プロジェクトX(p)
- 現場主導のアジャイルプラクティス実践による品質改善事例 -

ソニー㈱
PCアプリケーションソフトウェアセンター 設計部
小原　優
Yu.Kohara@jp.sony.com

本日の内容

- 対象PJ紹介
- 課題
- 改善活動方針
- 改善活動結果 (品質評価)
- PJとプラクティス群
  - XP実装
  - Iterative Postmortem
- まとめ

悪循環(デスマーチ)で燃え尽きやすい、実装担当者向けの内容を中心に発表。

対象PJの特徴

- **音楽ファイルを取り扱うPC Middleware**
  - Client Application (22**種類**)
  - **配信サービス** (3**種類**)
  - **接続可能機器** (11**種類**)

- **歴史/規模** ※2006/Dec**現在**
  - **1999年～現在, 約1000[KLOC]**
  - **全世界へ展開**

- **開発言語/環境** ※2006/Dec**現在**
  - C++ / C, VC7 compiler
  - Vista / XP / 2K / ME / 98SE (11**種類**, **ローカライズ版**)

“歴史”も”規模”も”配布数”も”組み合わせ”も、巨大。

対象PJメンバの特徴

- **構成** ※2007/Aug現在
  - PM：1[人]
  - PL：1[人]
  - **設計者**：10[人]（**大きく3チーム構成**）
  - **設計内検証**：8[人]

- **特徴**
  - **メンバ変更がほぼ無い**
  - **知識/実装の初心者が居ない**

メンバが変わらないため、色々と実施しやすい環境といえる。

品質向上活動の産声

何故自分だけが
こんなに大変

リース後に
したい

XP!!

ムに
負荷が集中…

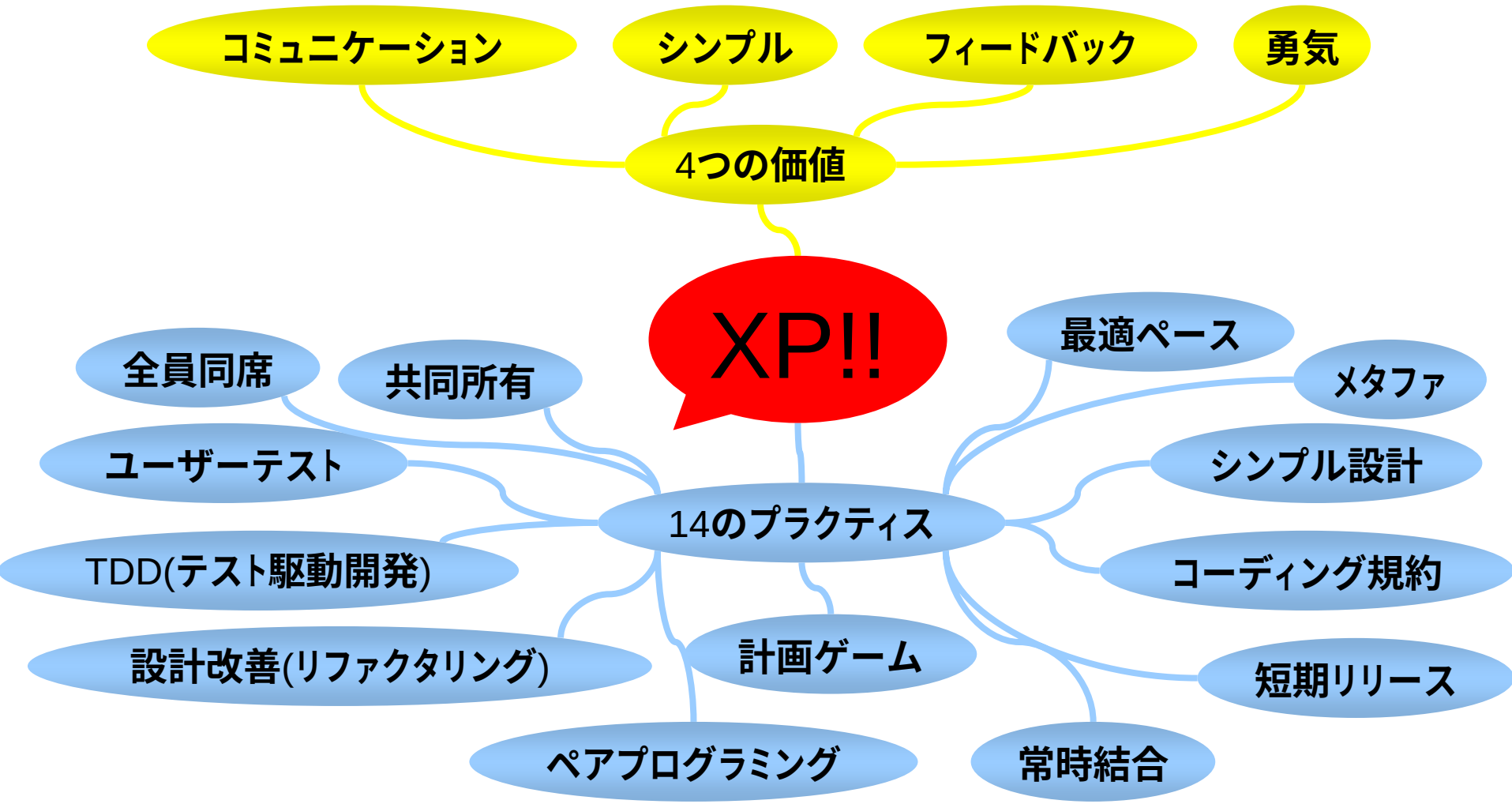
現場の悩みは尽きない…。

XP導入について

- 導入が容易で低コスト
  - 具体的なプラクティスが示されている
  - 導入費用が不要
- 思想が素敵
  - ソフトウェアは人間が作成するという事を意識
- 裏付けのあるプラクティス
  - 実装経験者なら直感で良いと感じる内容

安い / 早い / 旨いが決め手！必要なのは、やる気のみ！

XPとは

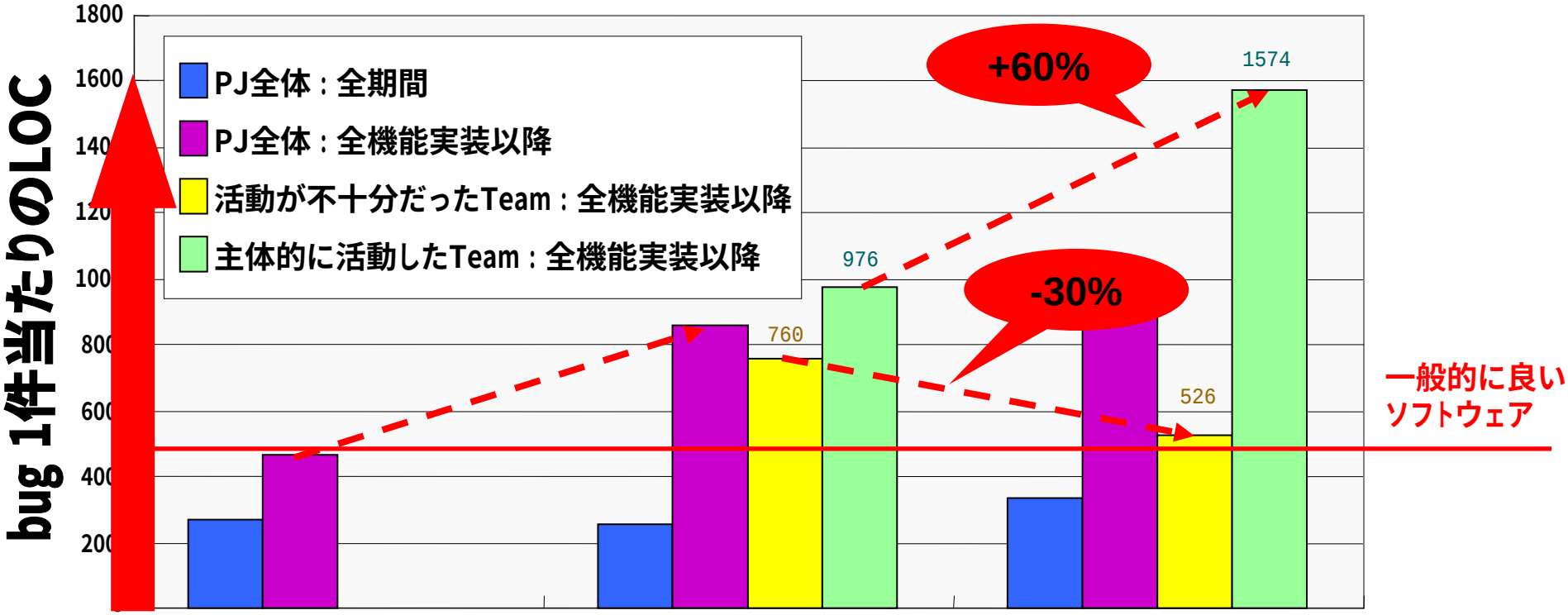


超反復型開発で、人を意識している。詳細は、XP-JPを参照の事。

品質向上活動の目標

- 出来るだけ早く”間違い”に気付きたい
  - 人間への依存が大きいので、”間違う”のは当然
- 設計時点での品質組込
  - 設計内検証での不具合指摘を無くす
    - 不具合1件あたりのコスト：チーム全体で最小0.25[人日]
- 独りで抱え込まない/込ませない
  - タスクもコードも不具合も、全てチームの物
- 常に改善
  - 皆の知恵でプロジェクトもメンバも変化し続ける

品質向上活動の結果

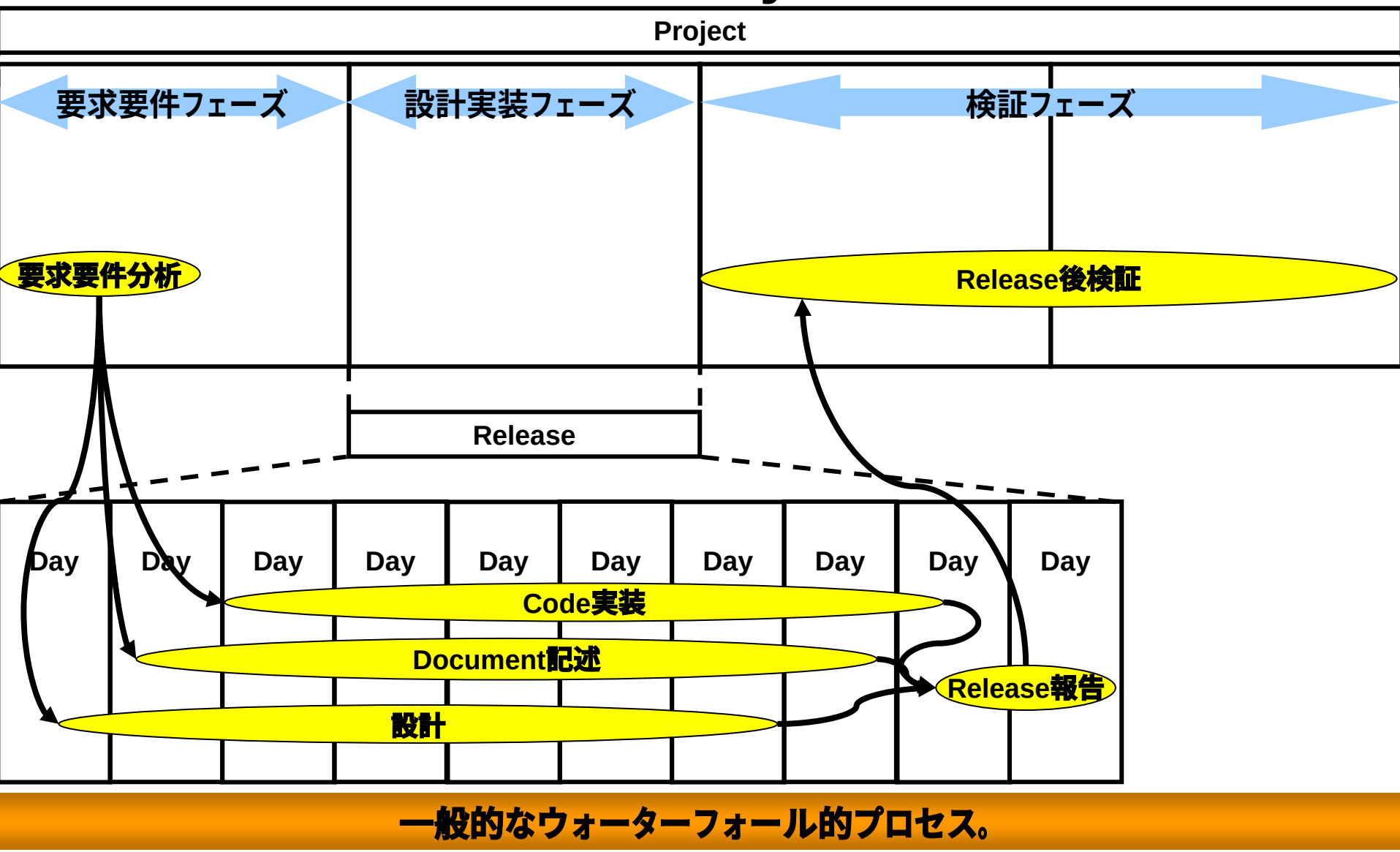


Ver.XX	Ver.YY	Ver.ZZ	ヴァージョン
26名	22名	18名	開発人数
4.9ヶ月	5.5ヶ月	3.9ヶ月	開発期間
120KLOC	126KLOC	61KLOC	追加変更LOC

プラクティスの効果が見える！

開発作業
改善活動

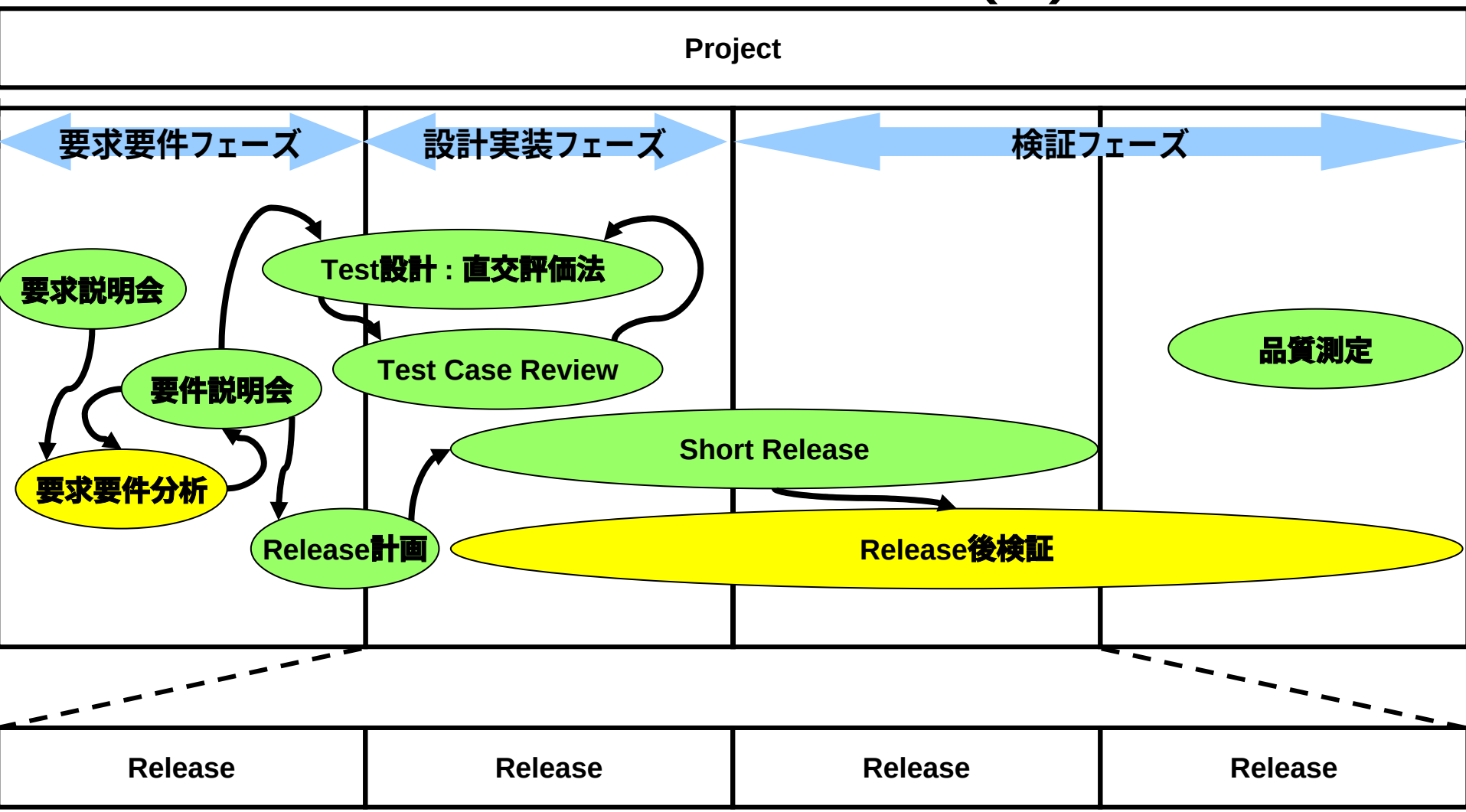
一般的なProject Flow



一般的なウォーターフォール的プロセス。

開発作業
改善活動

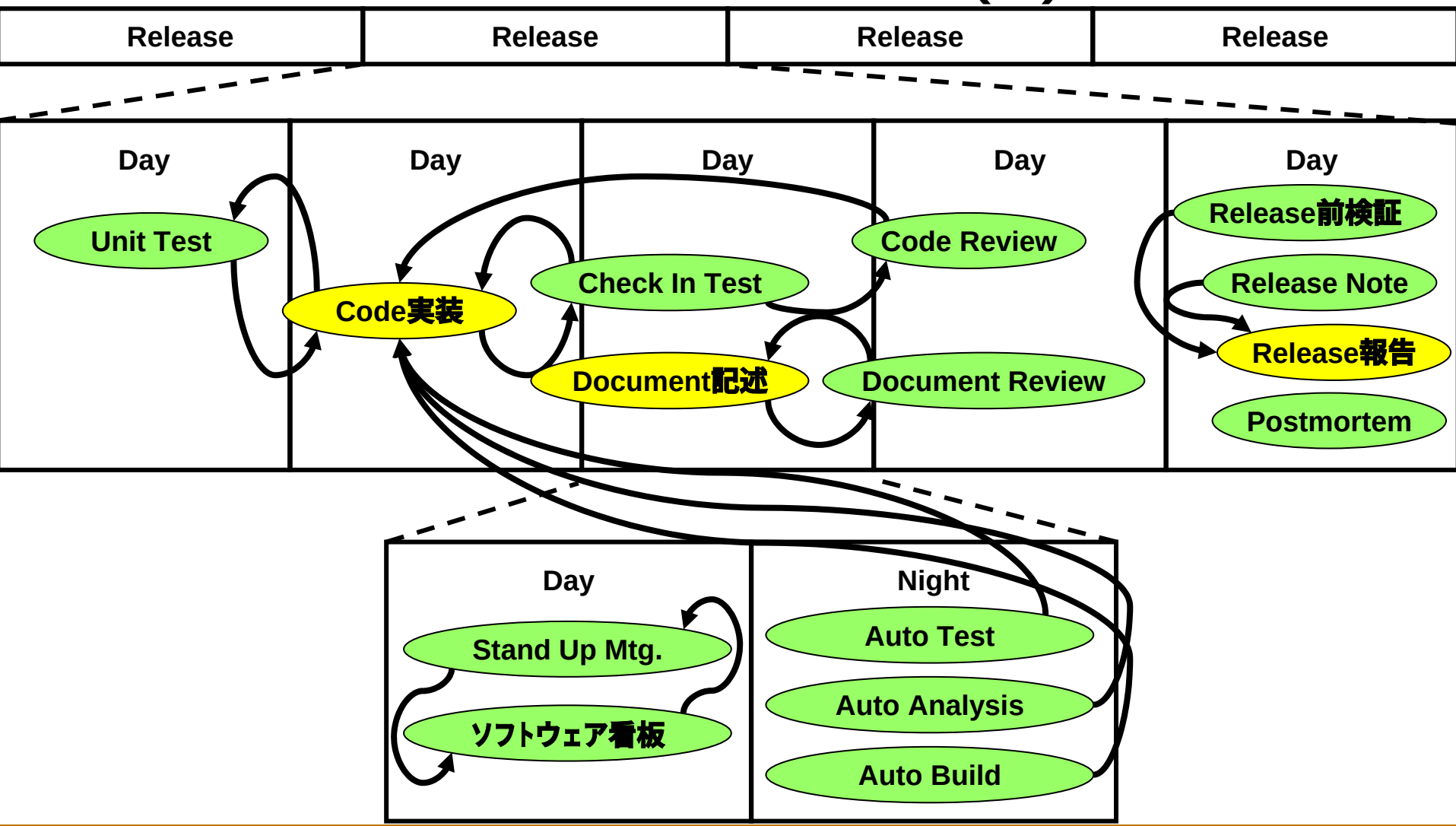
本PJでのFlow (1)



設計と設計内検証で強調して、見える化を促進！

開発作業
改善活動

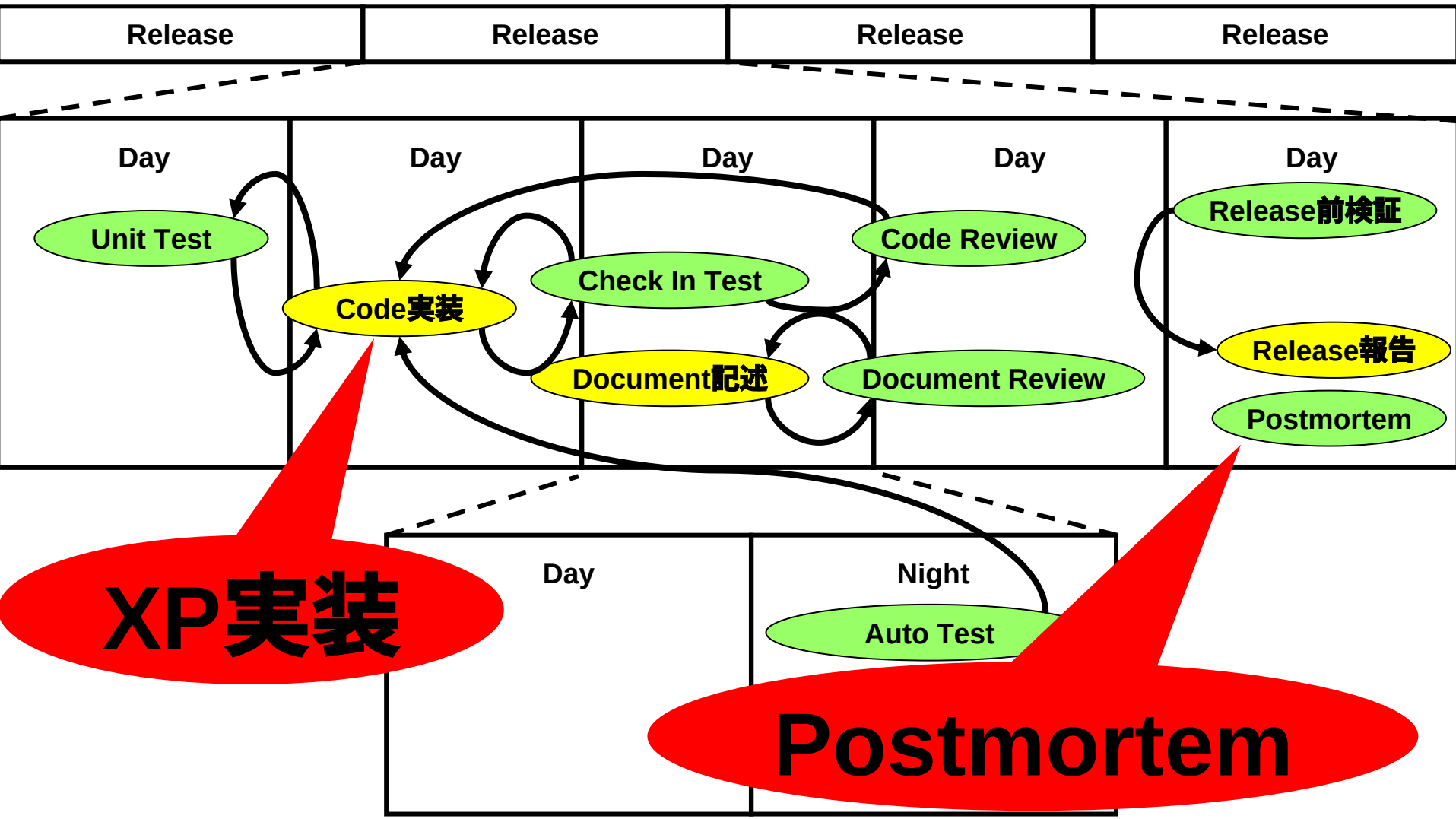
本PJでのFlow (2)



小さい単位で、繰り返す事で、見える化を促進！

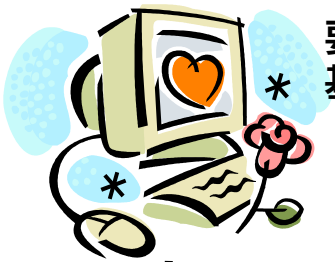
開発作業
改善活動

今回の焦点



忙し過ぎる実装担当者に効く部分。

XP実装 - 流れ -



要件
基本設計

8:Code Check In

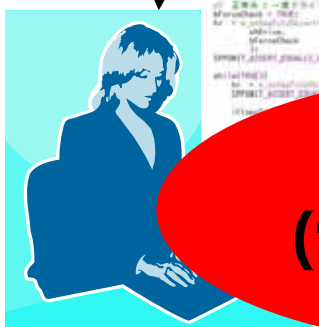
10:毎晩実行

自動検証

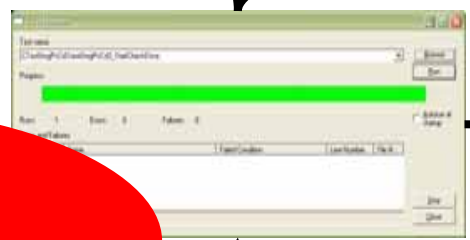
9:Auto Testへ組込

11:Release前検証で実行

1:Unit Test実装

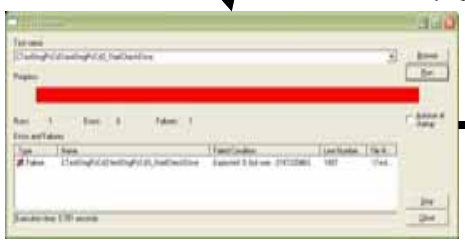


TDD
(テスト駆動開発)



4:Unit Test実行
7:Unit Test実行

2:Unit Test実行



3:Code実装

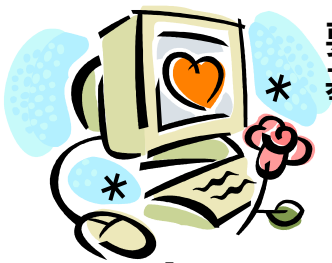
リファクタリング

5:Review

6:Refactoring

設計者が動作を保証する事が出来ないソフトウェアは、大きな手戻りが発生する。

XP実装 – テスト駆動開発(TDD) -

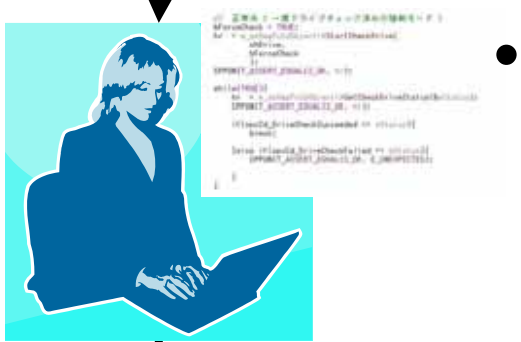


要件
基本設計

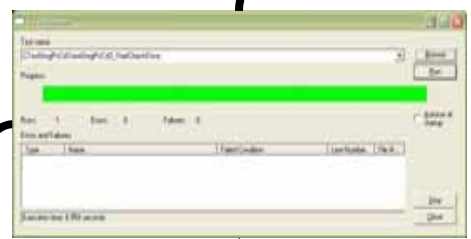
- **設計者がTest作成**
  - **仕様を明確に出来る**
  - **言葉でなくCodeで表現**

6:Code Check In

1:Unit Test実装



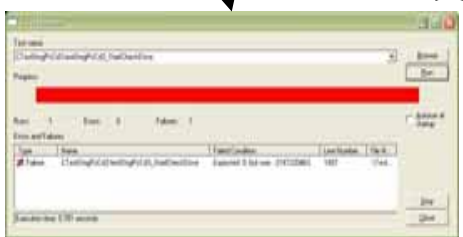
- **不具合発生から改修までの無駄が無い**
  - **自己完結**



4:Unit Test実行

- **影響範囲の確認をしながら実装出来る**
  - **全機能の動作確認**

2:Unit Test実行

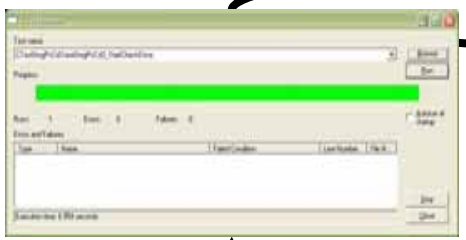


5:Code実装

3:Code実装

Test実装→Code実装のリズムや、Green Barの快感など、体感で良いと感じる。

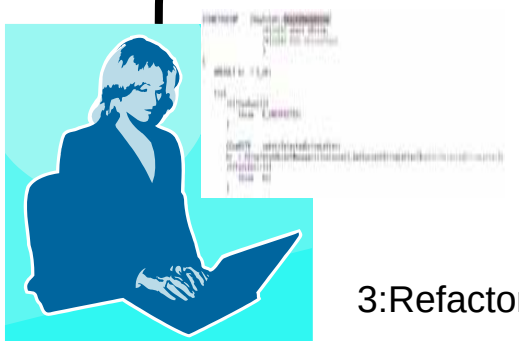
XP実装 – リファクタリング -



1:Unit Test実行

4:Code Check In

- **不具合検出率**1.92[**件**/h]
  - **Testと比べて検出効率が良い**
    - **計測結果** : Review 1.92[**件**/h], Test0.03[**件**/h]
    - **一般事例** : Review 0.7[**件**/h], Test 0.12[**件**/h]
- **保守が容易な**Code**になる**
  - **人に見られる事を意識**
- **知識の共有が出来る**
  - **自分以外の人もメンテ可能**

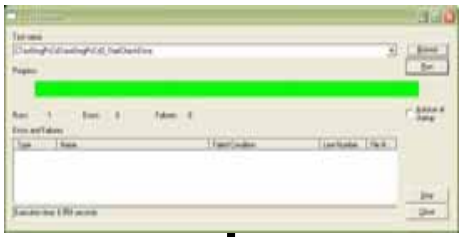


3:Refactoring

2:Review

Unit Test/Code Reviewが存在してこそリファクタリング。

XP実装 – 自動検証 -



- Test作成コストが不要
  - TDDのTestを再利用
- 人にかかるコストが小さい
  - 0.5人時/1OS
  - Test環境の準備のみ
- 不具合潜在時間が短い
  - 1日の差分だけを確認

1:Auto Testへ組込

2:毎晩実行

3:Release前検証で実行

1つ1つのAPIへのTestが全て実行される。TDDで作成したものを最大利用。

XP実装 - 所感 -

- 良かった点
  - TDDで、仕様不具合に気付く
  - Code修正の不安が激減
  - Release後の不安が無い
- 悪かった点
  - Test実装コストが増加
    - 1つのAPIに対して1[h] (含む正常系/異常系)
  - 簡単な関数(ロジック)の場合、ムダ感あり

Test/Review無しでは生きられない体になる。

XP実装 – 経験で得たこと -

- Testの品質
  - 個人依存
- Testのリファクタリング
  - 放置するとスパゲッティ状態
- 手動検証は必要
  - 人間でなければ気付けない不具合も存在
- 納期が短い場合
  - Testが書けない
  - 後で作成するのは、相当な気合とコストが必要

ソフトウェア開発で最も高コストとなるのは保守。ペアプログラミングで補完出来る箇所も。

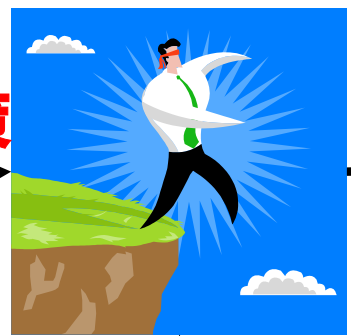
Postmortem - 流れ -

Keep(良かった事)

Try(次回挑戦)

成功

解決策



失敗

Problem(悪かった事)

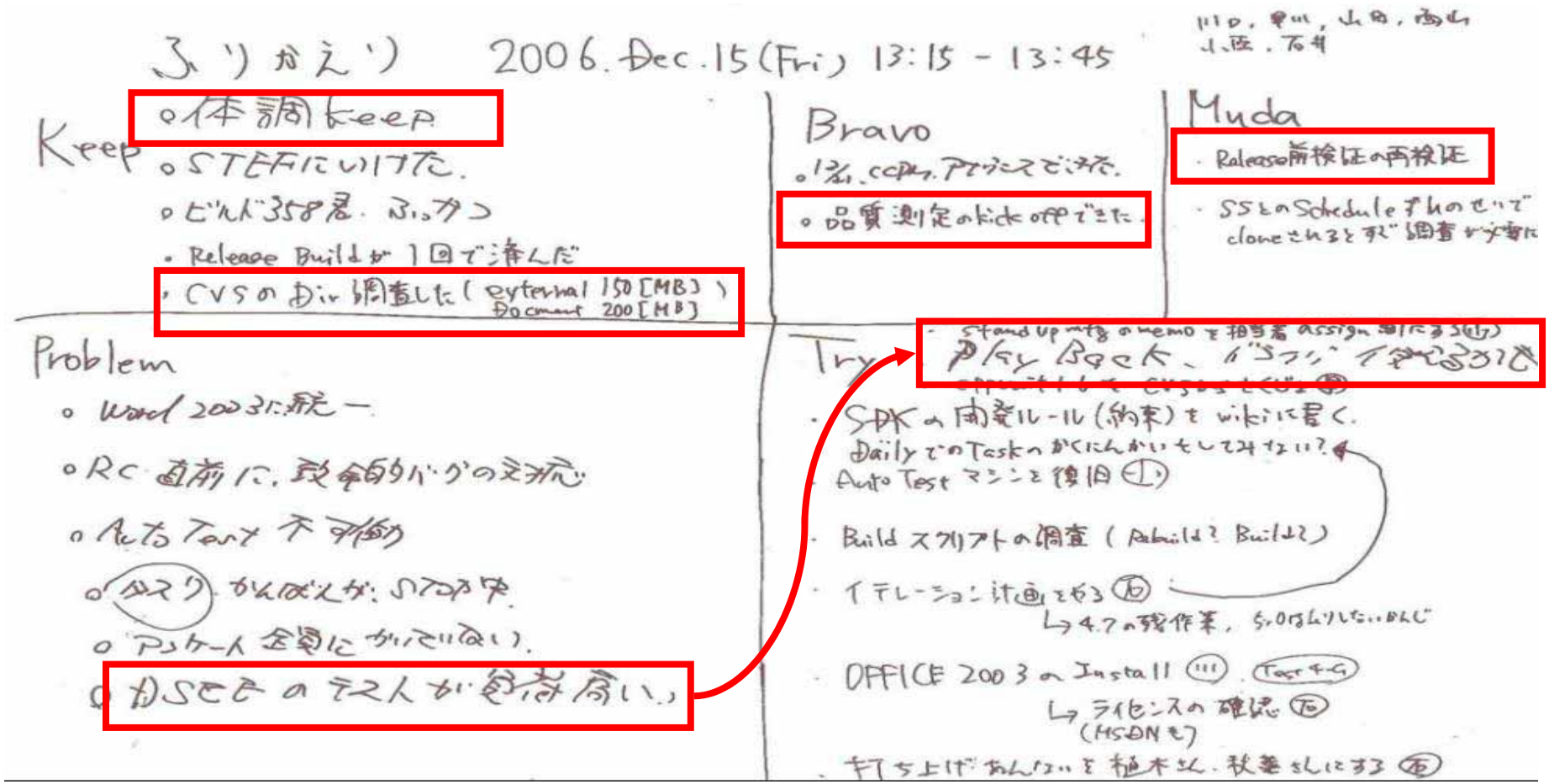
アイディア

問題

見える化だけでは改善されない。行動するためにタスクにする事が重要。

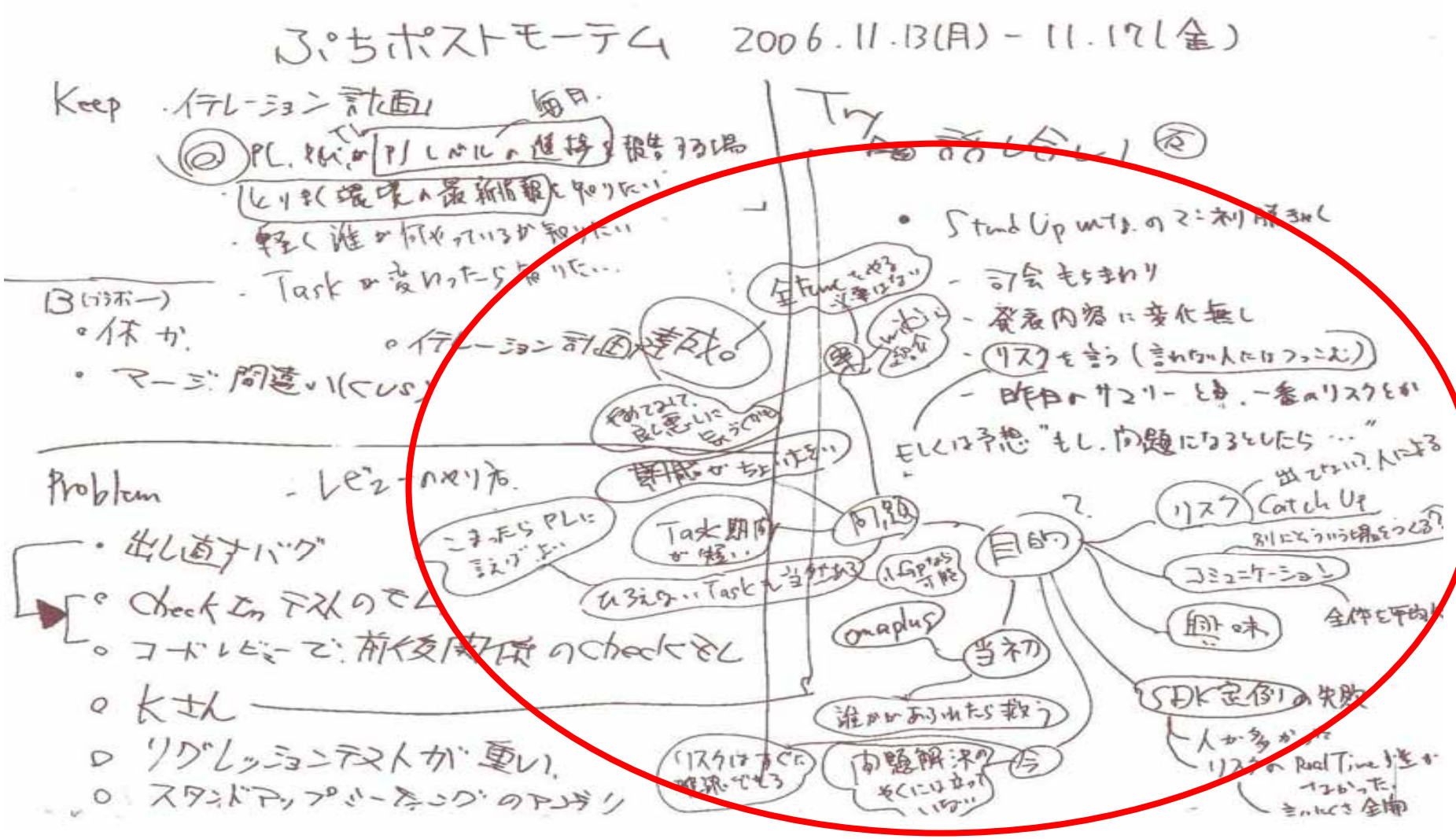
Postmortem – 実例 -

全員発表 **毎週30分** **Try担当者Assign！** **業務外ネタOK！**



事件は現場で発生している！押し付けでは無く、担当者から発生する活動は、継続する。

Postmortem – **実例** -



盛り上がってくると、話題が1つに絞れて、問題意識の共有が出来る (場合もある) 。

Postmortem – 実例 -

- **Problem**：自動buildが遅い
  ⇒**Try**：ヴァージョン管理ツールの整理
  ⇒**Try**：自動build scriptの効率改善
  ⇒**Keep**：複数マシンによる分散build

- **Problem**：Stand Up Mtg.に無駄感あり
  ⇒**Keep**：個人発表を止め、全体状況のみ共有
  ⇒**Keep**：議事録を取る
  ⇒**Keep**：開催時間が短くなった
  ⇒**Problem**：個人の状況が見えなくなった

皆で決めるので、取り組みが早く、実行に協力的。

Postmortem - 所感 -

- **良かった点**
  - **個人のProblemが全体を改善**
  - **Tryによって、”愚痴で終わり”にならない**
  - **親睦が深まる**
- **悪かった点**
  - **特に無し**

導入がとても簡単、且つ、その効果は絶大。

Postmortem – 経験で得たこと -

- **小さいTry**
  - **実施される**
- **硬い雰囲気**
  - **意見がでない (厳禁)**
- **実施間隔は短く設定**
  - **1ヶ月でも“気付き”を忘れてしまう**
- **個人のKeep**
  - **全体のKeepになりにくい**

色々な失敗を経て、今の形式で実施。

本日のまとめ

- XP
  - **安い･早い･旨い･人間を意識**
- XP**実装**
  - TDD, **リファクタリング**, **自動検証**
- Postmortem
  - Keep, Problem, Try
- **効果あり**
  - **不具合低減**, **保守性向上**

ボトムアップの施策は、間違い無く的を射ている！

品質向上活動をするにあたって

- 先ずは出来ることから
  - 拙い点はどんどん改善
- “出来ない理由”を探さない
  - “諦めたらそこで試合終了だよ”
- **継続的活動が大切**
  - “急がば回れ”

今後の課題

- 進捗管理
- 要求要件分析の強化
- テストケースの爆発
- 知識の形式知化

まだまだやるべき事は盛り沢山！

References

- **アジャイル開発**
  - XP-JP (http://www.objectclub.jp/community/XP-jp/)
  - **プロジェクトファシリテーション**
    (http://www.objectclub.jp/community/pf/)
- **見える化**
  - **「見える化 (強い企業を作る見える化の仕組み) 、遠藤功著、東洋経済新報社」**
    (http://www.amazon.co.jp/exec/obidos/ASIN/4492532013/503-4122174-4204733)