“ふりかえり”を用いた
レビュー方法の改善

2016年10月13日

インテック
木村　慎吾

本日のテーマ

- ふりかえり＋レビュー方法の改善
  - 開発方法（プロセス）について
  - 開発レビュー（単体開発）の改善について

自己紹介

■名前
　木村　慎吾

■所属
　インテック
　ネットワーク＆アウトソーシング事業本部
　ネットワークソリューション部

■主な業務
　結人・束人　開発リーダ

インテックと富山

富山駅北側に私たちの本社があります

開発対象について

・INTECのパッケージ商品（自社開発）
・アイデンティティ管理分野

結人　2008年9月リリース

（2008年3月開発スタート）

束人　2009年1月　リリース
（2008年6月開発スタート）

開発チームの特長

- チーム構成
  - 最初は4名から（最大10名ほど）
  - 開発経験者は私のみ
- 開発プロセスについて
  - XPをもとにしたアジャイル開発

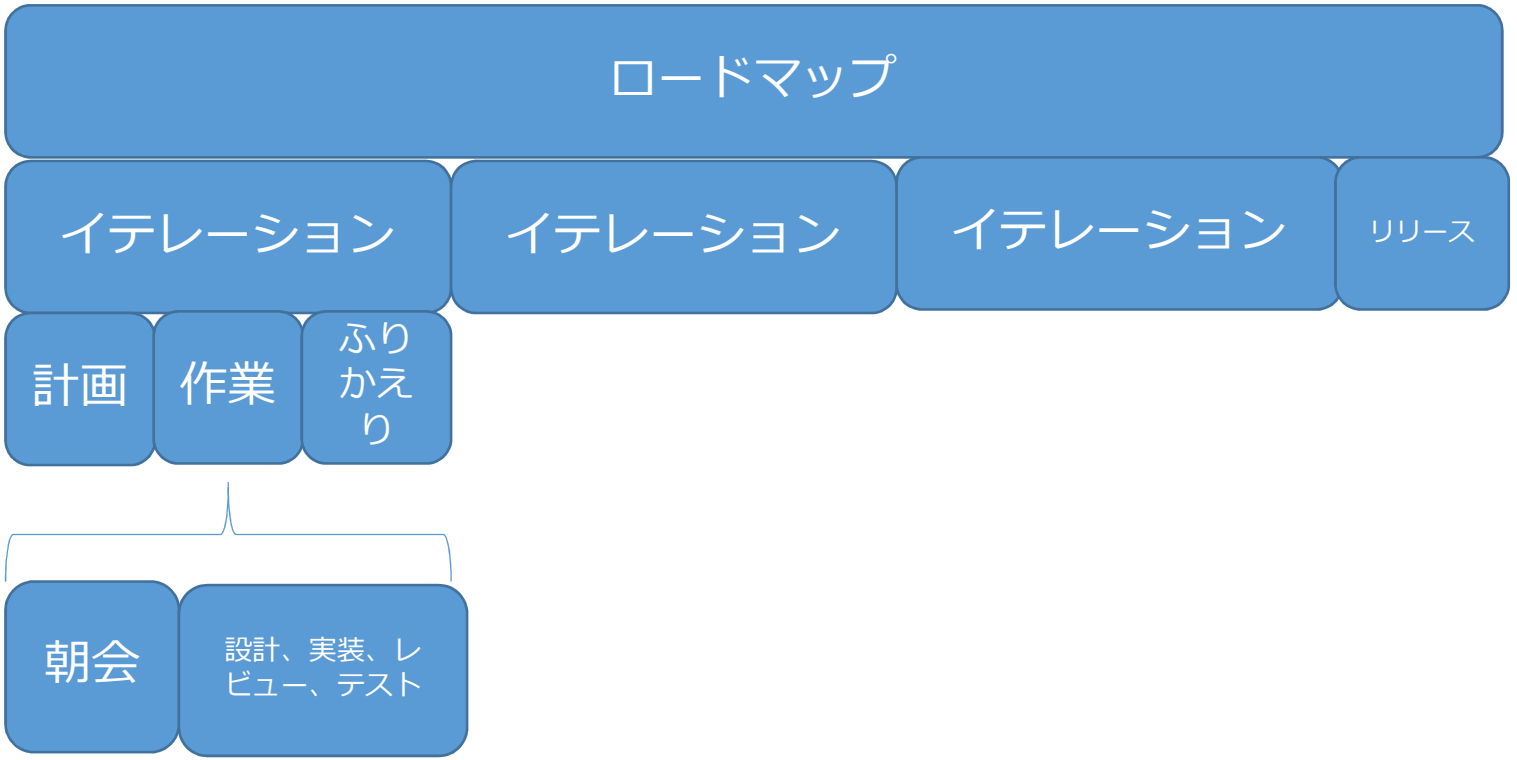
精鋭部隊ではなくて、ごく一般的なメンバで構成したチームです。

①開発プロセスについてご紹介

②レビューの改善についてご紹介

③まとめ

日々の開発について


ロードマップ
イテレーション
イテレーション
イテレーション
リリース
計画
作業
ふりかえり
朝会
設計、実装、レビュー、テスト

開発におけるGoogプラクティス！

ふりかえり

ふりかえりについて

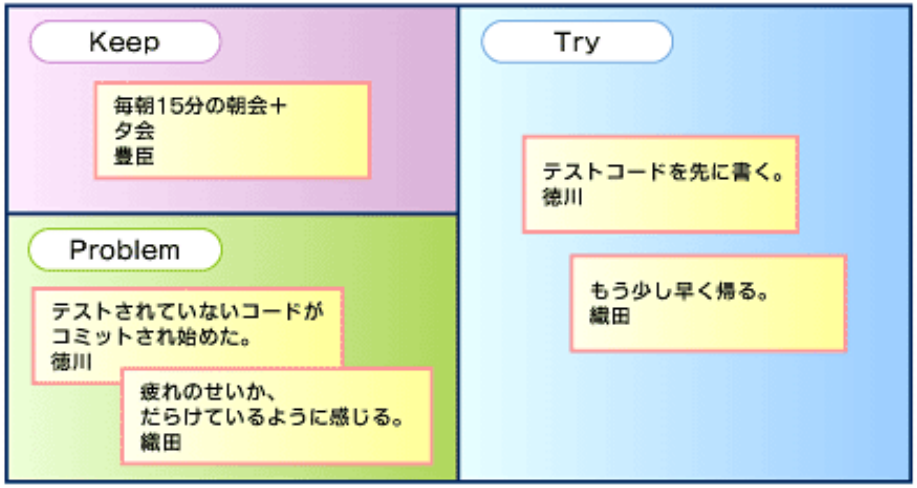
ふりかえりの方法について

「**KPT**（Keep，Problem，Try）」と呼ぶ手法

以下のようなプロセスを経て**ふりかえり**，行動する。

(1)行動する
(2)行動の結果や状況を思い出す
(3)集めた結果や状況に対して評価をする（Keep, Problem）
(4)評価が良いもの（Keep）は今後も続け，できれば名前をつける
(5)評価が悪いもの（Problem）は対応策（Try）を考える
(6)対応策を行動に移す（1へ戻る）

参考：*http://www.thinkit.co.jp/free/article/0610/9/1/*

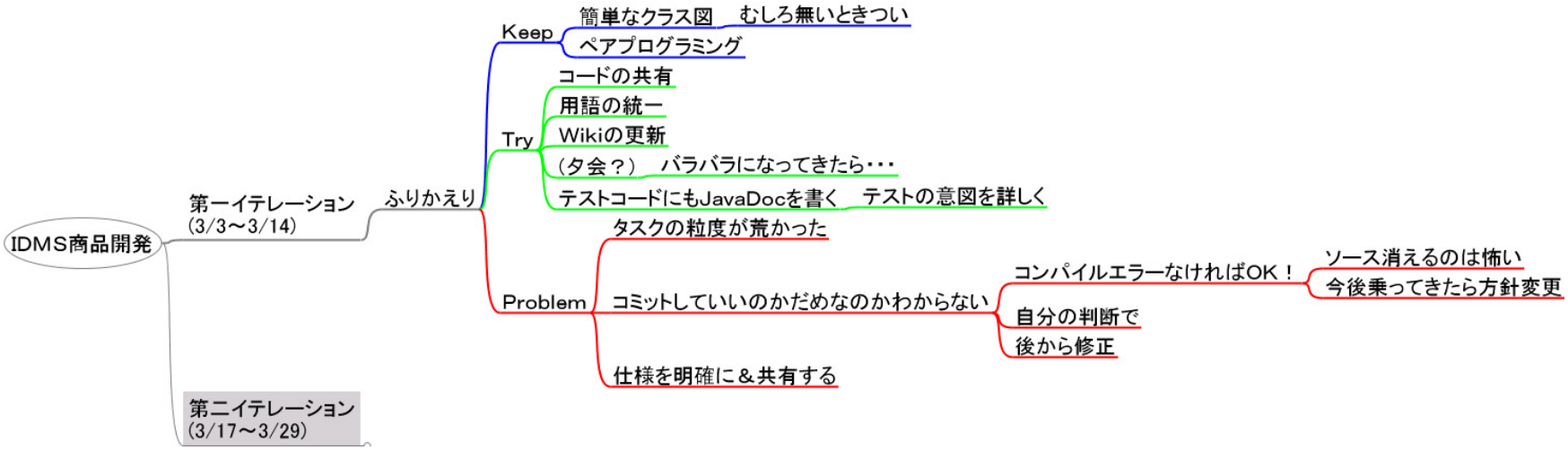


http://www.atmarkit.co.jp/farc/rensai/pl06/pl06.html

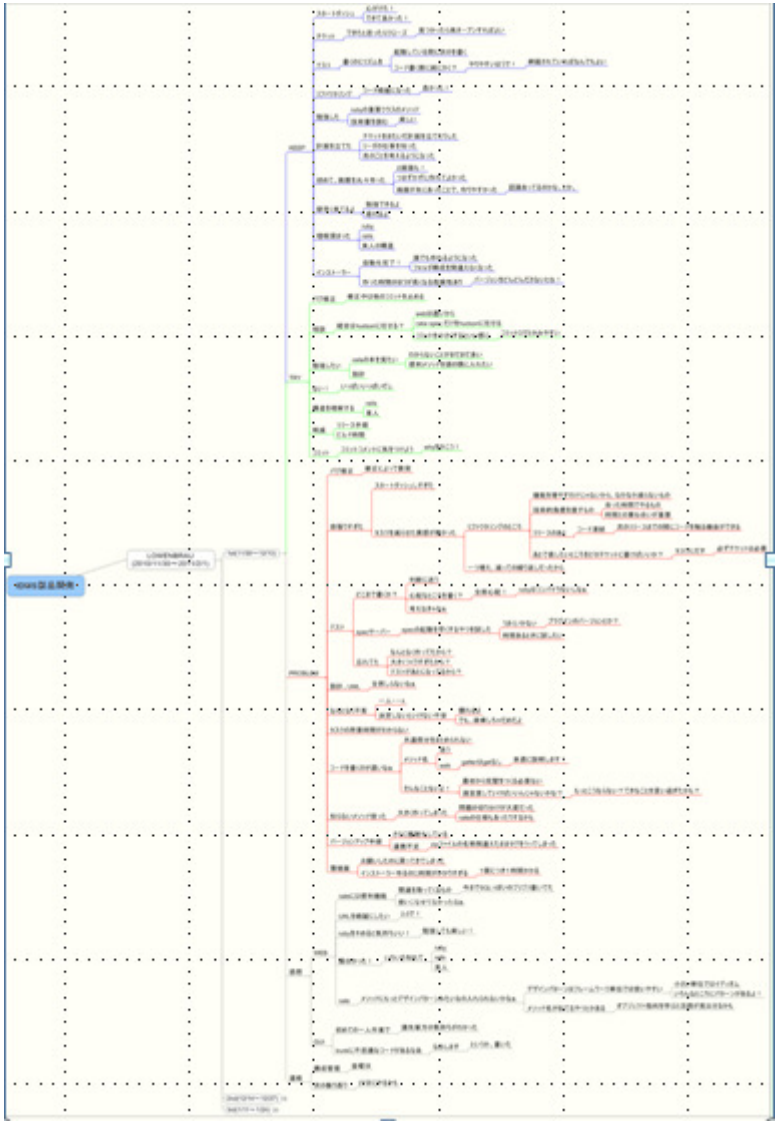
私たちの“ふりかえり”

- ふりかえりのルール
  - KPT方式を基本的に採用
  - 工夫としては＋感想の枠もある
- 気を付けていること
  - Tryはできることからやる。1つでもできればOK
  - 残Tryは大きな課題としてまとめる
- その他
  - イテレーション以外にもふりかえりを実施

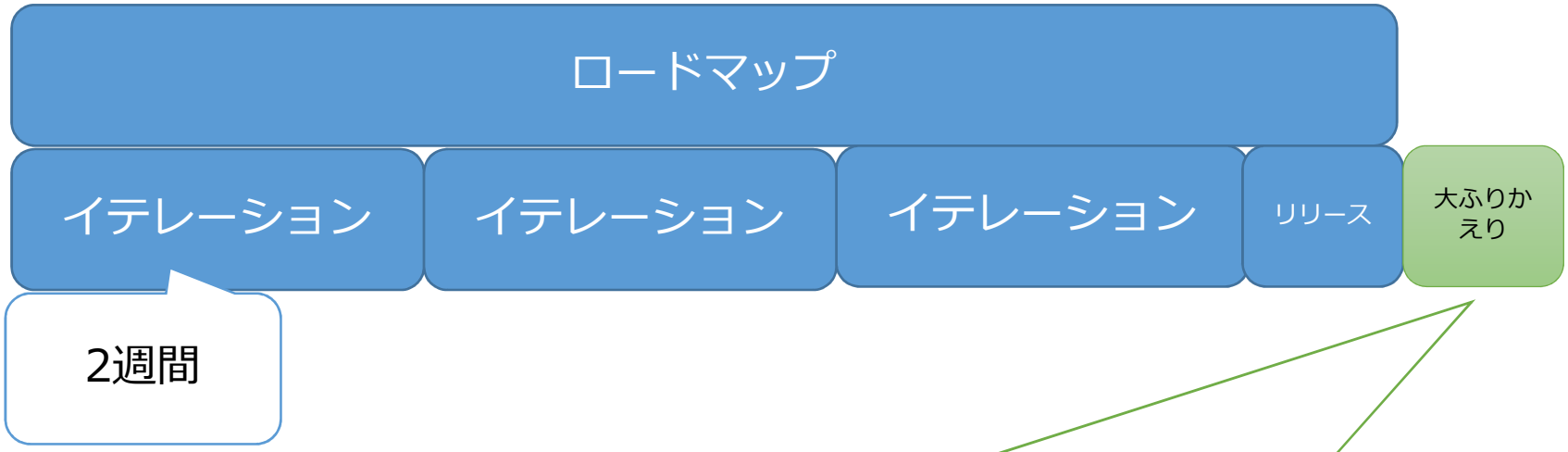
初回のKPT


IDMS商品開発
第一イテレーション
(3/3～3/14)
ふりかえり
Keep
簡単なクラス図
むしろ無いときつい
ペアプログラミング
Try
コードの共有
用語の統一
Wikiの更新
(夕会？)
バラバラになってきたら・・・
テストコードにもJavaDocを書く
テストの意図を詳しく
Problem
タスクの粒度が荒かった
コミットしていいのかだめなのかわからない
コンパイルエラーなければOK！
ソース消えるのは怖い
今後乗ってきたら方針変更
自分の判断で
後から修正
仕様を明確に＆共有する
第二イテレーション
(3/17～3/29)

いまのKPT



大ふりかえりについて



リリース後に
全体のふりかえりを実施

大ふりかえりの特長

- 残課題の整理
  - 大きな課題などを見直してやりっぱなしにしない
- 全体で考える
  - 普段のふりかえりより大きい視点で考える

大ふりかえりの例

- 10:00～11:45　営業より今後の戦略についての説明
  目的：営業より今後の戦略について認識あわせ
- 13:00～14:20　みんなで1年間のふりかえり
  目的：**1年間の振り返り（タイムラインを使ったふりかえり）**
- 14:30～16:20　プロセス＆機能に関するふりかえり
  目的：プロジェクト全体を通してふりかえる（KPT）
- 16:20～17:00　**LT大会**（一人持ち時間が5分のプレゼン大会）
  目的：熱い思いを声にだそう！　（プレゼンの練習も）
- 18:00～　ホントのふりかえり（懇親会）

①開発プロセスについてご紹介

②レビューの改善についてご紹介

③まとめ

改善の方法

- ふりかえりで出たProblemをTryで改善

※ふりかえりを常にやり続けているため、
変化は常に起こっているのですが、
わかりやすくするため、3段階に分けて説明

エピソード１：開発開始時

開発チームの状況

- 開発経験がない
- とにかく不安

開始時の品質に関する課題

- 私以外は素人なので、そもそもキホンがない
- チェックできるのは私だけ

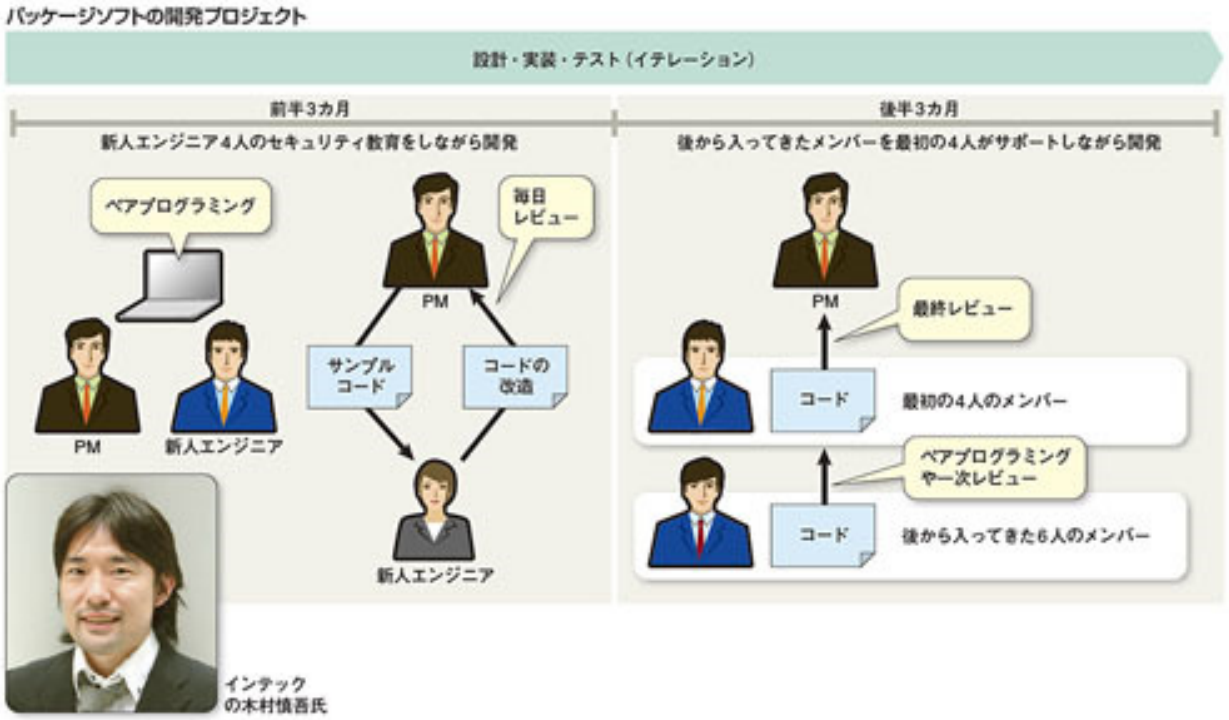
ふりかえりのProblem

- 個人的な作業に対する内容が中心

当初の施策

- レビューは私が実施
  - メンバの開発速度が遅かったので対応できた
- 基礎力の強化
  - 勉強会の開催

具体的なレビュー方法



http://itpro.nikkeibp.co.jp/article/COLUMN/20101208/354997/

具体的な勉強会の方法

主な方法

- 読書会

実施方法

- お昼休みを中心に実施
- そのほかに1週間に１時間別途時間を確保

エピソード２：開発中期

開発チームの状況

- 開発速度が向上
- 勉強会などにより品質の大切を理解

ふりかえりのProblem

- レビュー待ちの問題
  - 開発速度の向上で決まった担当者だけでは追いつかない

- 品質に関する問題
  - 自分の作ったものへの不安
  - 誰かの作ったものとの差異が気になる

変更されたレビュー方法

- 持ち回り制
  - 朝会でレビュー者を決める
  - レビューOKで完了

- 具体例
  - かんばんにレビューレーンを作り担当者を割り当てる
  - DoneはチケットのレビューOKの記載で確認

エピソード３：現在

開発チームの状況

- チーム体制が変わった
  - メンバの出入りが多くなってきた
- ルールは引き継がれているが、漏れが多くなる
  - 暗黙知が多くなってきた

ふりかえりのProblem

- 明確なバグ以外の指摘を躊躇してしまう
- 認識できていないルールが多くなってきた

改善の課題

- 全部をドキュメント化できない
- コミュニケーションを取るという気合だけではできない

改善について

- すぐに解決できないTry（課題）となった

解決策の発見

- 新しい手法に出会う
  - Pull Request方式（Github）
  - ツールで問題を解決

Pull Requestとは

- プログラムなどを構成管理システムへコミットする過程で内容を確認できる仕組み

Pull Requetのメリット

- プロセスに自然と組み込める
- レビューを通して暗黙知が共有しやすい
- コミットへの恐怖感解消

Pull Request導入への課題

- ツールの変化
  - 既存のものを変えることは難しい
  - できるところから試していく
    - 派生開発で利用
    - 体験することが大切

コミットへの恐怖感問題

- コミットすることに抵抗あるという意見がある
  - 見られるのがはずかしい
  - もう少し自分でなんとかしたい
- 管理面で気になったこと
  - 進んでいるというが、進捗がないにようにみえる
  - 金曜日の夕方にコミットが多発する

解決策

- みんなで確認できることのメリットを共有

①開発プロセスについてご紹介

②レビューの改善についてご紹介

③まとめ

まとめ

ふりかえりについて

- メンバの成長を促すには効果的
- プロジェクト全体など大きな粒度でも対応可能
- やり続けることが大切

レビューについて

- メンバの成長によって対応は異なる
- ツールの進化によってレビュー方法も進化できる
- プロセスに自然に組み込める仕組みづくりが大切