レビュー再定義

@TAKAKING22

Photo by Jantine Doornbos on Unsplash

質問

レビューしていますか？

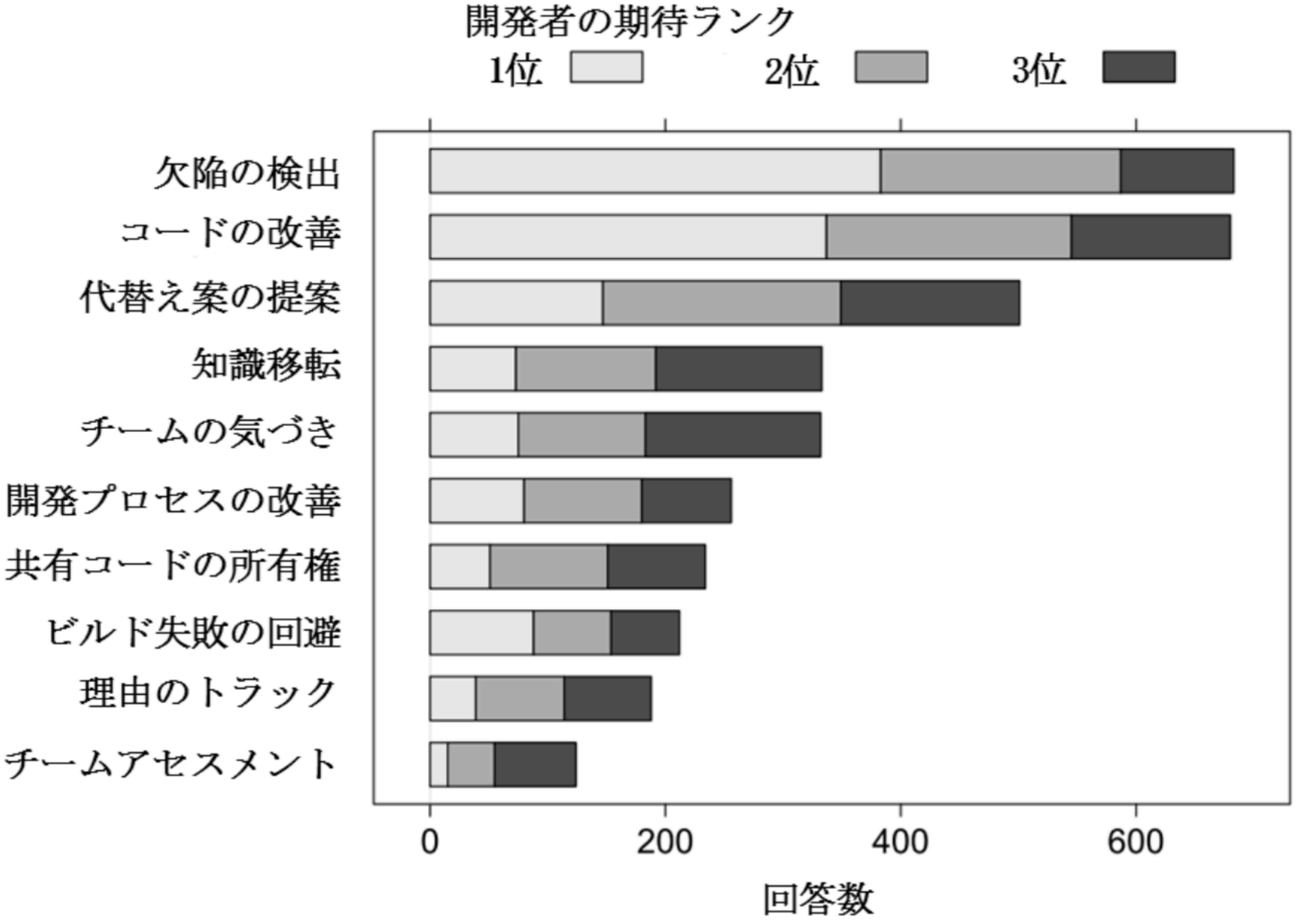
質問

あなたにとっての／あなたの考える
レビューの目的は何ですか？

レビューの目的

- 品質を担保するため
- バグを見つけるため
- コードの均一化をはかるため
- スキルの低いメンバーの教育のため
- スキルトランスファーをするため
- コードのクオリティを上げるため

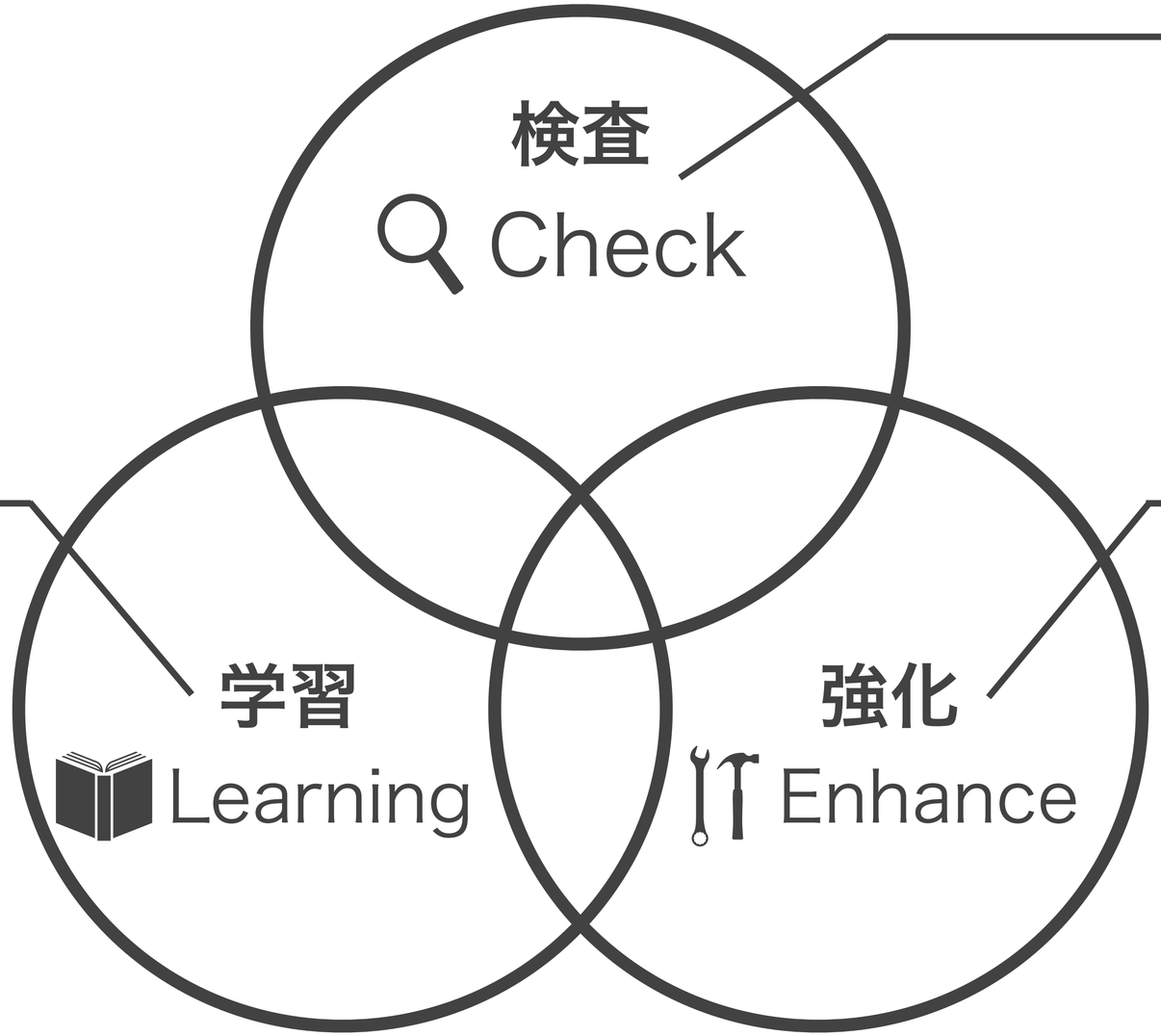
…



SQuBOK Review 2017 Vol.2 レビュー技術動向：図2 開発者のコードレビューに対する期待

@TAKAKING22

分類するとこの3つくらいになりそう



- 品質を担保する
- バグを見つける
- テストとして
- （最低限の）
  リファクタリング

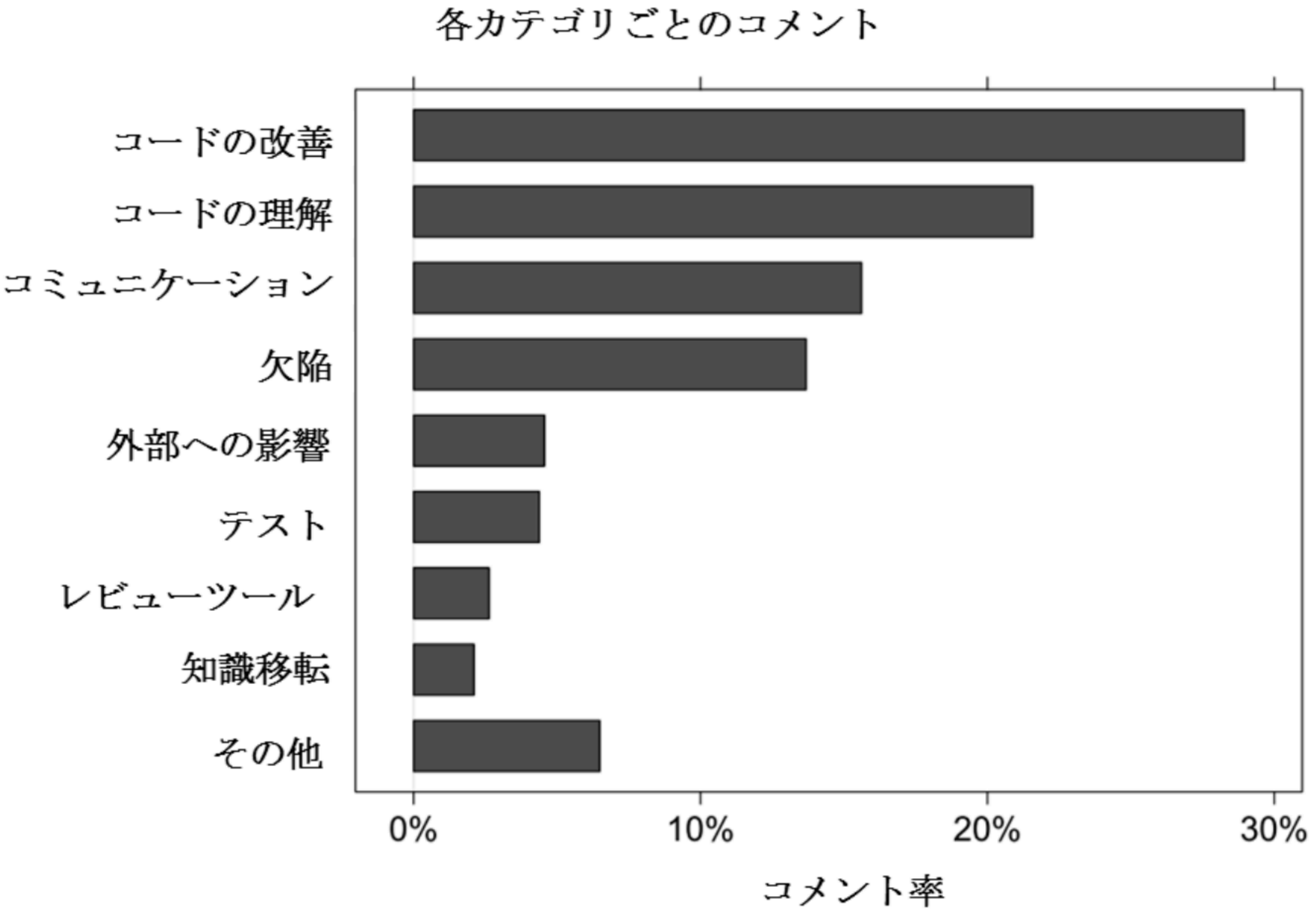
- ナレッジの共有
- スキルトランスファー
- 新人教育
- 文化の醸成

- リファクタリング
- もっとよく
- もっとキレイに
- 先行投資

@TAKAKING22

質問

レビューの目的を
達成できていますか？

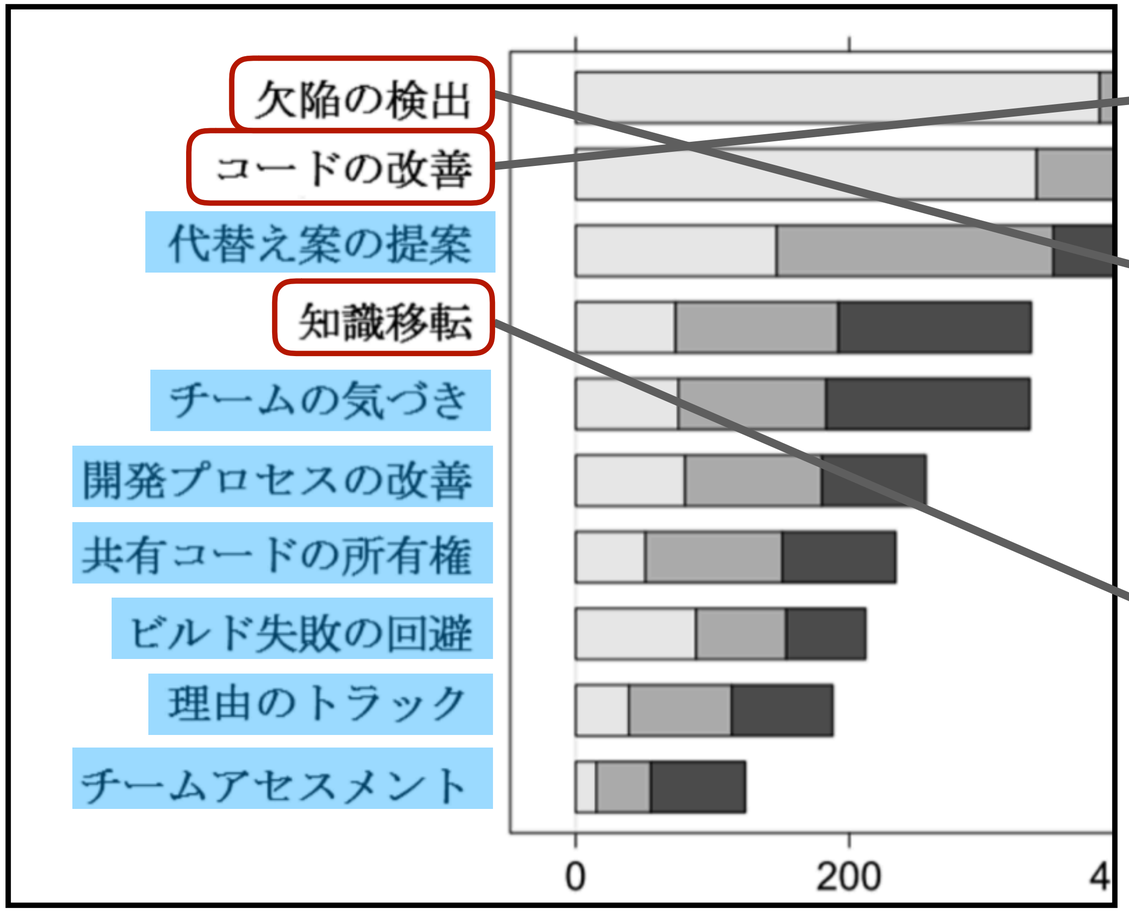


SQuBOK Review 2017 Vol.2 レビュー技術動向：図3 モダンコードレビューの目的に対するそれぞれのレビュー結果（指摘率）

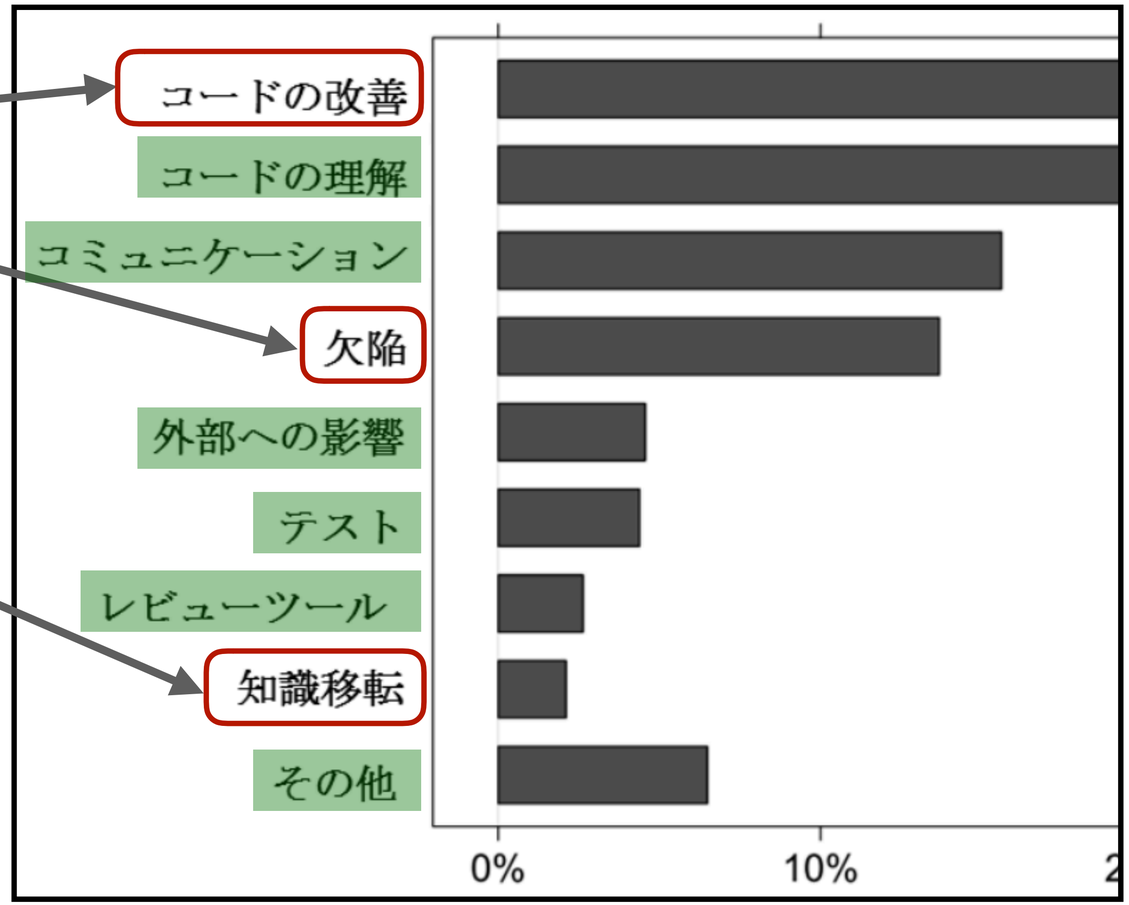
@TAKAKING22

レビューの目的が達成できていない！？

レビューの目的



実際のコメント内容

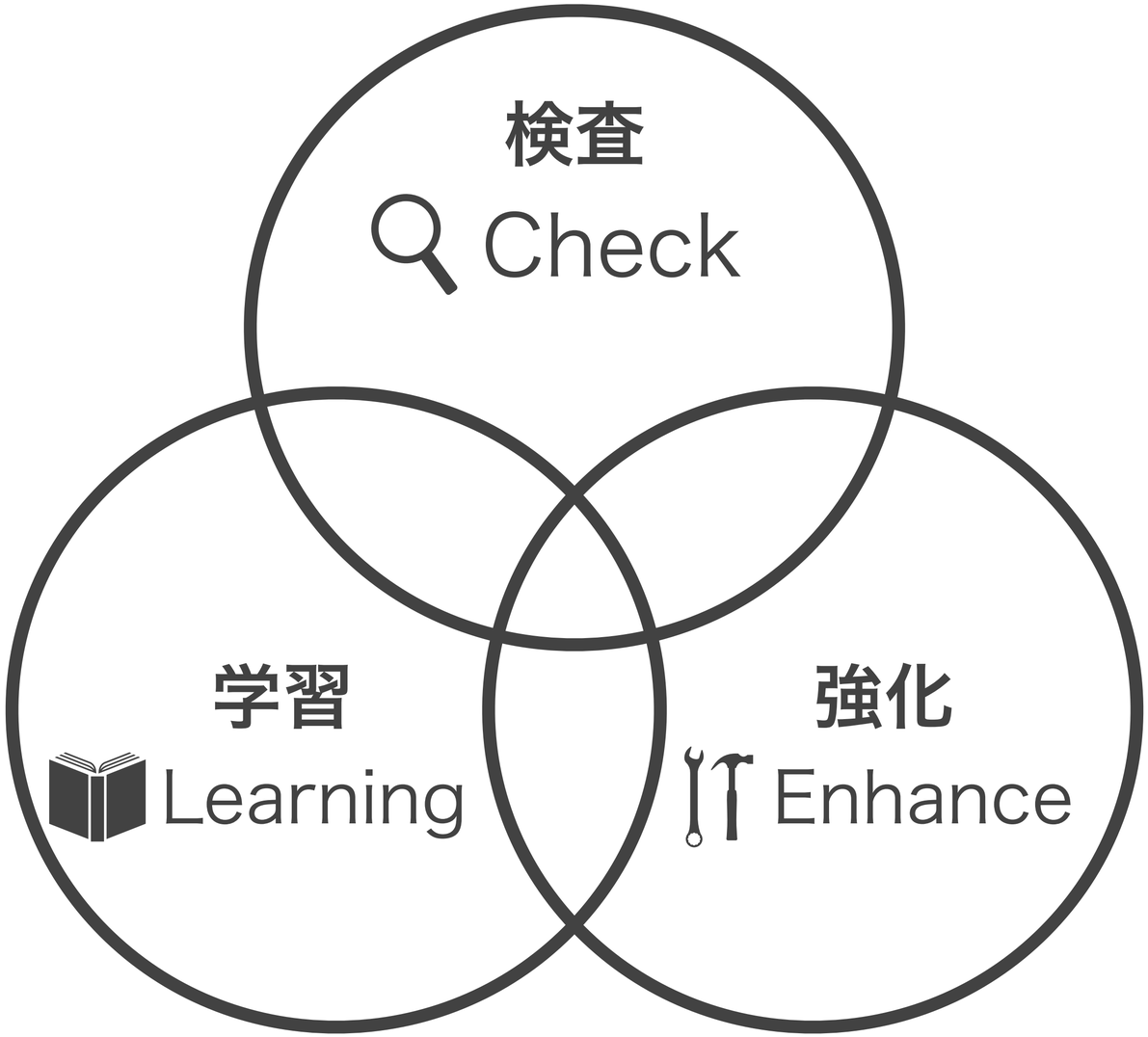


@TAKAKING22

質問

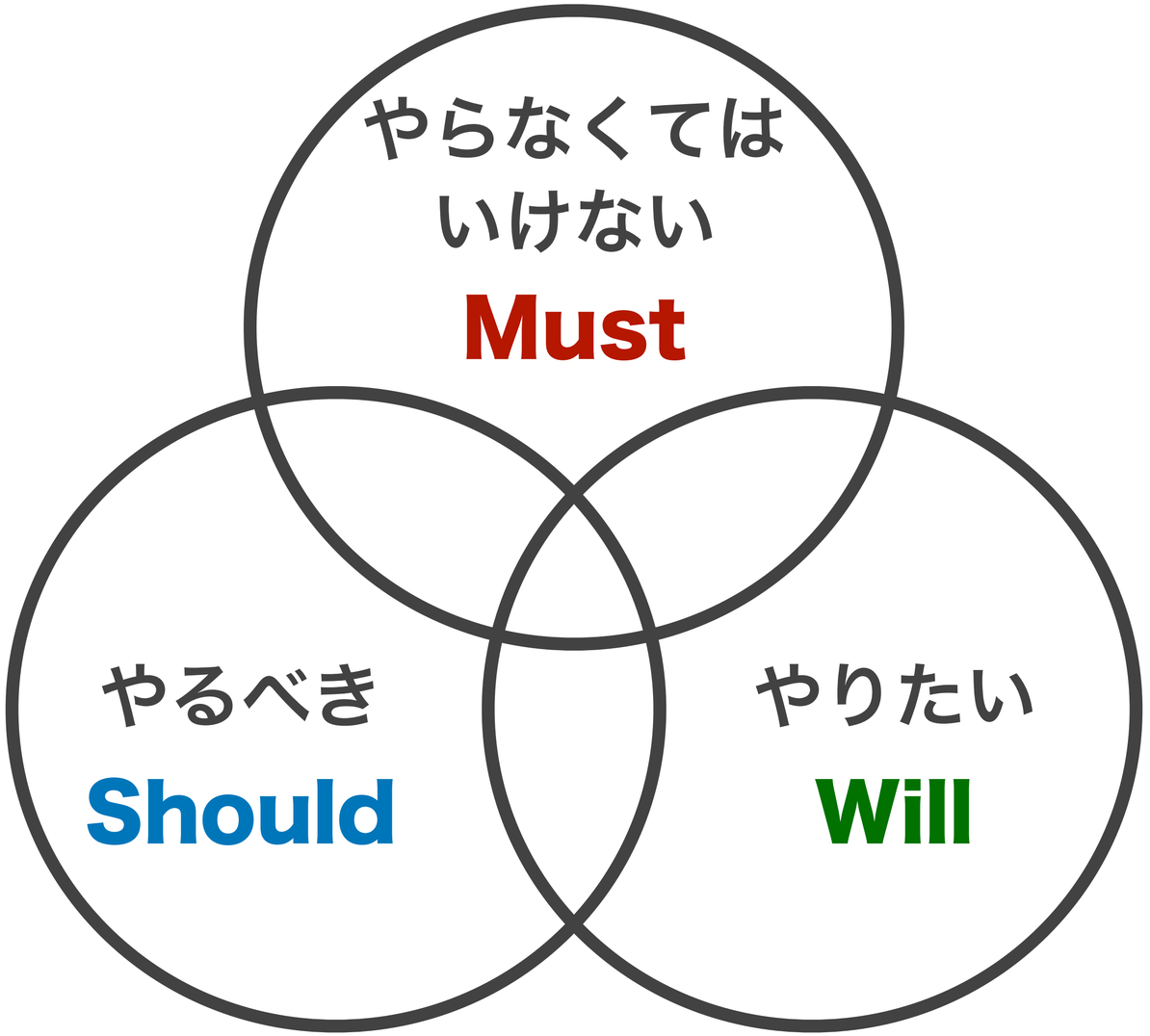
レビューは楽しいですか？

レビュー目的の3要素



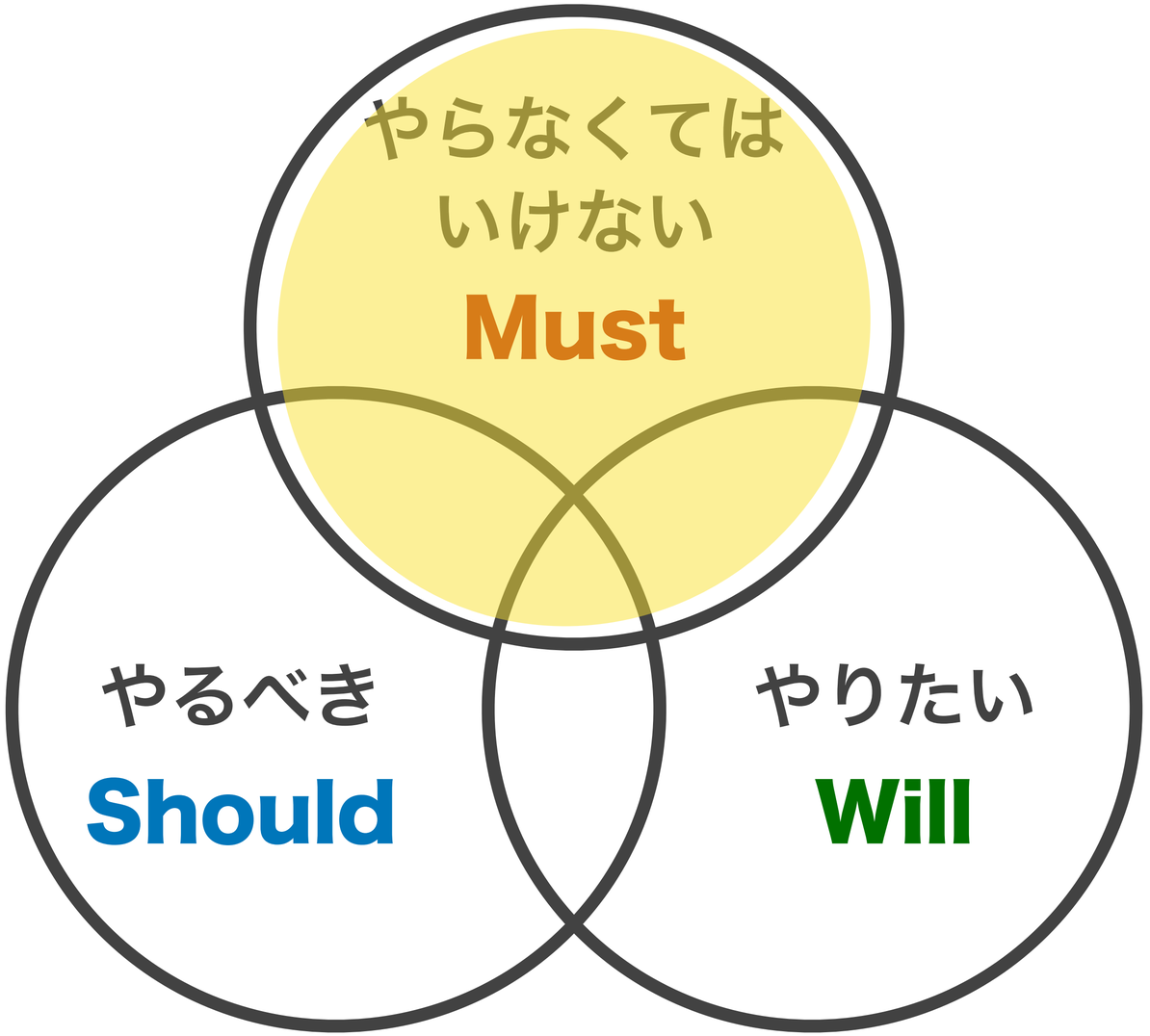
@TAKAKING22

もう少しわかりやすくすると…



@TAKAKING22

レビューの話をするとこのあたりに終始しがち



@TAKAKING22

実際にはやらなくてはいけないことすら難しい

意図がわからない

なんでこんな変数名
になってるの？

ここに置くべき
じゃないよね

こっちを直したら
こっちも直すだろ

仕様理解してる？

Photo by フリー写真素材ぱくたそ (https://www.pakutaso.com/)

エンジニアのキャリアの話

エンジニアとしてスキルアップしていくと、
エンジニアリーダーとかテックリードとかを任されたりする。

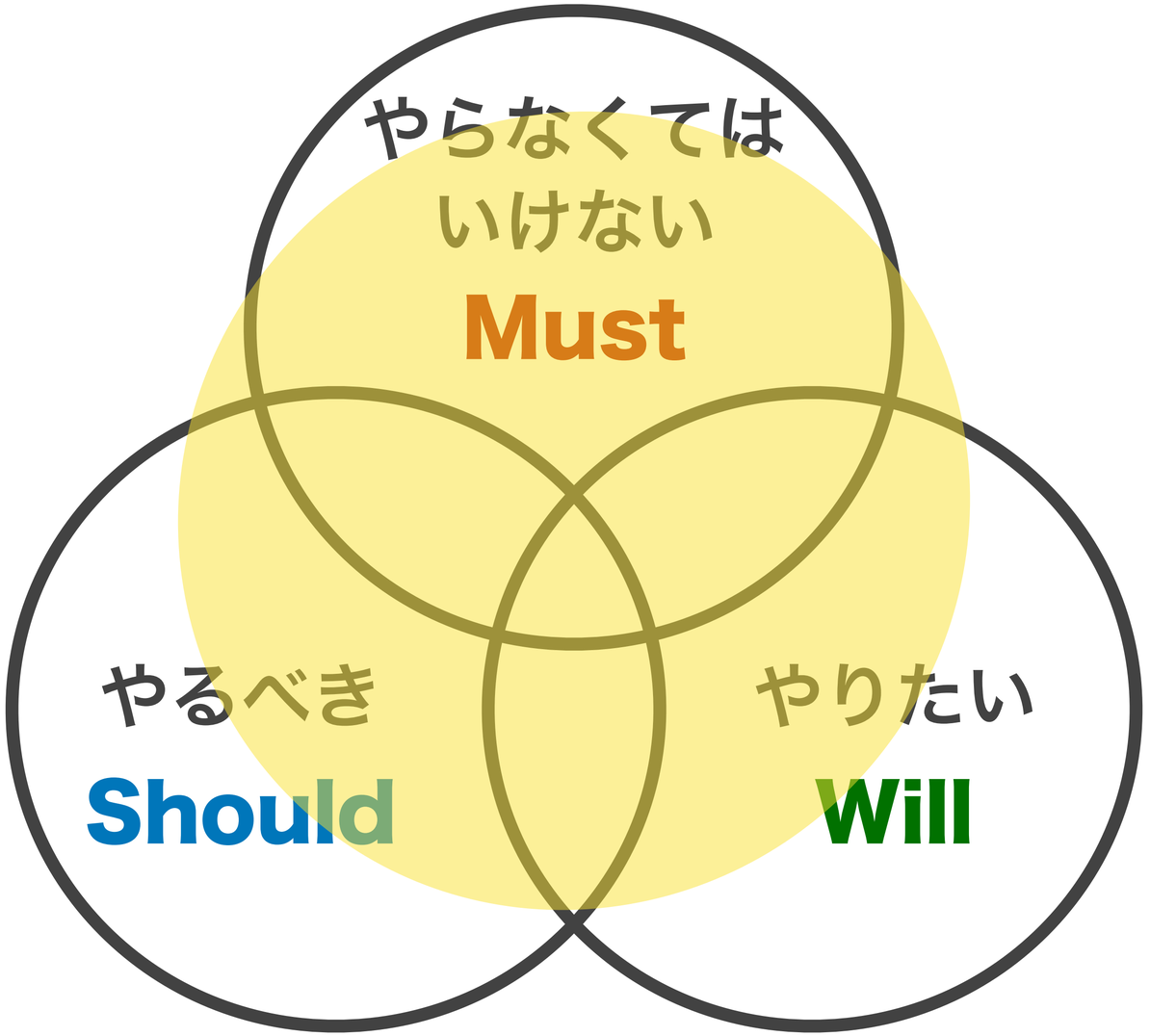
そうなると、メンバーの教育やシステムの安定稼働の責任がのってきて、
例えばレビューをする時間がどんどん増えていく。

エンジニアリングが好きで頑張ってきたんだけど、
スキルアップしていくとどんどんコードを書く時間が減っていくのってなんかツライ。

なんかツライ

@TAKAKING22

もっとわくわくするレビュー



@TAKAKING22


モブプログラミング

モブプログラミングの経験を通して、
“もっとわくわくするレビュー”
についてのアイデアが出てきたので
そんな話をします。

@TAKAKING22

及部 敬雄

@TAKAKING22

育児休暇中

楽天株式会社 エンジニアリングマネージャー

一般社団法人アジャイルチームを支える会 理事

アジャイルモンスター　モブプログラミングおじさん

推しメン：山本 彩

- FA宣言中
- https://takaking22.com/contact/

ブログ

https://takaking22.com/

スライド

https://speakerdeck.com/takaking22

アジャイルチームを支える会

一般社団法人 アジャイルチームを支える会

私たちは、日本国内におけるソフトウェア開発、特にアジャイル開発に関する活動の振興や人材の育成を目的とした団体です。
「もっと国内によいアジャイルチームを増やしたい」という想いを持っています。
2014年6月にたくさんの方のご支援を得て、2014年11月に一般社団法人として設立しました。

http://www.agileteam.jp/

@TAKAKING22

今日のお話

- モブプログラミングを始めてレビューが不要になった
- でもレビューはやっぱり必要だった
- 我々にとってのレビュー
- レビューを再定義する

やらないことリスト

- モブプログラミングを薦めること
- 詳細なレビュー方法についての説明
- ○○レビューに限定した話

こうなるといいなー

- レビューについて再定義するきっかけ
- もっとわくわくするレビューが少しだけ増える

@TAKAKING22


モブプログラミング

モブプログラミングとは

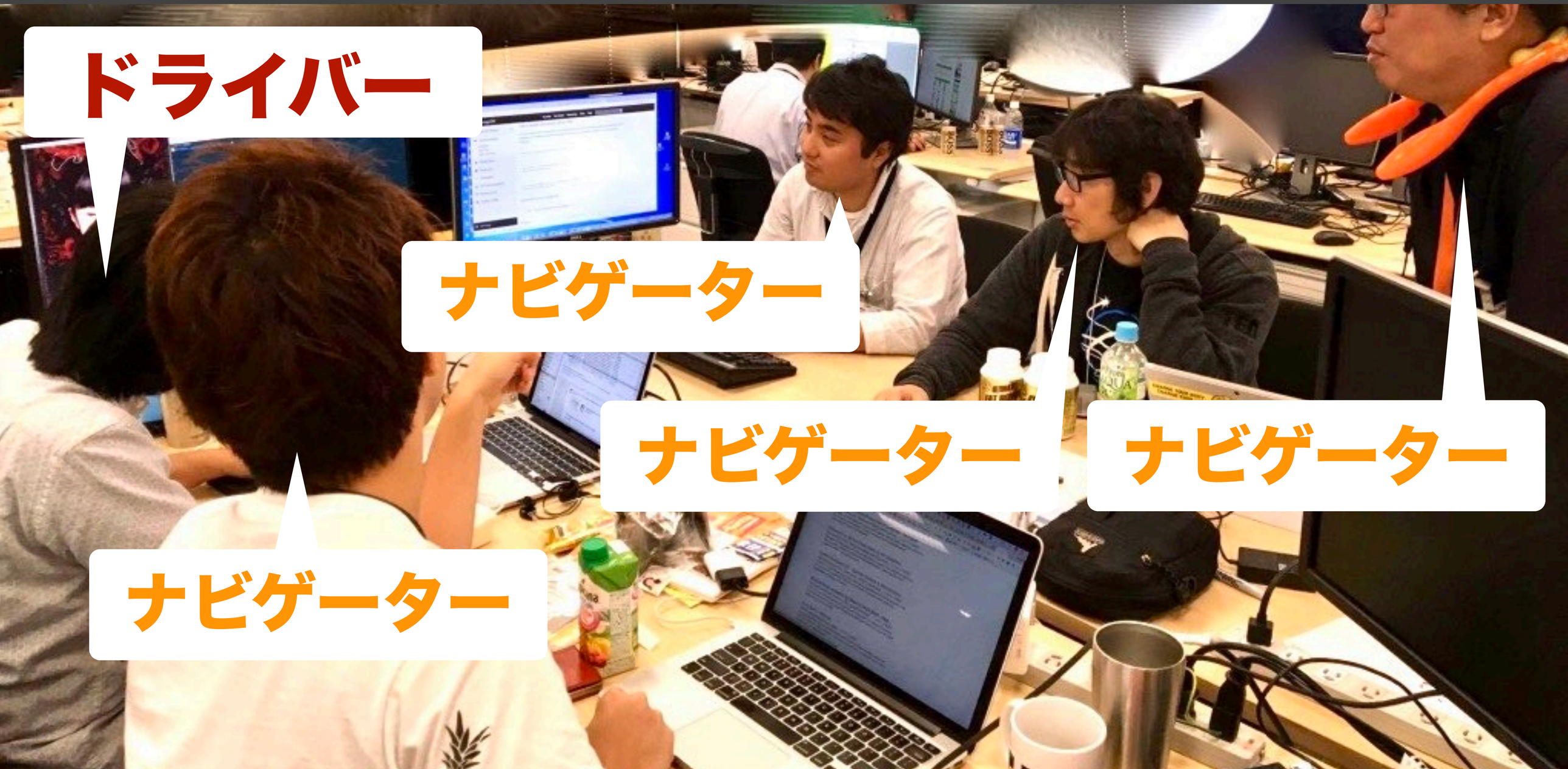
チーム全員で

- ✳ 同じ**仕事**を ..
- ✳ 同じ**時間**に ..
- ✳ 同じ**場所**で ..
- ✳ 同じ**コンピューター**で ..

すること

1つのディスプレイと1つのコンピューター

1人のドライバーと複数人のナビゲーター



我々のモブプログラミング

- 楽天株式会社 新規事業開発部
- 4人チーム→3人チーム
- 働き方＝モブプログラミング
- 開発も運用も企画も営業も
- 2017年5月〜現在

@TAKAKING22

働き方としての~~モブプログラミング~~
モブワーク

09:00-11:00　個人時間
11:00-11:15　モブプランニング
11:15-11:30　モブモンキーテスト
11:15-13:00　モブワーク
13:00-14:00　モブランチ
14:00-17:15　モブワーク
17:15-17:30　モブふりかえり

@TAKAKING22

モブワークを始めてやらなくなったもの

- 朝礼（3つの質問）
- チーム内ミーティング
- 合意形成のためのドキュメント
- レビュー全般（ソースコード、ドキュメント）
- Git Flow
- カンバンの個人レーン

レビューもやらなくなった

@TAKAKING22

分担作業とモブワークを比較してみる

分担作業

VS

モブワーク

@TAKAKING22

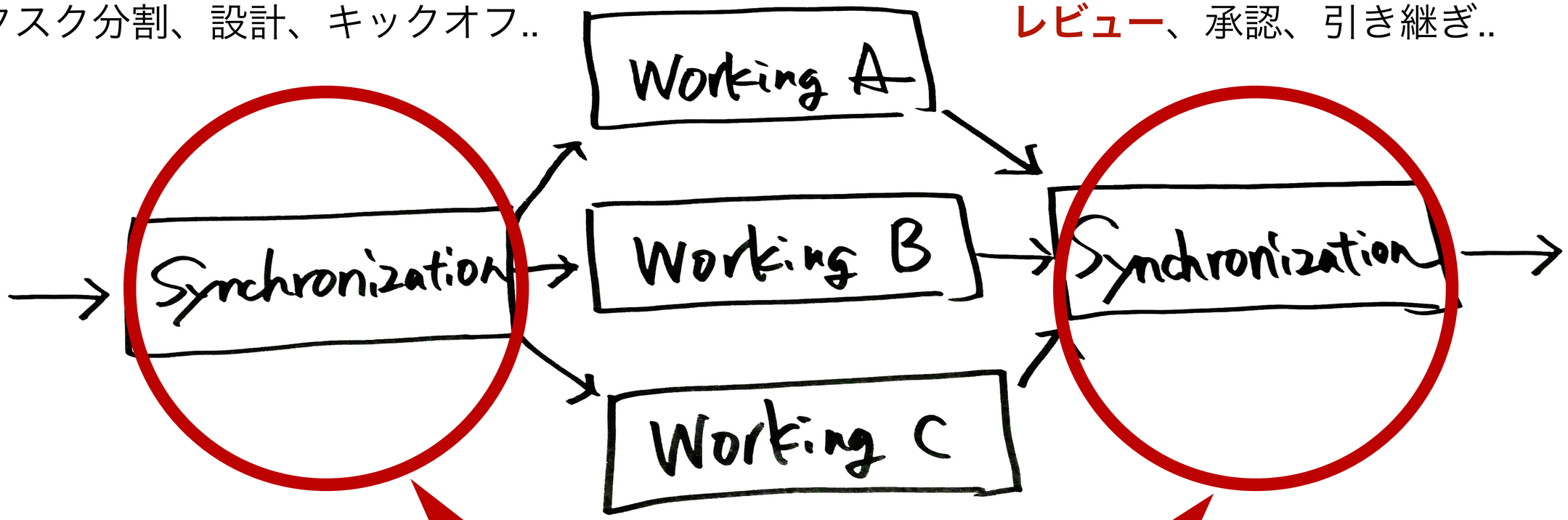
分担作業

ex.

タスク分割、設計、キックオフ..

ex.

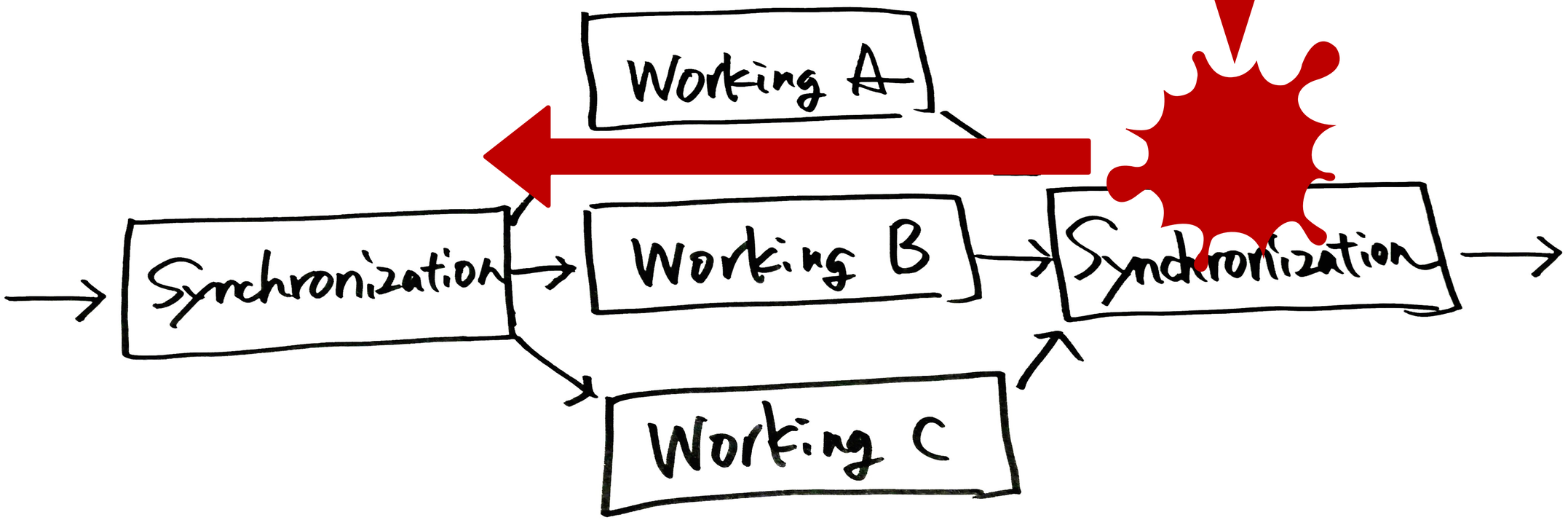
レビュー、承認、引き継ぎ..



分担作業の前後に
同期する作業が必要になる

分担作業

ミスコミュニケーションによって
手戻りする可能性もある



モブワーク

ex.

設計、レビュー、ノウハウ共有

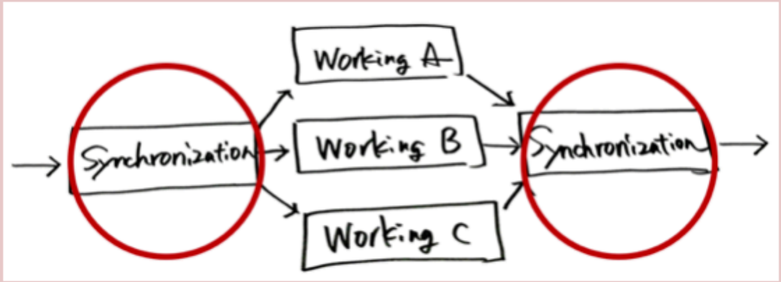
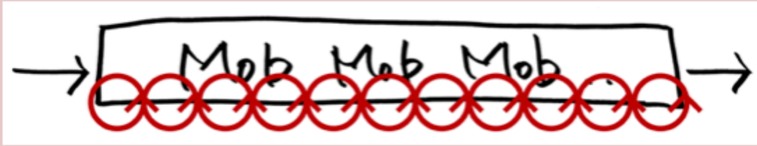
同期していないではなくて
常に同期している

@TAKAKING22

モブワークを始めてやらなくなったもの

- 朝礼（3つの質問）
- チーム内ミーティング
- 合意形成のためのドキュメント
- レビュー全般（ソースコード、ドキュメント）
- Git Flow
- カンバンの個人レーン

これらはすべて作業前後に同期する作業

	分担作業	モブワーク
イメージ		
同期コスト	高い	低い
リソース効率	良い	悪い
フロー効率	悪い	良い
生産量	多い	少ない
生産性	?	?
学びの機会	個人に依存しがち	全員で共有できる
	誰がやっても同じ結果になりやすい作業 ex. 定常運用作業、検索	問題解決など学びと情報共有が必要な作業 ex. 環境構築、POC

Rakuten

Teamwork Revolution チームとものづくりに真正面から向き合うモブプログラミング

@TAKAKING22

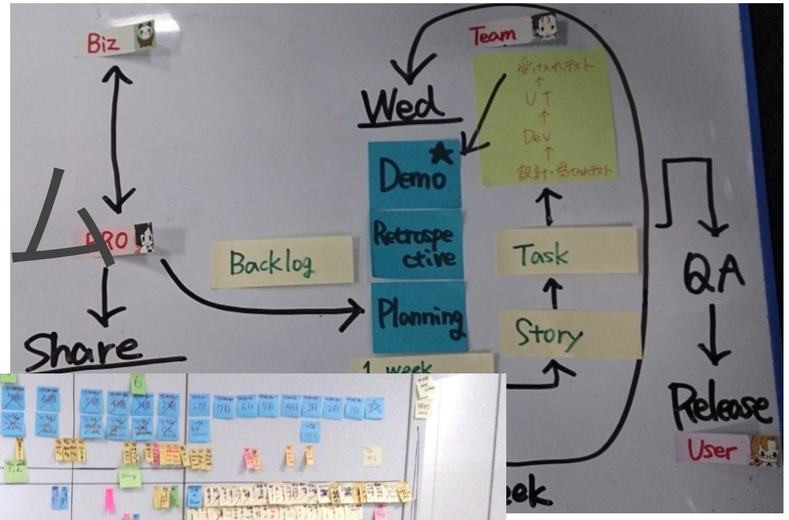
分担することが前提になっていなかったか

「分担する＝生産性が高い」という思い込み

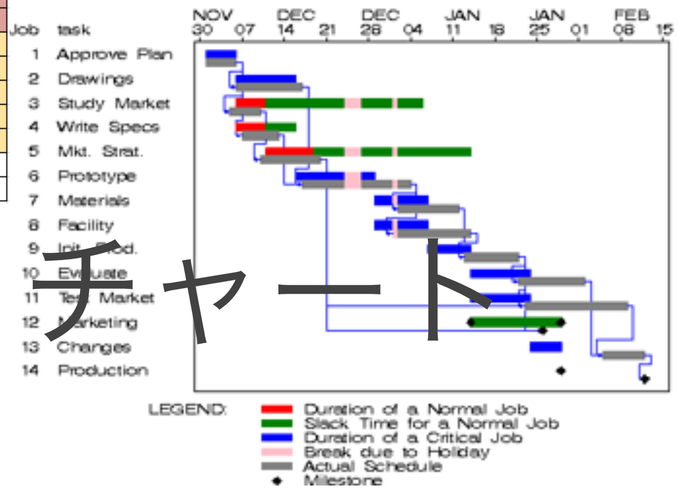
WBS

No	要望	優先順位
1	AAA機能	高
2	BBB機能	高
3	CCC機能	高
4	DDD機能	高
5	EEE機能	高
6	FFF機能	高
7	GGG機能	高
8	HHH機能	高
9	III機能	高
10	JJJ機能	高
11	LLL機能	高
12	MMM機能	高
13	NNN機能	高
14	OOO機能	中
15	PPP機能	中
16	QQQ機能	中
17	RRR機能	中
18	SSS機能	中
19	TTT機能	低
20	UUU機能	低

スクラム



ガントチャート



カンバン

@TAKAKING22

二者択一ではない

分担作業

VS

モブワーク

@TAKAKING22

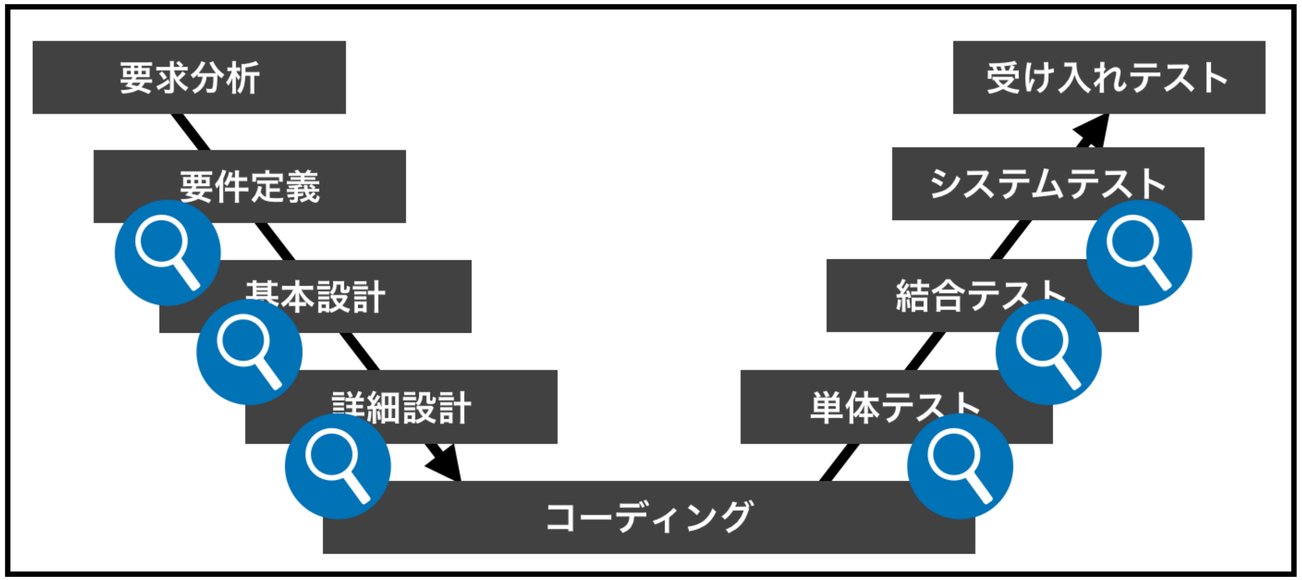
仕事や状況に合わせて使い分けている

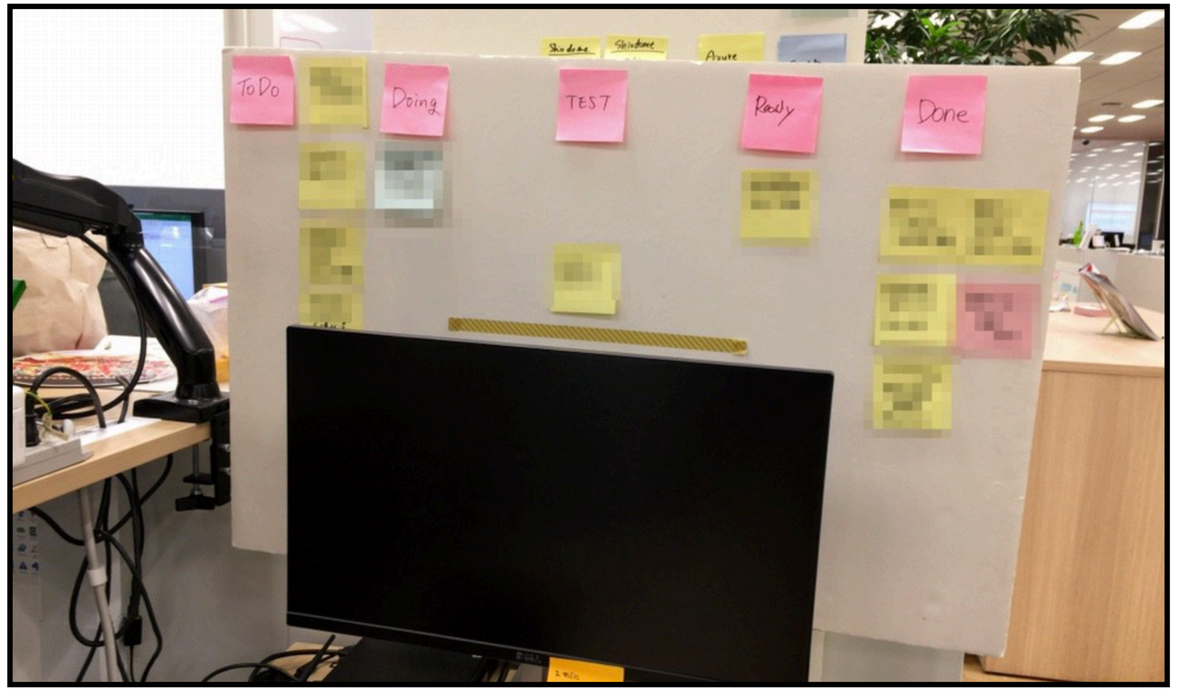
分担作業

モブワーク

@TAKAKING22

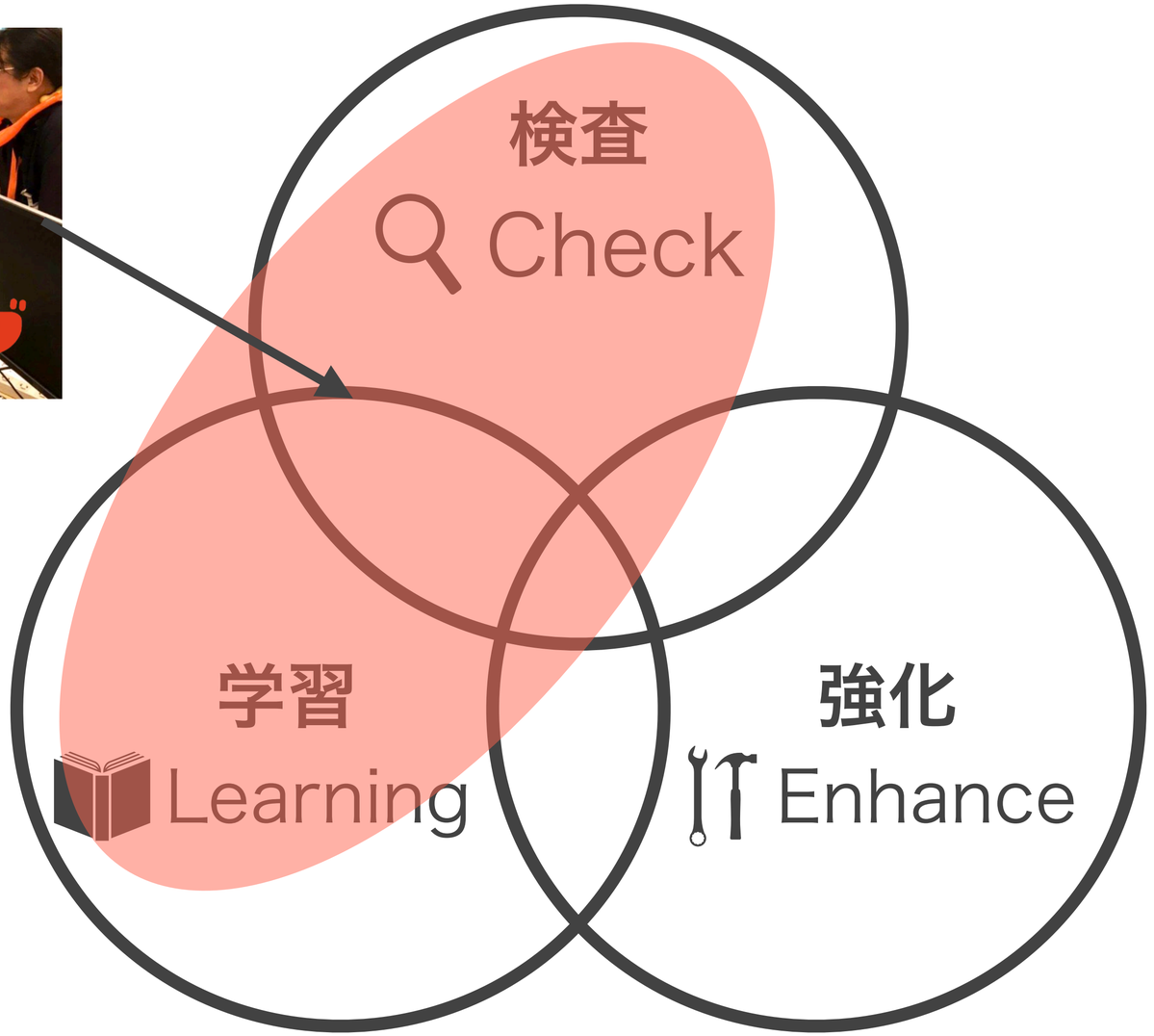
工程としてのレビューは不要になった





@TAKAKING22

モブプログラミングの強み



@TAKAKING22

モブプログラミングの検査

- リアルタイムでフィードバック
- タイポすると親の仇のように指摘される
- チーム全員の合意のもとでコードが組まれていく
- 忖度せずにチームの最善を目指すことができる

@TAKAKING22

レビューにおける忖度

ちょっとイケてないんだけど、
今パツパツだろうし、
動くからとりあえずいいか…

まだこれは理解できないから
言わなくてもいいや

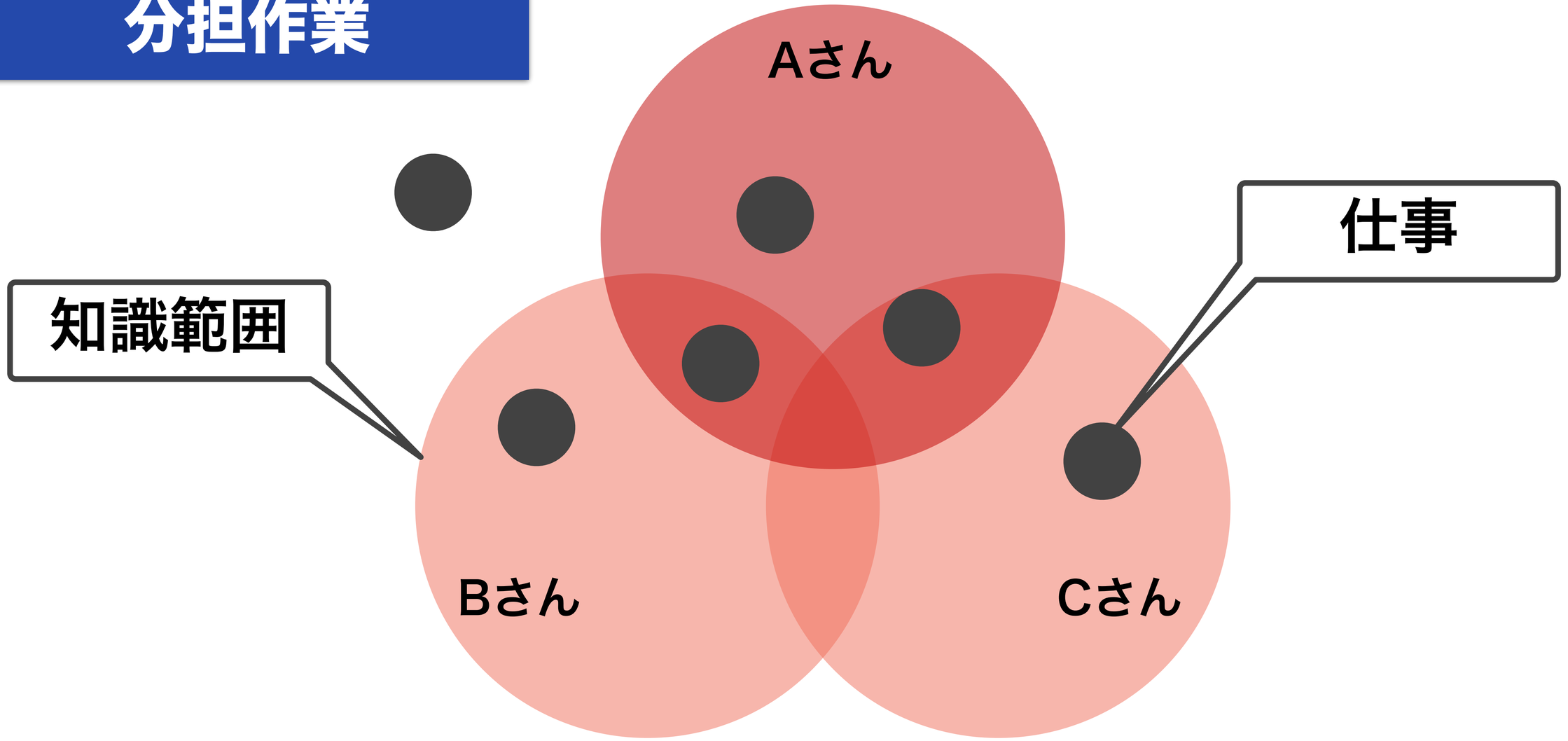
今回は指摘多いから
これくらいにしておこう

ストレス源

Photo by フリー写真素材ぱくたそ (https://www.pakutaso.com/)

チームの最善を目指す

分担作業



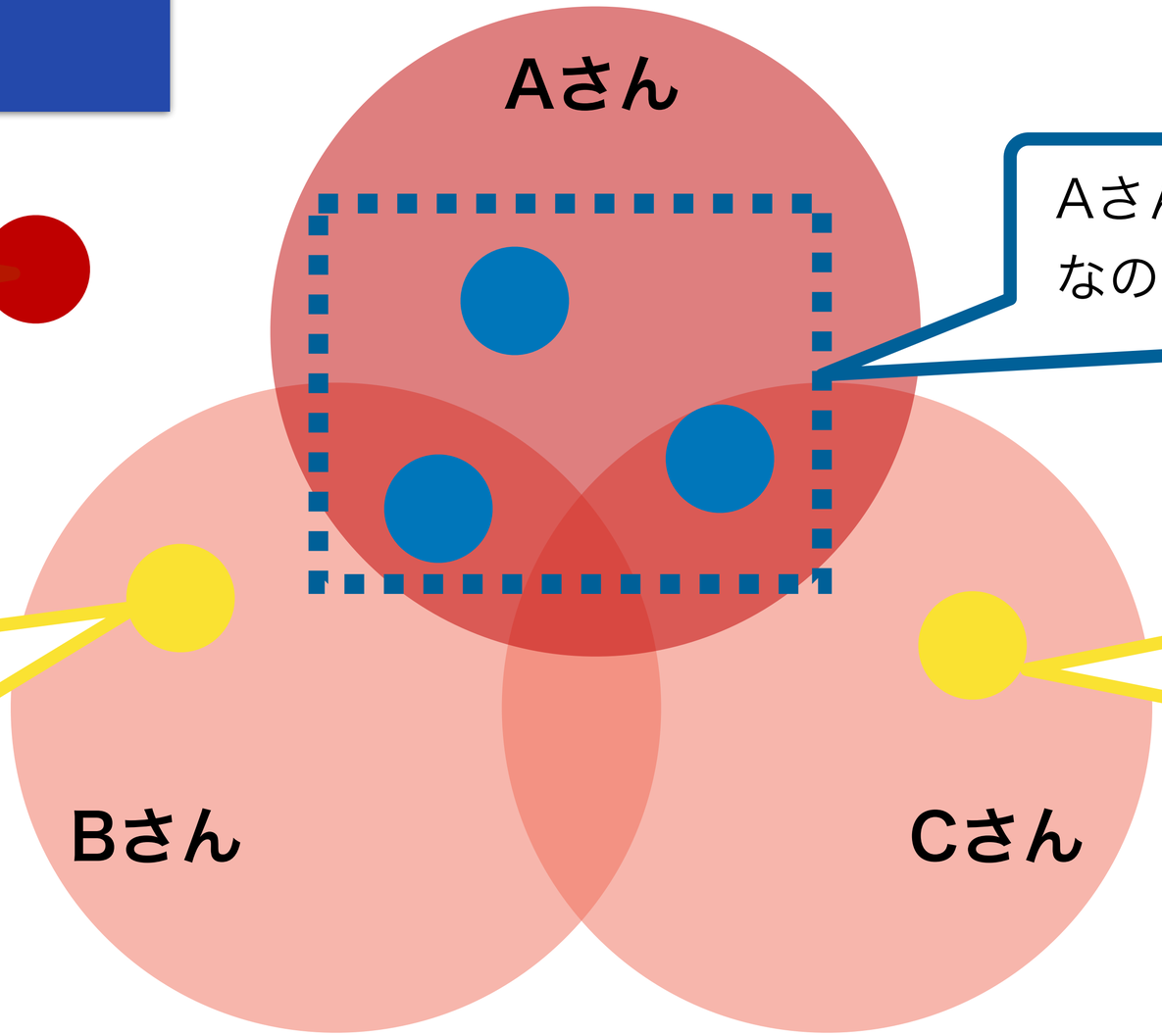
@TAKAKING22

チームの最善を目指す

分担作業

このチームでは
対応が難しい

アサインされると
不安で仕方がない



Aさんの知識範囲内
なので対応可能

Aさんの知識範囲外
Bさんにレビューして
もらうことができれば
フォロー可能

Aさんの知識範囲外
Cさんにレビューして
もらうことができれば
フォロー可能

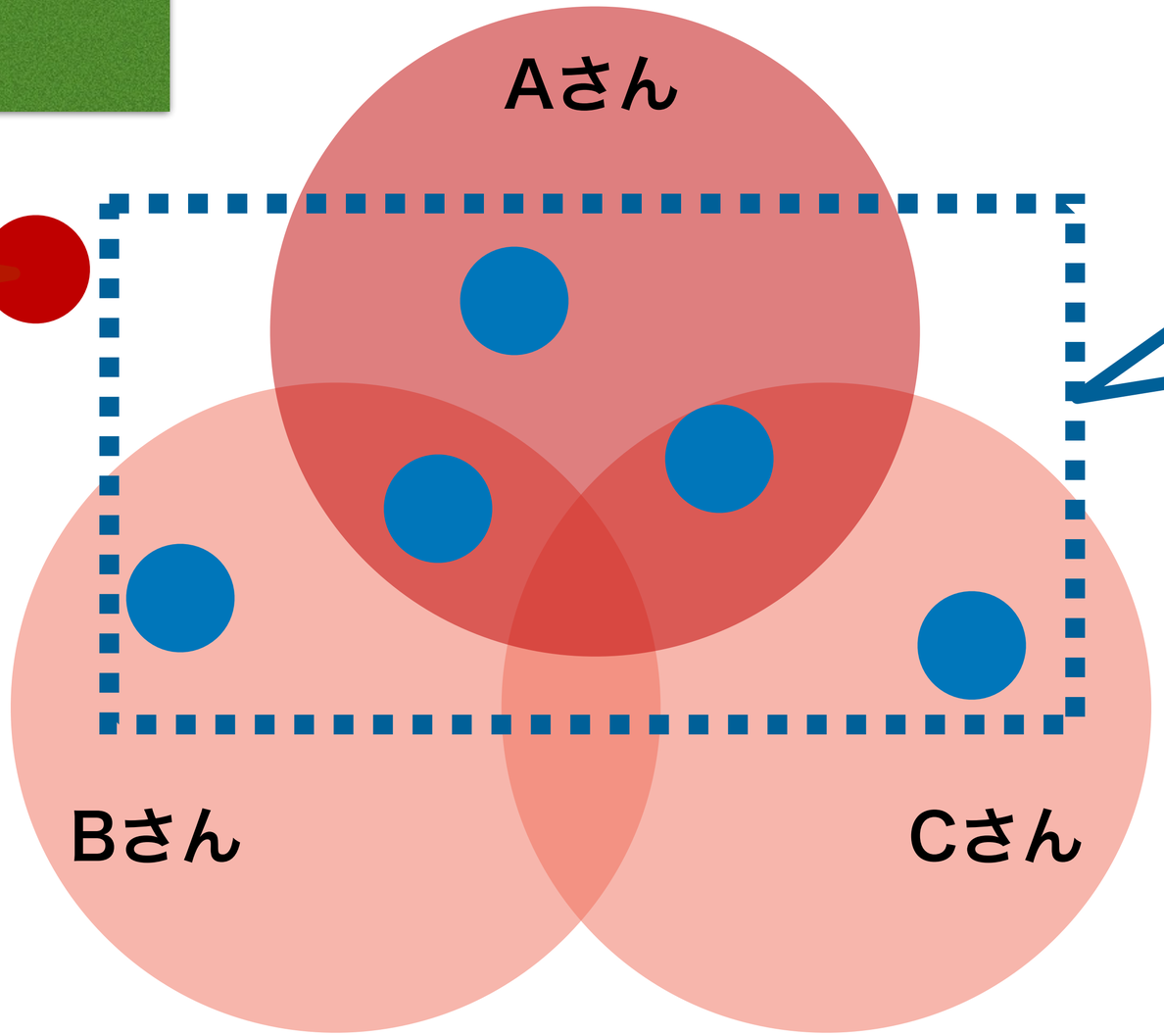
@TAKAKING22

チームの最善を目指す

モブワーク

このチームでは
対応が難しい

チーム全員でやっても
ダメなので割り切って
対応する判断ができる



チームの知識範囲内
なので対応可能

Aさんにとっては
はじめての仕事でも、
まわりのナビゲーターに
フォローしてもらいながら
進めることができる

@TAKAKING22

モブプログラミングはチーム全体の活動である

- チーム全員で立ち向かってダメなら仕方がない
- 判断をチームですることができる
- 絶対なる安心感とそこから学ぶ姿勢

@TAKAKING22

モブプログラミングの学習

- チームの学びの量が最大化する
- 結果だけでなく過程を学ぶことができる
- 新しい技術の習得もチームで立ち向うことができる
- 新メンバーの受け入れや教育にも向いている

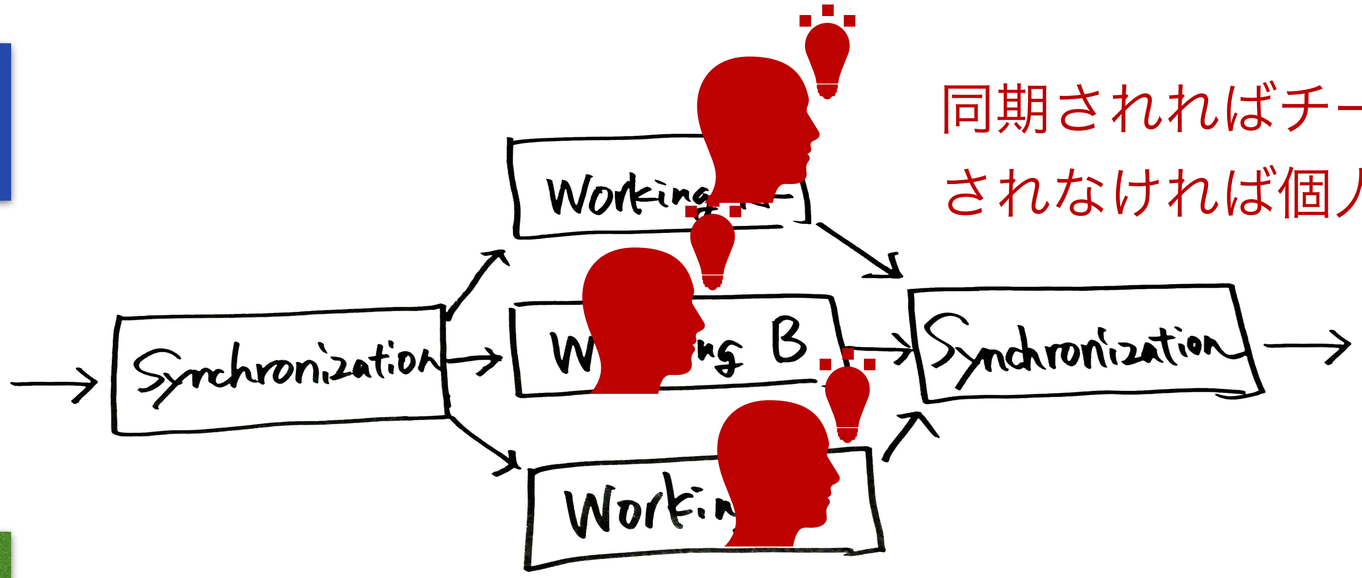
モブプログラミングで教育

- 経験値が低い人をドライバーにするとよい
- わからなければ手を止める権利を
- 聞きながら手を動かして覚えることができる
- ノウハウのJust in time

@TAKAKING22

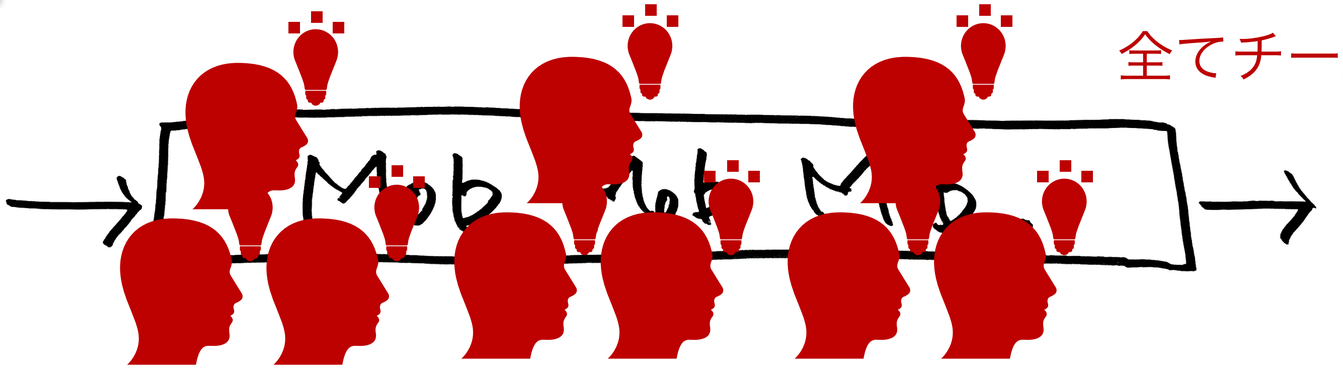
学びの量の最大化

分担作業



同期されればチーム全員の学びになるが、
されなければ個人の学びだけになる

モブワーク



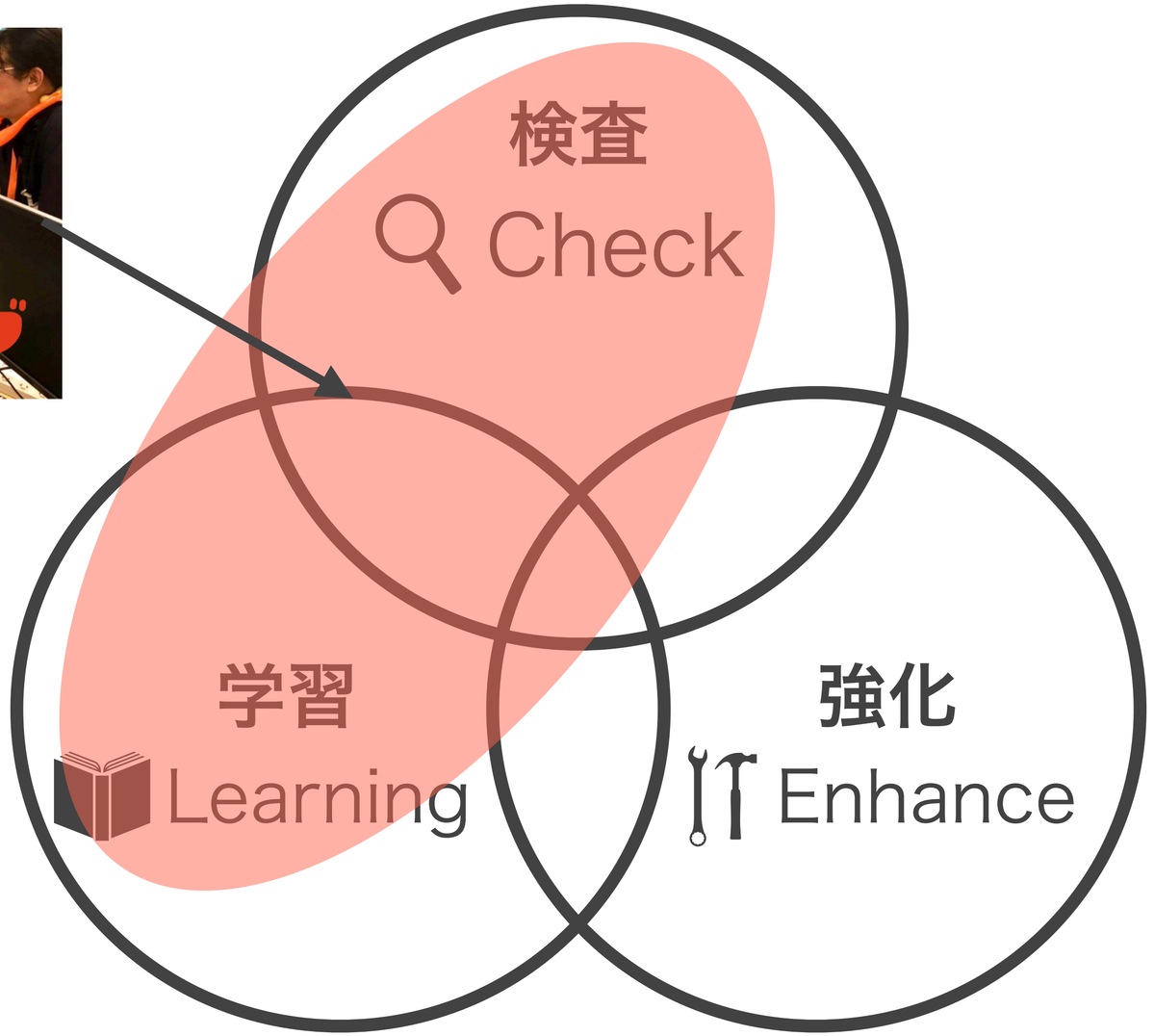
全てチーム全員の学びになる

@TAKAKING22

形式知も暗黙知もまとめて伝える

- 暗黙知は意識できないから暗黙知になっている
- 暗黙知は暗黙知のまま共通体験で伝える
  - なぜつくったのか
  - どうやってつくったのか
  - どんな問題解決のプロセスを経たのか
  - どんな学びがあったのか
  - どこに苦労したのか

モブプログラミングの強み

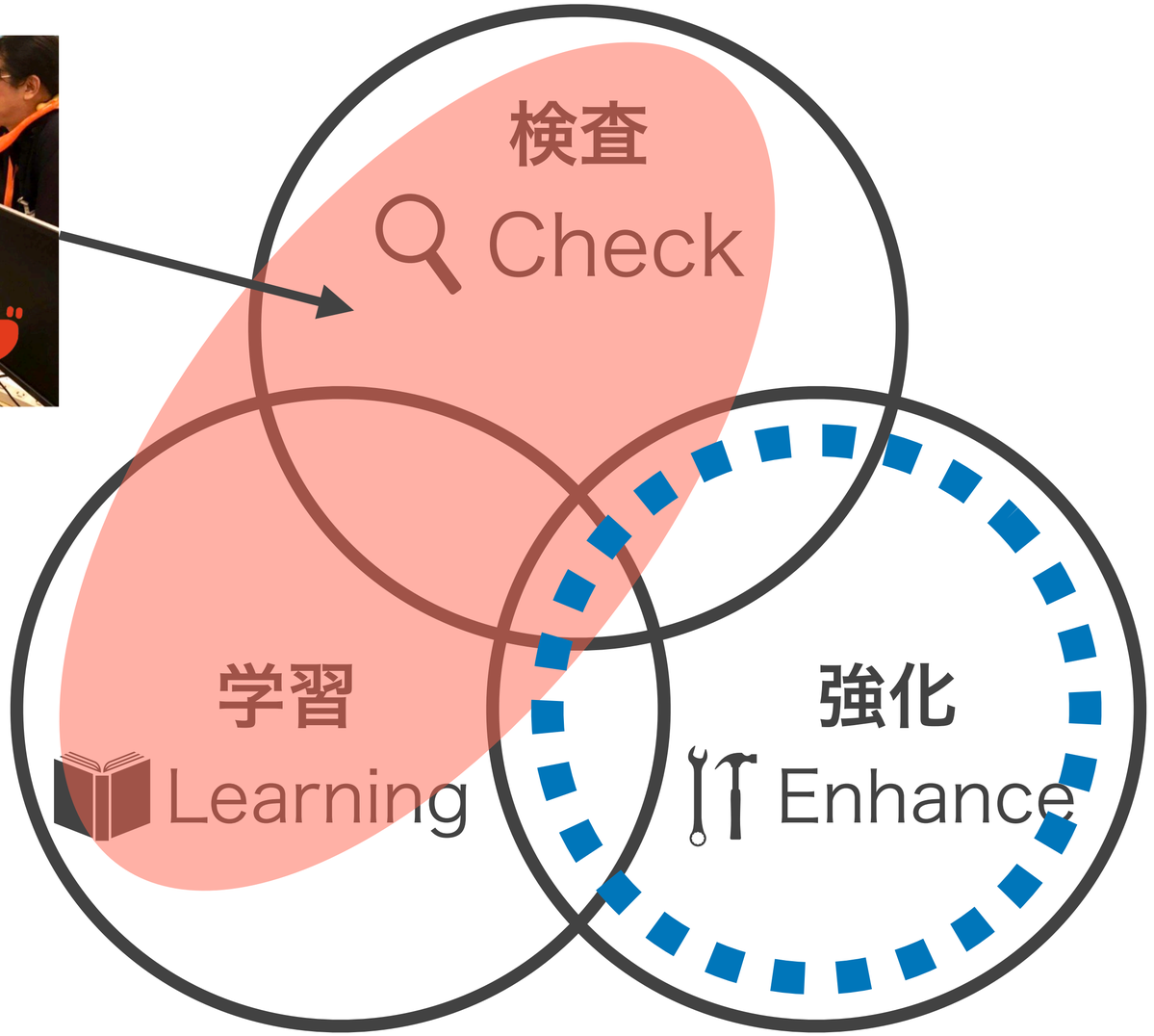


@TAKAKING22

ところがレビューの必要性を感じてきた!!

@TAKAKING22

足りない部分を補うレビュー



@TAKAKING22

我々に必要なレビュー

- Pull-Request＝自分たちの仕事の成果を
  自分たちで見直す
- 最＆高の仕事ができたのか
- もっとよくする余地があるのか
- 次に自分たちが学ぶべきことは何か

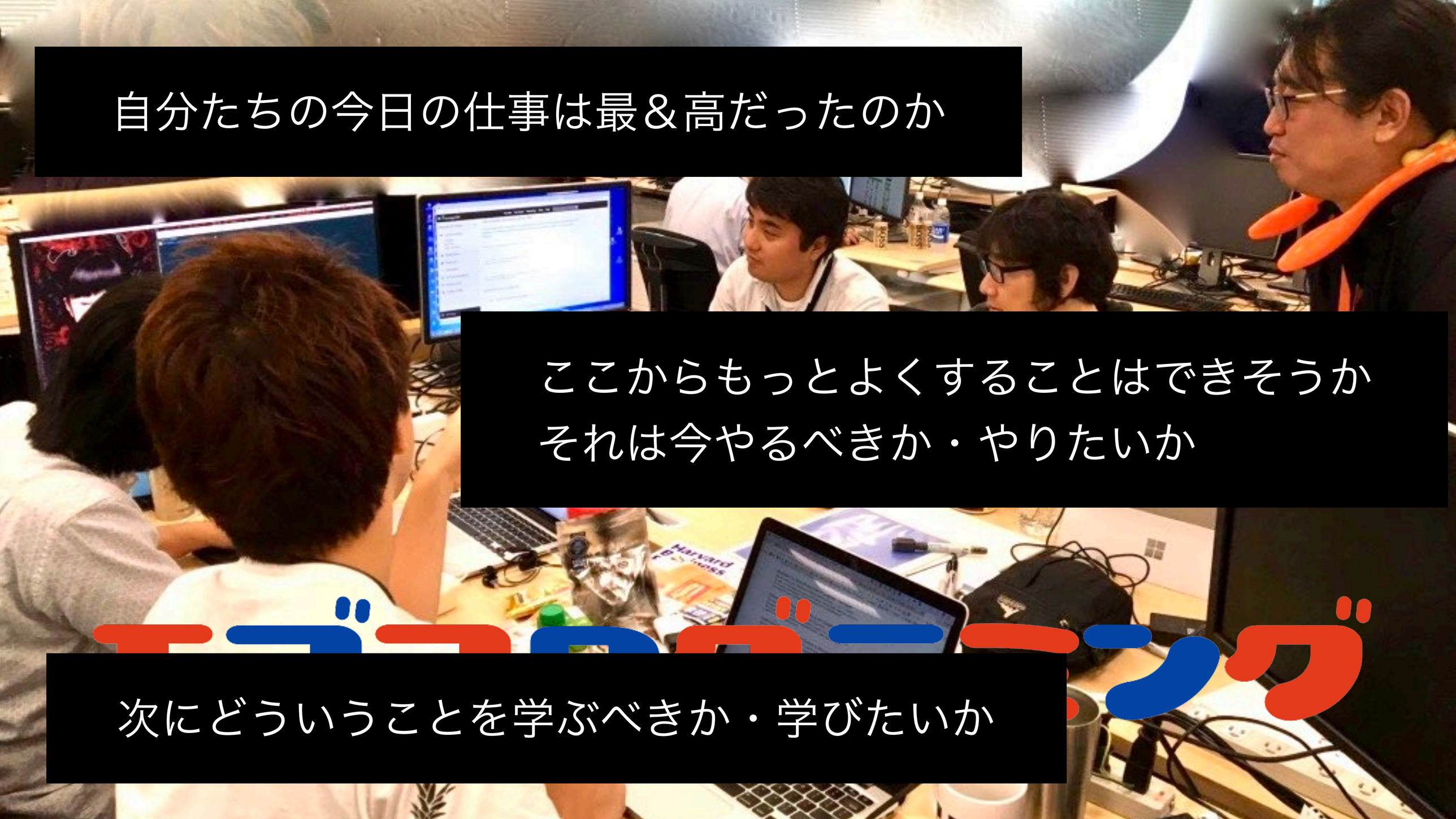
ふりかえりに近いレビュー

@TAKAKING22

働き方としての~~モブプログラミング~~
モブワーク

09:00-11:00　個人時間
11:00-11:15　モブプランニング
11:15-11:30　モブモンキーテスト
11:15-13:00　モブワーク
13:00-14:00　モブランチ
14:00-17:15　モブワーク
17:15-17:30　モブふりかえり

@TAKAKING22


自分たちの今日の仕事は最＆高だったのか
ここからもっとよくすることはできそうか
それは今やるべきか・やりたいか
次にどういうことを学ぶべきか・学びたいか

再び　　見る

Re-view

@TAKAKING22

あなたの言うレビューは本当にRe-viewなのか

- 実際にはFirst-viewであることが多い
- 初めて見る場合は理解するコストが発生する
- 非同期で他人の思考・仕事を理解をすることは、自分で思っているよりも難しい

	強み	弱み
リアルタイム モブプログラミング	✳ 常にチームの最善を尽くすことができる ✳ 結果だけでなくプロセスを評価することができる	✳ その場の熱によって誤った判断をしてしまう可能性 ✳ 局所最適に陥る可能性
レビュー Re-view	✳ 冷静に見ることができる ✳ 全体最適を考えてフィードバックしやすい	✳ 忖度が生まれやすい ex. イマイチだけど時間がないから… ✳ 結果のみで読み取れない部分を知るためには、コミュニケーションが必要

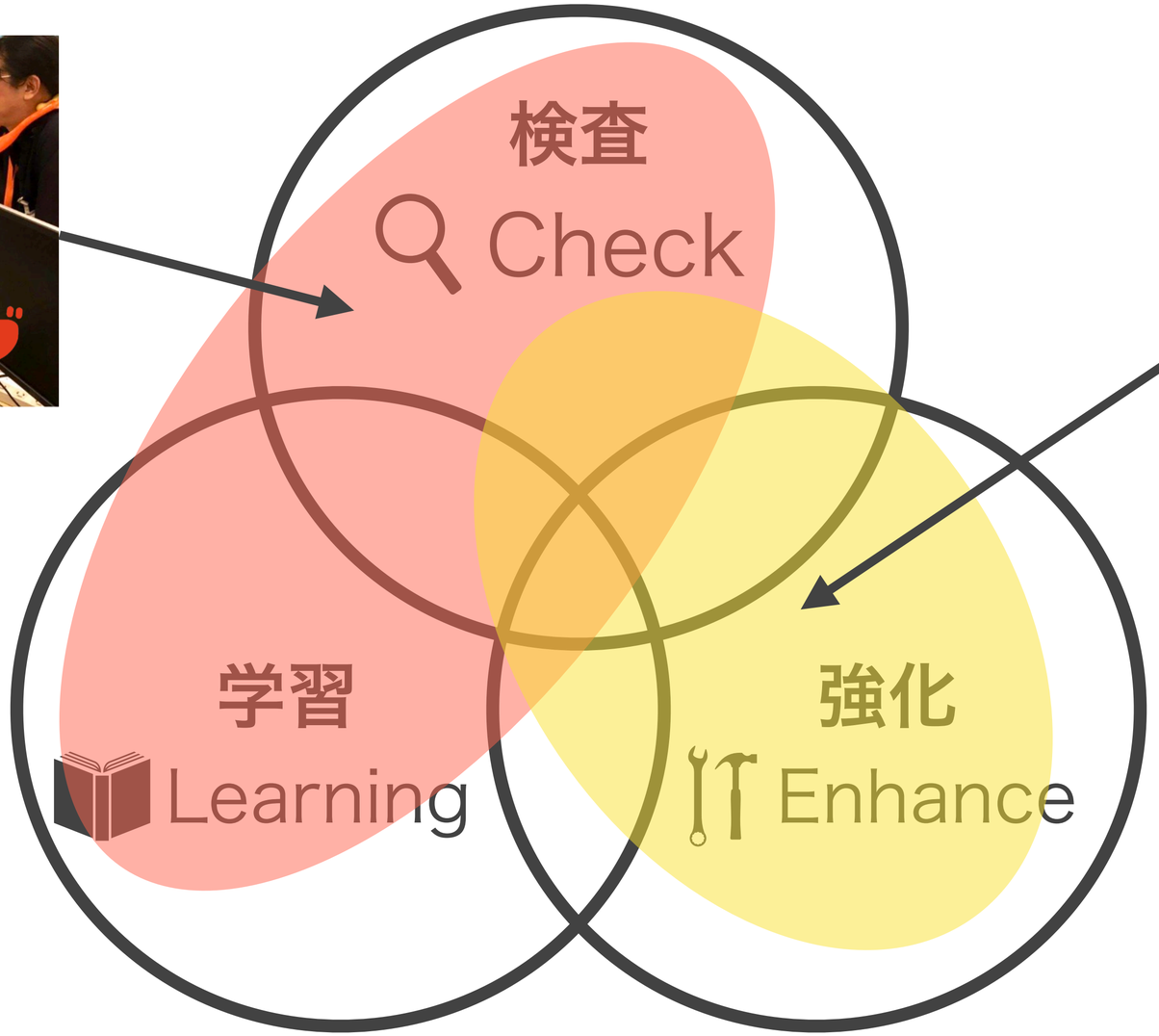
@TAKAKING22

	強み	弱み
リアルタイム モブプログラミング	✳常にチームの最善を尽くすことができる ✳結果だけでなくプロセスを評価することができる	✳その場の熱によって誤った判断をしてしまう可能性 ✳局所最適に陥る可能性
レビュー Re-view	✳冷静に見ることができる ✳全体最適を考えてフィードバックしやすい	✳[illegible]度が生まれやすい ex. イマイチだけど時間がないから… ✳結果のみで読み取れない部分を知るためには、コミュニケーションが必要

補完関係

@TAKAKING22

我々のチームにおけるレビューの位置づけ



レビュー

- ふりかえり
- 自分達の仕事の評価
- もっとよくするには

@TAKAKING22

まとめ

- モブプログラミングを初めて、
  それまでの工程としてのレビューは必要がなくなった
- Re-viewとFirst-view
- リアルタイムにもレビューにも強み・弱みがある
- 自分たちに合ったレビューを再定義

レビュー再定義

Photo by Jantine Doornbos on Unsplash

レビュー再定義

- 自分たちの知っているレビューを見直す
- レビューの強みを活かすことができているか
- レビューではないプロセスの方が解決しやすいことはレビューで解決する必要がない

プロセスを改善する

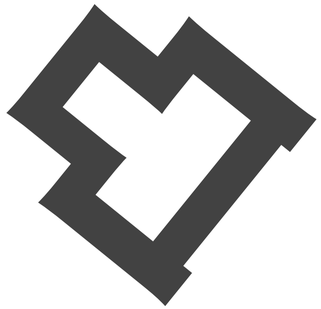
- プロセスを改善するという意識は大切
- しかしそのプロセスでの解決に固執しなくてもよい
- 目的を達成するために、そのプロセス自体をやめるという選択肢も常に頭に入れておく

完璧にやらなくてはいけないからの解放

Photo by Pierre Herman on Unsplash

無限に湧いてくる
やらなくてはいけないこと

有限のリソース



@TAKAKING22

完璧にやらなくてはいけないという幻想

- 完璧もやらなくてもいけないも考えたらキリがない
- 目指したところでたどりつけるものではない
- たどりつけないのでツライしギスギスする

完璧にやらなくてはいけないからの解放

- ROIを考えて自分たちにできることをする
- これでダメなら仕方がないという割り切り
- 最善を尽くしても失敗したときはそこから学ぶだけ

Photo by Lucas Clara on Unsplash

変化に適応する

- 他人のうまくいったことが、
  自分たちにそのまま当てはまるとは限らない
- チームの練度や状況によって最適解は変化する
- 一度定義したら終わりではない
- 変化に適応し続けること

情緒を大切にする

Photo by rawpixel on Unsplash

情緒を大切にする

- 小手先のテクニックだけでどうにかできると思わない
- 人間であることを楽しむ
- 完璧ではないけれど成長することができる
- 楽しいと楽しくないのインパクト

人間らしいレビュー

Photo by rawpixel on Unsplash

あなたのレビューは楽しいですか？

@TAKAKING22