか？
原因を紐解いてみた・・・

- 機能テストでの期待結果の不明確だったために・・・
- 機能の組み合わせが、不十分だったために・・・
- 上流工程で想定していない動作を実施したために・・・

「フリーデバッグ」の不具合を分析してみた

Akatsuki

なぜ「フリーデバッグ」まで不具合が見つからなかったのか？

原因を紐解いてみた・・・

- **機能テストでの期待結果の不明確だったために・・・**

テスト手順書作成（実装）に課題あり

機能テストのテスト項目を調べたところ・・・

期待結果に「表示が正しいことを確認」となっていたため
テスト実施者が表示内容を確認した際に、画像サイズ・見切れ・ノイズ等もなかったために、「正しい」と判断してしまっていた

実際には、古い施策の画像が使われてしまっていたために、
前回の施策を知っていた、ベテランがフリーでバッグで不具合を発見することができた

「フリーデバッグ」の不具合を分析してみた

Akatsuki

なぜ「フリーデバッグ」まで不具合が見つからなかったのか？

原因を紐解いてみた・・・

- 機能テストでの期待結果の不明確だったために・・・

テスト手順書作成（実装）に課題あり

- **機能の組み合わせが、不十分だったために・・・**

テスト観点組合せ（設計）に課題あり

-

機能テストのテスト項目を調べたところ・・・

追加された、単一機能のテストは特定の因子・水準でのみ実施されていた

特定の因子・水準でのみ、計算が間違っていた
既存機能に詳しい、ベテランが複数の機能・因子・水準の組み合わせで実施して、不具合を発見した

「フリーデバッグ」の不具合を分析してみた

Akatsuki

なぜ「フリーデバッグ」まで不具合が見つからなかったのか？

原因を紐解いてみた・・・

- 機能テストでの期待結果の不明確だったために・・・

テスト手順書作成（実装）に課題あり

- 機能の組み合わせが、不十分だったために・・・

テスト観点組合せ（設計）に課題あり

- 上流工程で想定していない動作を実施したために・・・

テスト観点抽出に課題あり

機能テストのテスト項目を調べたところ・・・

対象画面のテストは実施していたが・・・
特定のルートを通ると不具合が発生した
メインパスでなかったために、多くのテスターが気づかなかった

Akatsuki

まとめると大きな課題は3つ！

- テスト手順書作成（実装）
- テスト観点組合せ（設計）
- テスト観点抽出

課題解決への道②

課題解決への道②

分析結果から判明した課題の中で

・テスト手順書作成（実装）

については、新規メンバーでもわかるように
期待値を記述することを徹底することで、
改善のめどがたちました！！

Akatsuki

- テスト観点組合せ（設計）
- テスト観点抽出

上記2点について・・・

テスト設計において課題解決し、機能テストで品質担保できれば、属人性を軽減できると考えていましたが・・・良い手が思い浮かばないまま・・・

私は、**ベテランQAメンバー**に相談してみました・・・

課題解決への道②　～ 相談編 ～

Akatsuki

河内

ベテラン

(;・・)「テスト項目の観点漏れを手っ取り早く減らすには
ベテランに項目作成を全部お願いするしかない」

「負担がでかいよ。」Σ(￣Д￣;

(;;・・)「じゃあベテランレベルまで項目作成できるように
メンバーを教育するしかない！」

「教育までしてる余裕ないんだけど」
Σ(￣Д￣;;

(;;;・・)「じゃあどういう考えで項目作ってるか
教えてください！！」

「え、仕様書見ればわかるじゃん」(￣ー￣)

Akatsuki

(ノToT)ノ┤ :･’…･┻┻:･’…･

「それが出来たら
相談してないよぉ〜!」

Akatsuki

人材育成が先か？
教育方法が先か？

「にわとり」と「たまご」
のような、状態に・・・

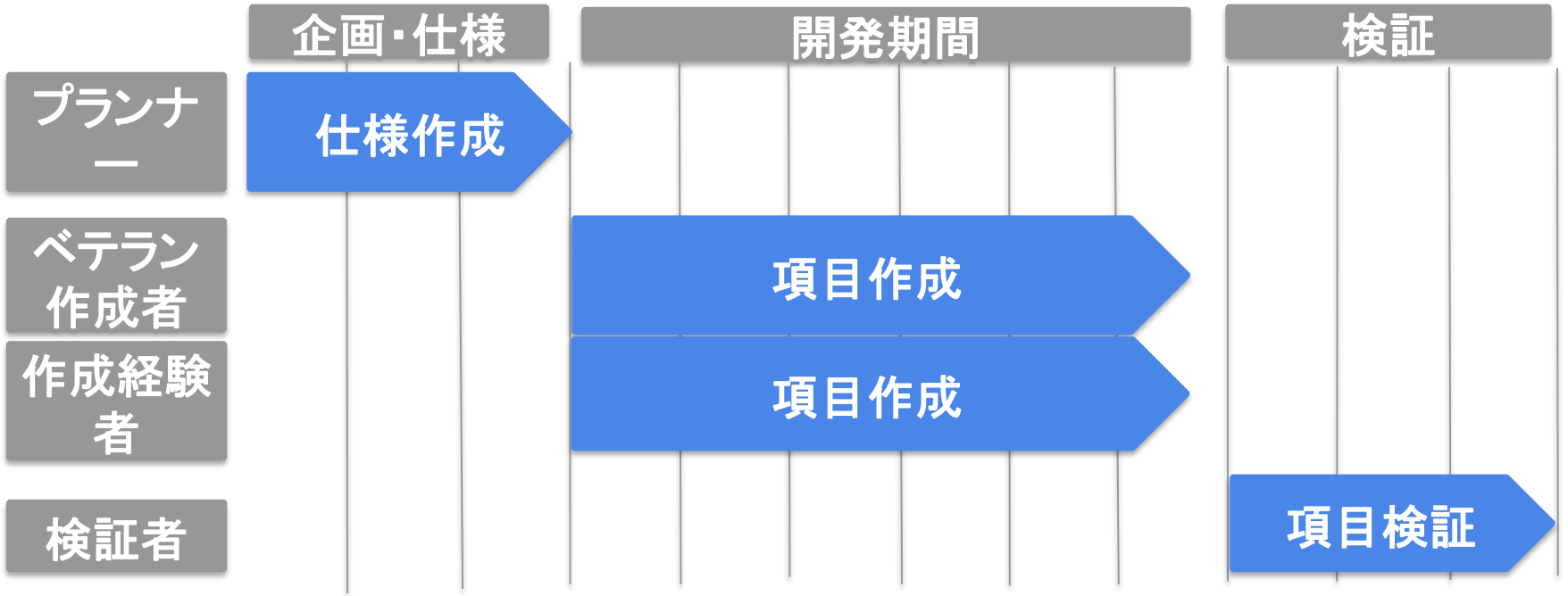
そんなときに・・・
QAチームの有識者から
救いの一言が！！。

(๑･｀д･´)✧+°
「マインドマップでベテランの
考え方を整理してみたら？」

Akatsuki

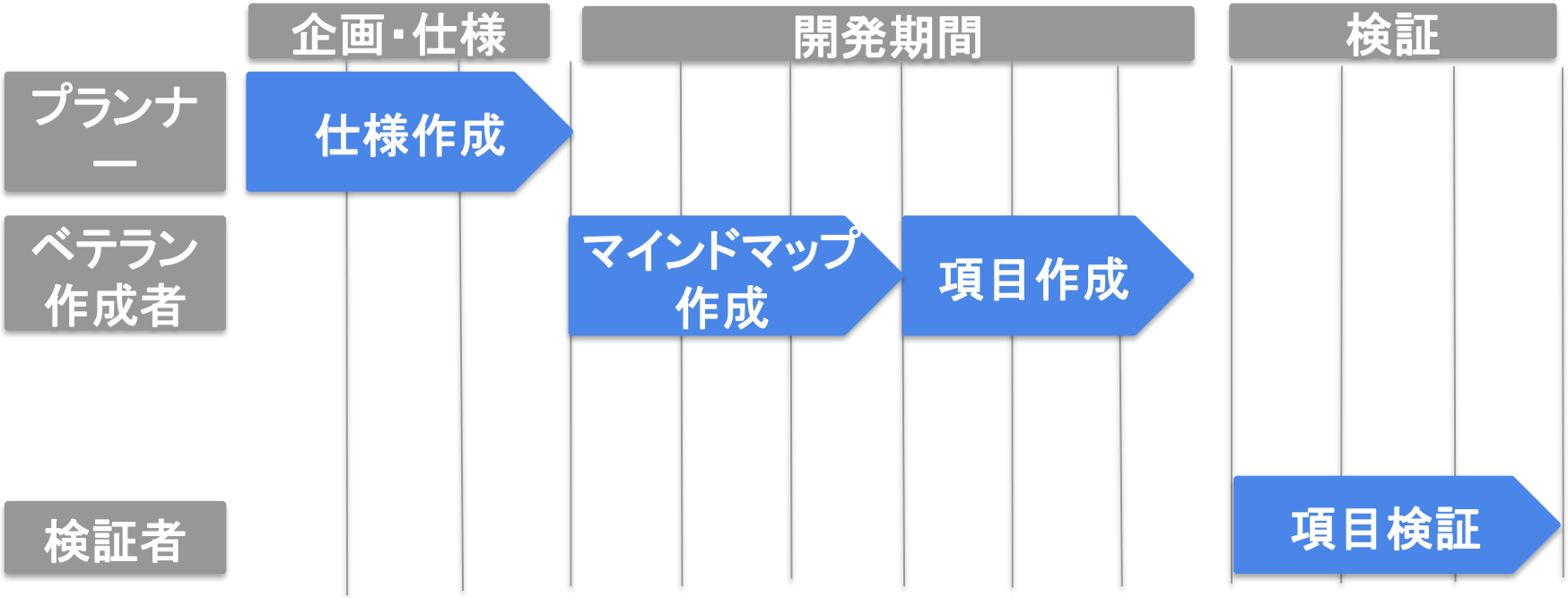
マインドマップを使ってテスト設計をやってみたら、「可視化」される要素で観点が変わった現場の話

【テスト設計が浸透するまでの流れ】フェーズ0



先ほど説明した課題時の段階での図解。
・仕様書から直接テスト項目を作成。（項目の粒度は作成者の力量に左右される）
・多少は項目レビューが実施されていたが膨大な項目に対しては効果的には至らず。

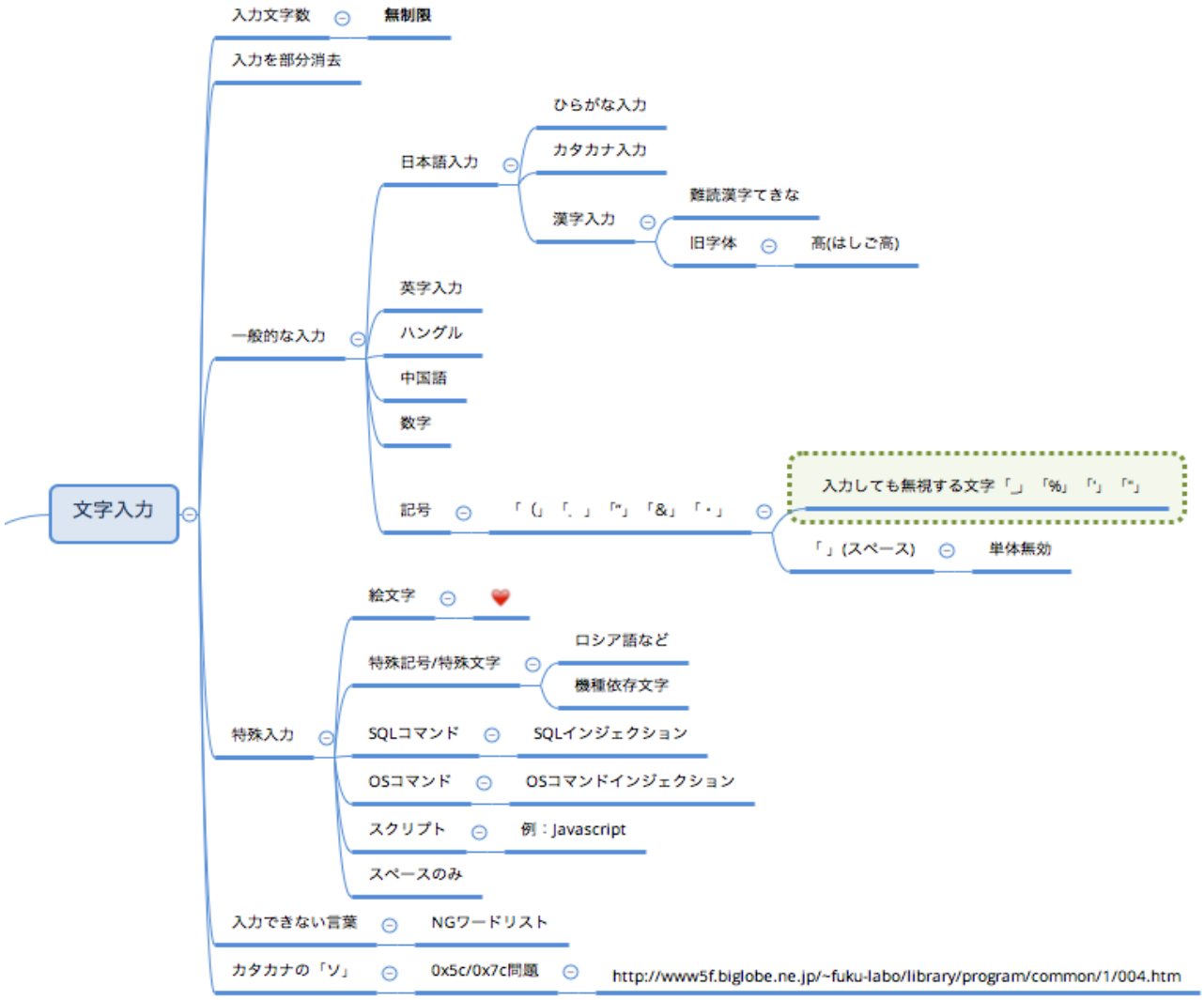
【テスト設計が浸透するまでの流れ】フェーズ1



ベテラン項目作成者が、まずテスト観点を
マインドマップ化し、それを元に項目作成を行う。

※これまでの通常作成経験者は一旦未経験者
として扱うことにしたため空白。

実際のマインドマップ

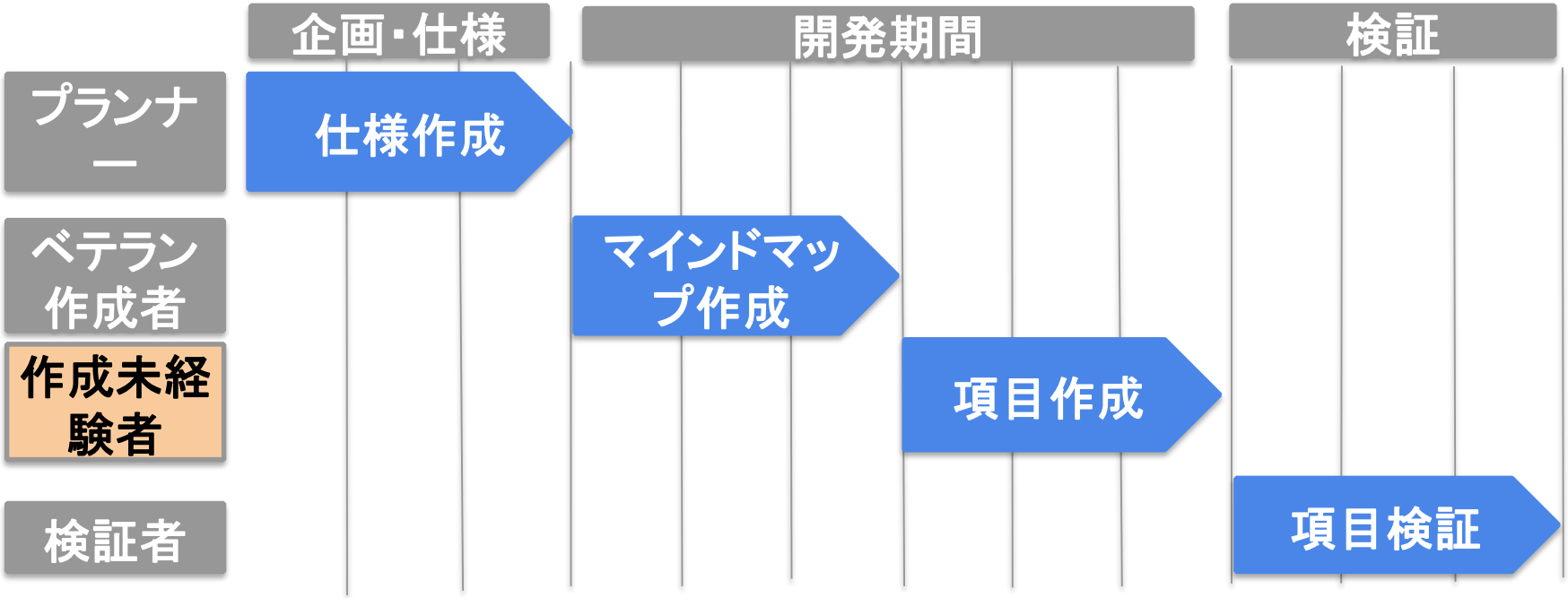


マインドマップから項目化したもの

	事前条件1	事前条件2	操作	カテゴリ	実行結果/期待結果
テキスト入力挙動	入力文字数		しばらく文字を入力する	動作	文字数無制限で文字が入力できる
			入力を消去する	動作	入力した文字が消去できる
	入力できる文字	日本語入力	ひらがな・カタカナを入力	動作	ひらがな・カタカナが入力できる
			漢字を入力	動作	漢字入力ができる(人造人間17号など)
		英数字入力	英字を入力	動作	英文字入力ができる
			数字を入力	動作	数字入力ができる
		特殊入力	記号入力	動作	記号入力ができる「(」「・」「&」「"」
			環境依存文字	動作	環境依存文字が入力できる「①」「ﾝ」「㍻」など
			特殊文字	動作	特殊文字が入力できる「Д」など
				動作	絵文字入力ができない
		0x5c/0x7c問題	「ソ」を入力	動作	「ソ」が入力できる

	SQLインジェクション対策	事前準備(破壊対象の用意)	下記クエリを入力↓ INSERT INTO [redacted] '2018/09/01', '2018/09/01');	マスタ	[redacted]テーブルのidが [redacted] のデータがある
		破壊コマンド入力	下記を検索に入力 '; DELETE FROM [redacted]	動作	検索結果が表示されない
			コマンド入力後に確認	マスタ	[redacted]テーブルのidが [redacted] のデータがある

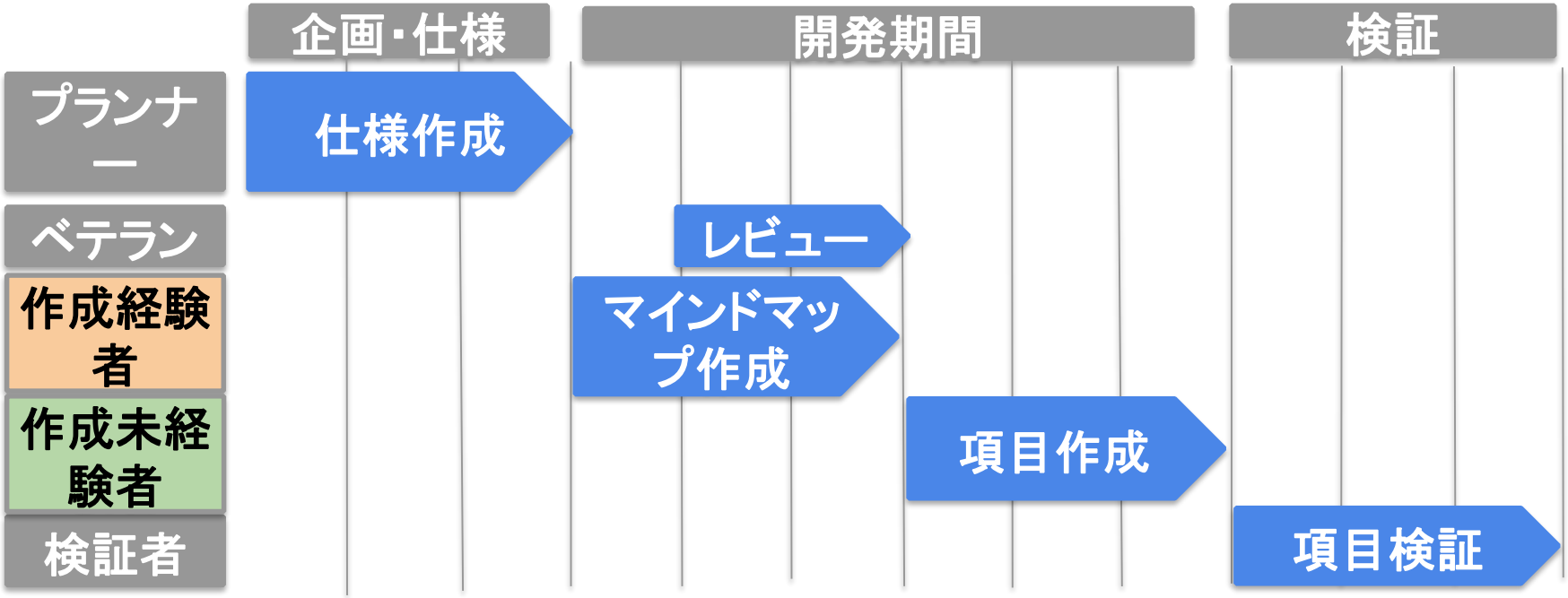
【テスト設計が浸透するまでの流れ】フェーズ2



熟練者が作成したマインドマップを、項目作成未経験者がそれを元に項目作成を行う。

・項目作成（文章に起こす作業）をベテランが行わないことで、観点出しに集中することに成功。

【テスト設計が浸透するまでの流れ】フェーズ3



ある程度の項目作成経験者がマインドマップを作成し、熟練者がマインドマップレビューを行う。

・テストの観点を実践的に養える場を設定することに成功。

このフェーズ1～3を繰り返すことで、
メンバーの育成、及び
スキルの底上げを促進することに
成功しました。

導入後の現場の声

- マインドマップで細分化していくことで、
  パターンが洗い出しやすい。
  仕様の観点もれも指摘しやすい。
- 「いきなり項目作ってくれ」って言われるより、
  マップから文章を起こすだけなのでやりやすい。
- マインドマップの他機能への再利用が可能に
  なった。

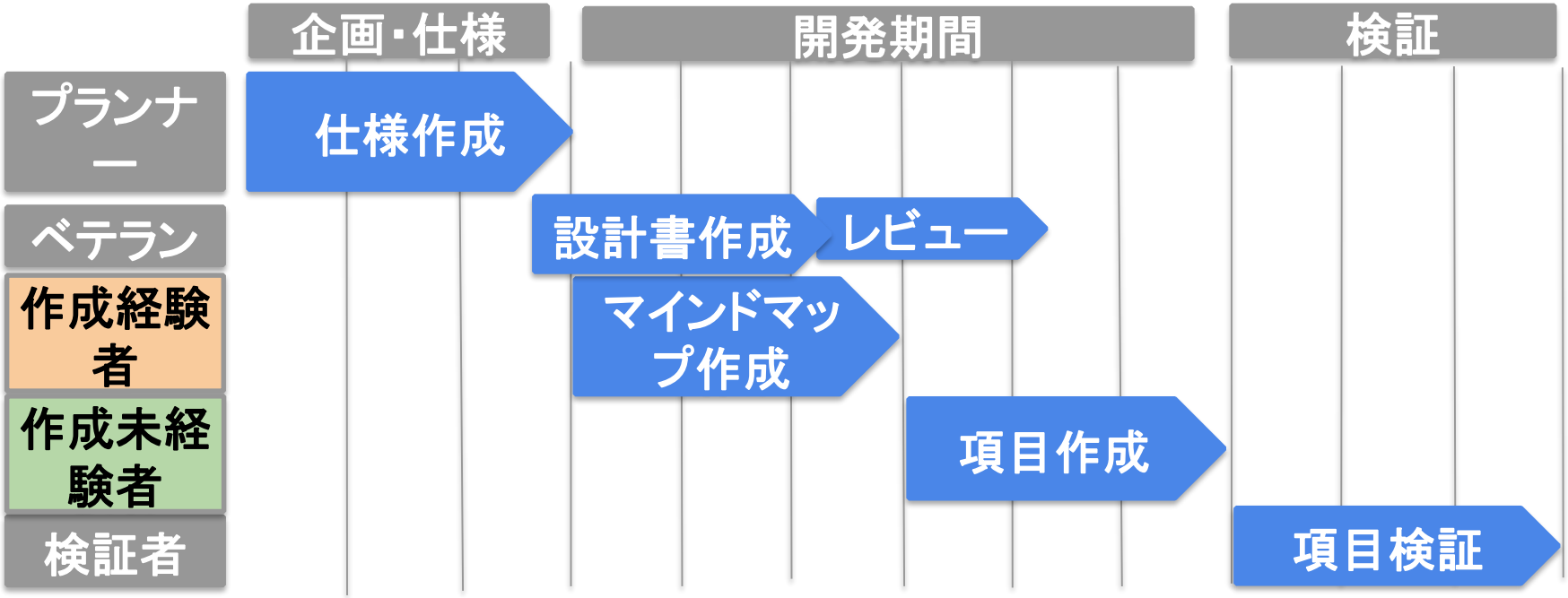
caution！！！！

現場適用、人材教育時、
マインドマップ初心者には身近なもの
（弊社では「自己紹介」から行いました）から
慣れさせることをおすすめします。

弊社ではマインドマップ自体の意図（頭の中を整理する、構造化する）が伝わらないままツールを使ってしまい、上手く教育に繋がらなかった過去がございます。。。。

今後の取り組み

【テスト設計が浸透するまでの流れ】フェーズ4



項目作成者に口頭で伝えていた注意点や重点確認箇所等をテスト設計書として作成し、マインドマップとともに参照してもらうようにしたい
と思っています。

次への課題

「フリーデバッグ」から、
機能的観点を出し分けることができた。
次はゲームにおける「ユーザー感情」を
三銃士モデルを活用して
テスト観点に導入していきたい
と思っています。

ご静聴
ありがとう
ございました。

m(_*_)m

Appendix

そもそも「フリーデバッグ」とは何か　～ 詳細 ～

そもそも「フリーデバッグ」とは何か ～ 詳細 ～

Akatsuki

- **モンキーテスト（ monkey testing ）:**
  製品の使用法については全く考慮せず、広範囲の入力からランダムに選択し、ボタンをランダムに押すことでテストを行なう方法、ランダム打鍵テストとも呼ばれる

そもそも「フリーデバッグ」とは何か ～ 詳細 ～

Akatsuki

- **スモークテスト(smoke test) :**
  定義・計画した全テストケースのサブセットを用いて実施プログラムの必須機能が正常に動作することを確認するのが目的で、コンポーネントやシステムの主要機能を網羅し、細かな点は無視する。
  日次のビルドやスモークテストは、(ソフトウェア)業界において最も効率的・効果的な実践 方法の一つ。build verification test, intake test も参照のこと。
  多くの現場では、全体を流すテストとして使われているプランナー・開発者が、全体的な実装確認をする場合は、フリーデバッグとしてスモークテストをしていることになる

そもそも「フリーデバッグ」とは何か ～ 詳細 ～

Akatsuki

- **インテークテスト（ intake test ）:**
  スモークテストの特別な形態。コンポーネントやシステムが、詳細なテストやさらに詳細なテストを開始できるか判定するためのもので、テスト実行フェーズの開始時に行なうことが多い。
  組織によっては、検証開始条件としてフリーテストというなのインテークテストを実施する場合もある。

そもそも「フリーデバッグ」とは何か ～ 詳細 ～

Akatsuki

- **アドホックテスト（ad hoc testing）：**
  非公式に実施するテスト。公式なテストの準備をせず、実績のあるテスト設計技法を用いず、テスト結果も予測せず、毎回、テスト方法が変わる。
  上記のように、自由度の高いテストのように書いていますが、方向性・担当領域を決めてから実施している組織は、多いようです。

そもそも「フリーデバッグ」とは何か ～ 詳細 ～

Akatsuki

- **探索的テスト（ exploratory testing ）:**
  非公式なテスト設計技法の一つ。テストを実施する過程で、テスト担当者がテスト実施情報を活用しながらテスト設計をコントロールし、積極的に質の高い新しいテスト ケースを設計する。探索的テストには、多くの流派が有り、上記のようにテスト設計のために実施する場合もあれば、任意の工程で、抜き取り検査的に有識者・熟練者によるフリーテストとして実施する場合もある。
  品質特性(ISO/IEC 9126、JIS X 0129、ISO/IEC25010)をもちいて網羅的な観点で、探索的テストを実施するアプローチもある。

そもそも「フリーデバッグ」とは何か ～ 詳細 ～

Akatsuki

- **ユーザーテスト(user test) :**
  コンポーネント又はシステムの使用性を評価するために現実のユーザが 参加するテスト。
  ユーザーによるフリーテスト、使用性（usability）・理解性（understandability）・習得性（learnability）を目的としたテストとして実施されることが多い。

E. 0. F