あなたもできる：
「うまい、やすい、はやい」のソフトウエア開発

オージス総研のオフショア開発における
アジャイル・プラクティス導入事例

株式会社　オージス総研　技術部

クラウドインテグレーションセンター　茨木　良昭
アジャイル開発センター　藤井　拓、張　嵐

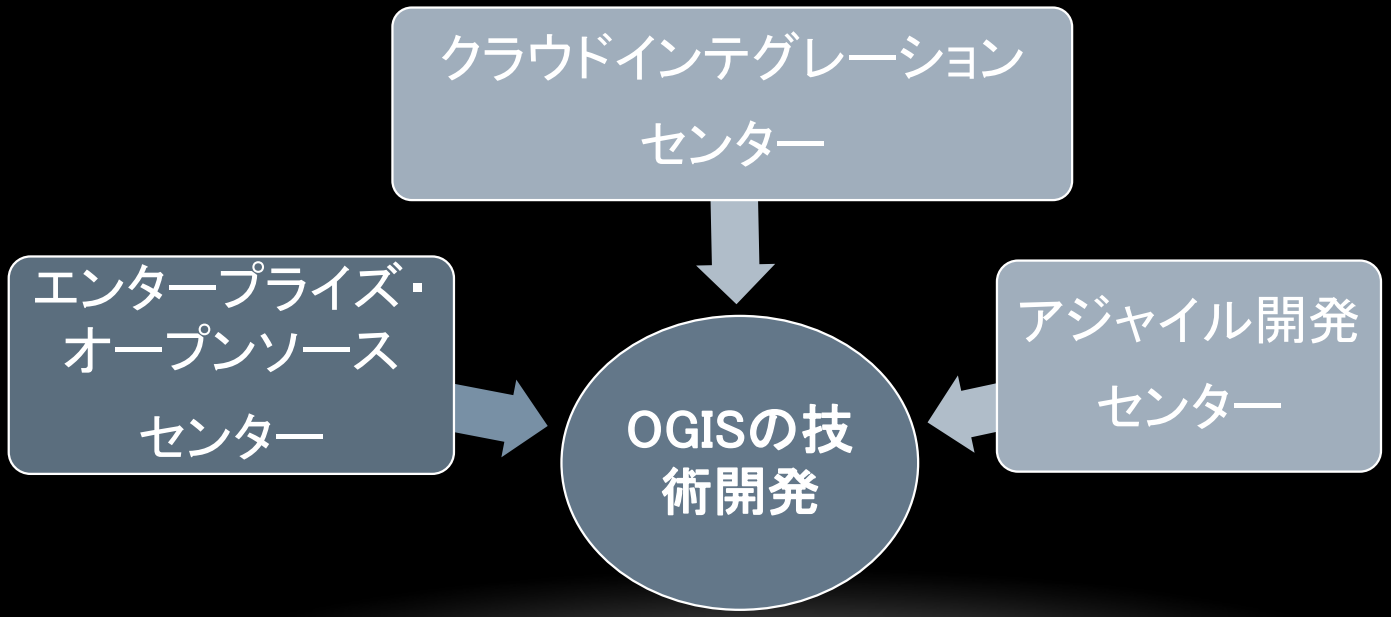
アジェンダ

1. 自己紹介
2. プロジェクト概要
3. 見えてきた課題
4. オフショア　＋　アジャイル開発
5. 結果
6. 今後の予定
7. まとめ

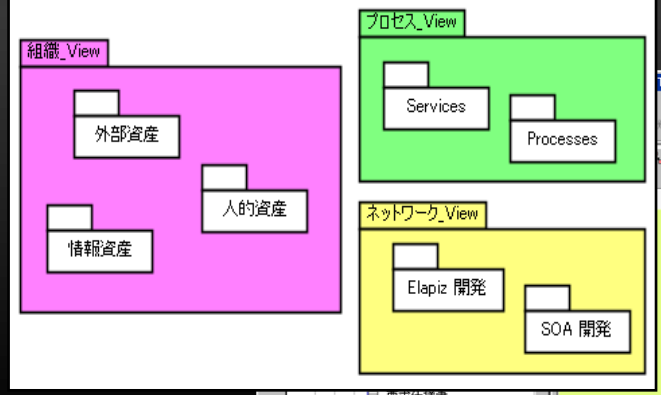
1. 自己紹介

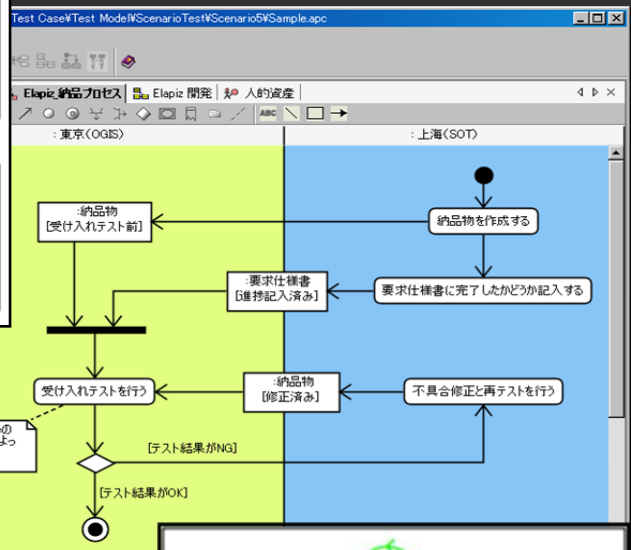
自己紹介

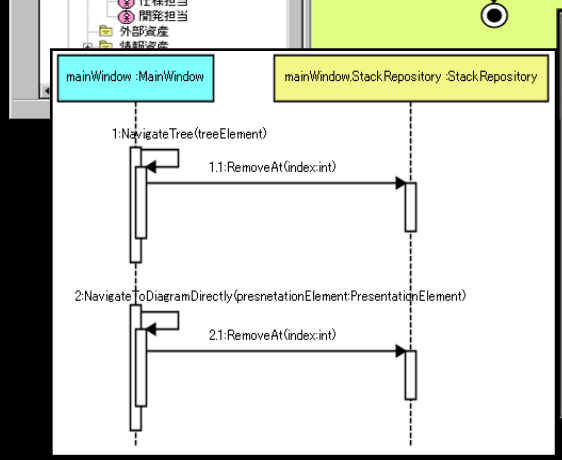
- (株)オージス総研に勤務
  - PM
  - 開発ガイドライン策定
  - オフショア開発(途中参加)←今ここ

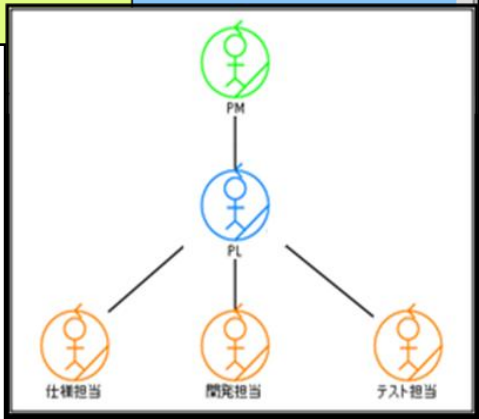


2. プロジェクト概要


組織_View
外部資産
人的資産
情報資産
プロセス_View
Services
Processes
ネットワーク_View
Elapiz 開発
SOA 開発


Test Case¥Test Model¥ScenarioTest¥Scenario5¥Sample.apc
Elapiz_納品プロセス
Elapiz 開発
人的資産
: 東京(OGIS)
: 上海(SOT)
:納品物
[受け入れテスト前]
納品物を作成する
:要求仕様書
[進捗記入済み]
要求仕様書に完了したかどうか記入する
受け入れテストを行う
:納品物
[修正済み]
不具合修正と再テストを行う
Milestoneの
重要性によっ
ては省略
[テスト結果がNG]
[テスト結果がOK]


mainWindow :MainWindow
mainWindow.StackRepository :StackRepository
1:NavigateTree(treeElement)
1.1:RemoveAt(index:int)
2:NavigateToDiagramDirectly(presnetationElement:PresentationElement)
2.1:RemoveAt(index:int)


PM
PL
仕様担当
開発担当
テスト担当

プロジェクト概要

- 開発内容
  - ElpaizBE（弊社製UMLモデリングツール）の開発
  - 開発環境（Visual Studio、C#、.NET）
- 委託先：上海欧計斯軟件有限公司（SOT）
- 委託期間：2005年～
- 委託範囲：基本設計以降
  - オージスは要求・受入検査のみ

オフショア開発委託先

- 中国の拠点となる上海オージス(SOT)と連携し、数多くのオフショア開発案件を実施している

＜SOTメンバー構成のご紹介＞

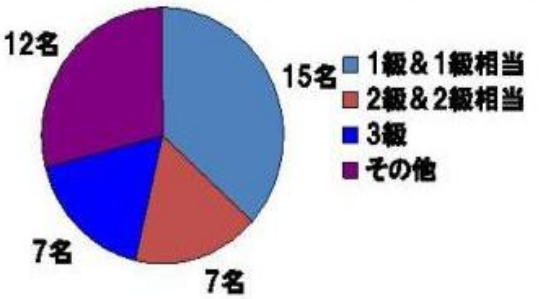
A Promise in Quality SOT

平均年齢 26．3歳

学歴	人数	割合
大卒	34	83%
短大	7	17%

経験年数

日本語能力　•現地採用SE41名中22名が日本語検定2級以上（全体の54%）



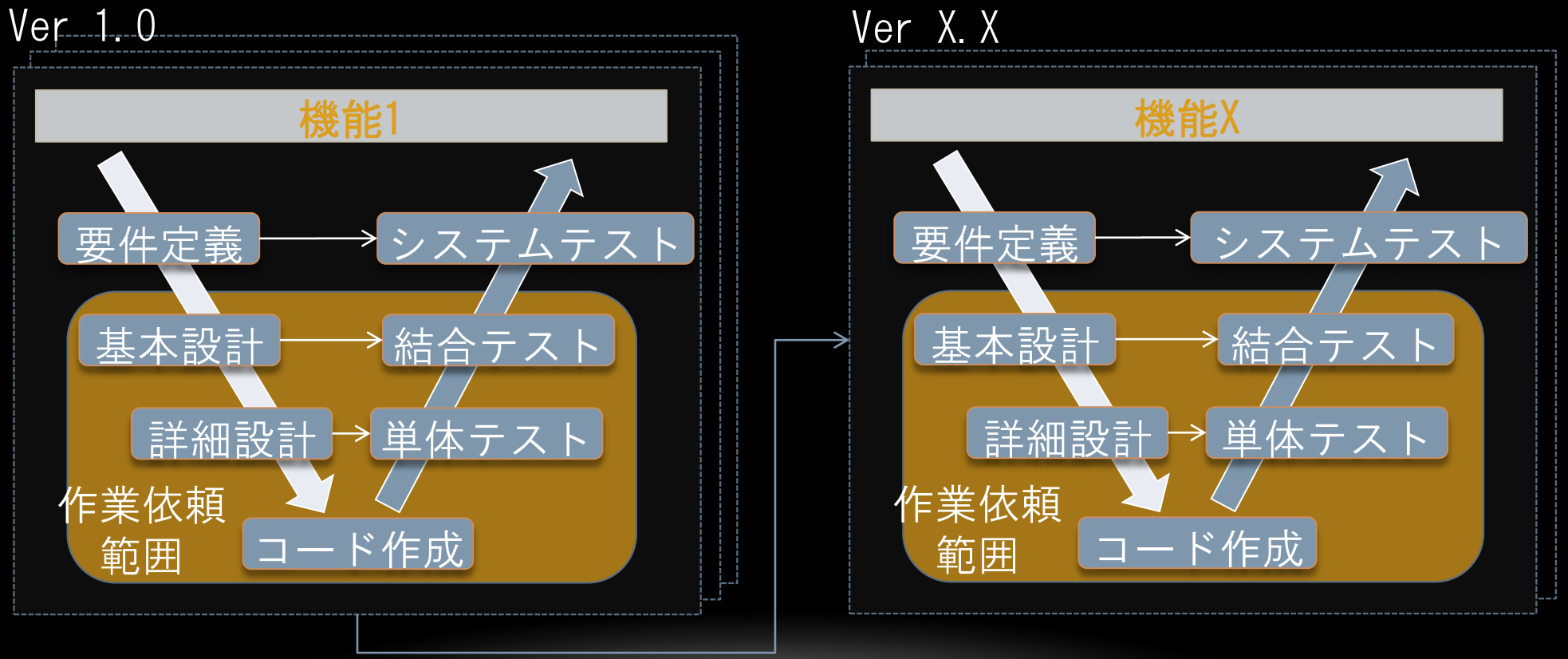
プロジェクト体制

- 総勢11名
- メンバは要求管理者、開発者、テスタに分ける

所属会社	役割	人数	作業内容
OGIS	OG担当者	1	要求定義、外部仕様のレビュー、Q&Aの回答、受け入れを行う
SOT	PM	1	主にOGISとのコミュニケーションを行い、進捗と課題を報告する
	PL	2	プロジェクトの実行を行う
	要求管理者	1	要求を外部仕様にまとめる
	開発者	5	タスクの完成に努める
	テスタ＆QA	2	システムテストとQA

初期(2005年～)の開発プロセス

- 反復開発(機能単位)/V字型開発
- 開発手順書を作成して作業を依頼する



3. 見えてきた開発の問題点と課題

問題点

- 品質の問題⇒コストメリットの減少
  - 受入検査の結果がNGとなる項目が多い
    - 対応漏れがある
    - 非機能要求、異常系のテストが出来ていない
    - 改修前のものが納品されてくる場合がある
  - NGとなった項目の修正を拒否される
- 開発者の不満
  - 作業フロー、ルールが細かい
    - 基本設計書修正のルール、テストのルール、承認フロー
    - 納品のルール、出荷判定が厳密すぎる
  - 要求が細かい

なぜか？

- 要求仕様があいまい
  - 日本流の要求仕様では、あいまいな部分が多い
  - 自分たちだけで解決しようとする
  - 口頭で伝えたつもりが伝わっていない
- テストが不十分
  - 単体テストレベル
- 受入検査NG時が不十分
  - 瑕疵対応という考え方がない
  - 非機能要求を軽視される

根底にあるもの（言葉、文化、地域、組織の違い）

- 向上心が高い
  - 仕事を自分の成長の糧にしようという向上心は高い
    - 技術的好奇心（＝成長の糧）が強いが、非機能要求に対する意識は低い
- 合理的である
  - 仕事を効率よく進めようとする
    - しかし一歩間違うと「手抜き」に
- プライドが高い
  - 質問は恥
    - 自分だけで解決しようとする
- 言葉の壁
  - 会話に難有り
    - 読む＞書く＞聞く＞話す

課題の整理

- 仕様書
  - 文書化されない常識の扱い
- 品質
  - 単体テストの精度向上
  - 非機能要求の実現
  - 仕様変更に伴った不具合
- 情報共有
  - 開発の全体状況、進捗状況の可視化
  - 問題・課題の対応状況の把握
- コストメリット
  - 受け入れの負荷の軽減
- 開発者のモチベーション
  - 細かすぎる仕様書、ルール

4.オフショア　＋　アジャイル開発

適用したアジャイル開発のプラクティス

オフショア開発の課題

仕様書
- ✓ 文書化されない常識の扱い

品質
- ✓ 単体テストの精度向上
- ✓ 非機能要求の実現
- ✓ 仕様変更に伴った不具合

情報共有
- ✓ 開発の全体状況、進捗状況の可視化
- ✓ 問題・課題の対応状況の把握

コストメリット
- ✓ 受け入れの負荷の軽減

開発者のモチベーション
- ✓ 細かすぎる仕様書、ルール

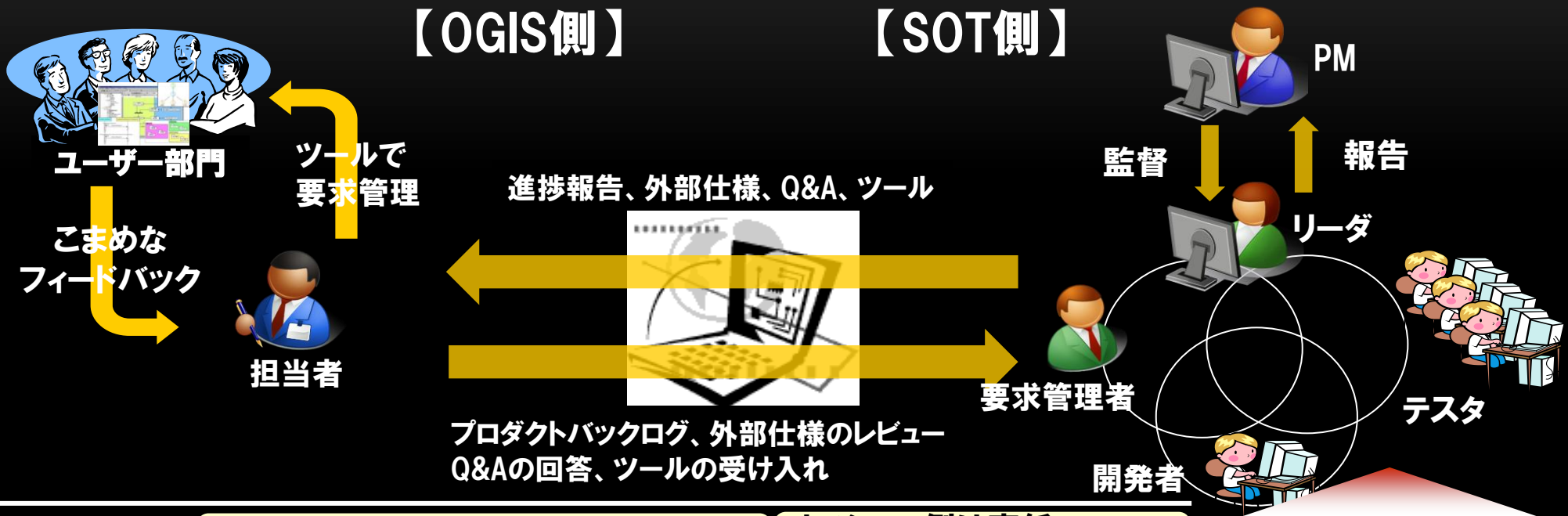
＜解決方針・理念＞

①人と人のインタラクション
- **要求管理**
- **情報共有とコミュニケーション**
- **プロジェクトの可視化**
- **チーム全体アプローチ**

②オフショア側の開発方式
- **反復開発**

③持続的な改善の促進
- **KPTによる振り返り**
- **自主性（自律性）**

①人と人のインタラクション
～発注側と開発側の緊密なやり取りを実現～



要求管理	Trac によって要求をチケットの形で管理	オフショア側は専任の要求管理者を設置
情報共有	TracWikiによる情報共有、経験蓄積	
コミュニケーション	複数のコミュニケーション手段を用意（TV会議、スカイプ、SOBA 等々）	
プロジェクトの可視化	不具合状況をmantisを利用し、管理する Tracによって動作確認結果を管理する	

チーム全体アプローチ

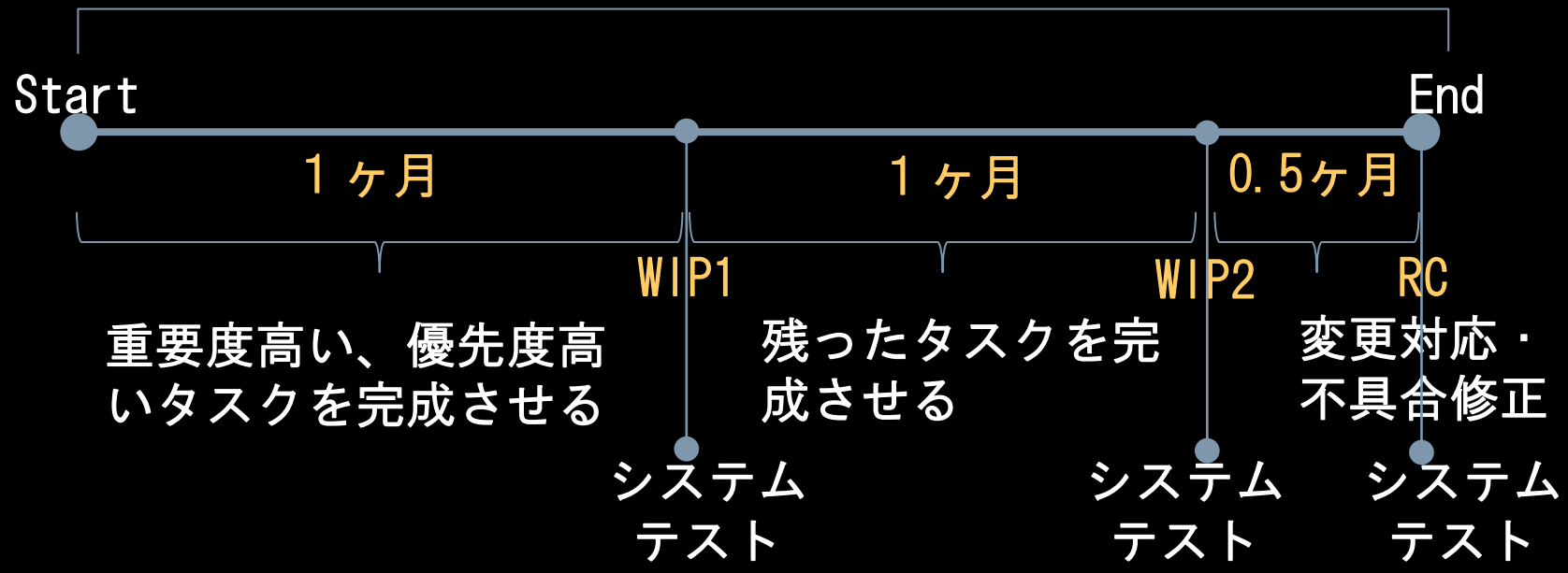
- 相互の役割を皆で担いあう
- 密に連携をとって作業を進める

コミュニケーションにおける工夫点

向上心が高い	・なぜ非機能要求を重要視するのかを理解してもらう ・要求事項の背景を説明する
合理的である	・指摘事項は論理的であること ・細かいところも指摘 ・プラスαの意識付け
プライドが高い	・開発している物を自分たちの「作品」と認識してもらう ・開発上流から参画してもらう ・仕様変更はお互いが納得するように
言葉の壁	・口頭のあと文書でも通知 ・質問の内容から理解度を判断 ・開発者による要求仕様説明会の実施 ・言葉の意味の違いに注意

②オフショア側の開発方式
～反復開発によるこまめなリリース～

- 反復開発により、変更に対して柔軟に対応する



※）WIP(Work In Progress)：中間リリース。動作確認を行う
RC(Release Candidate) ：最終リリース。受け入れを行う

③持続的な改善を促進

～目標＆振り返り～

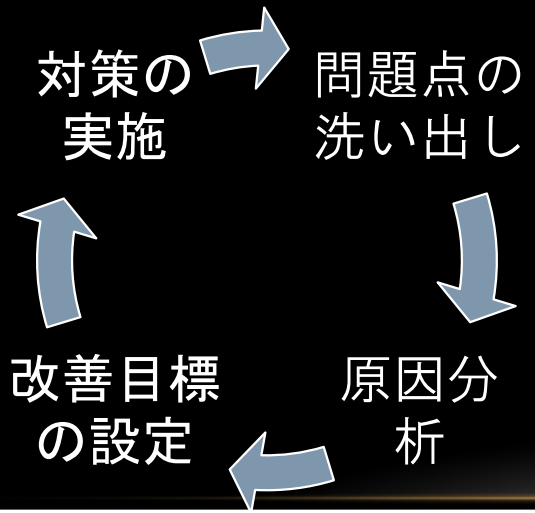
- 目標を設定し、それに向かって最大限の推進力を発揮する

＜恒久的目標＞

【機能目標】
- Criticalな開発目標については、全て対応を完了すること。
- Majorな開発項目については、80%以上の対応を完了すること。

【品質目標】
- Major以上の不具合がないこと。
- 残存不具合件数が100件以下。
- 旧バージョンの優先修正不具合が改修されている。
- 旧バージョンに比べ、レスポンスの低下がないこと。

- “KPTによる振り返り→改善”のPDCAサイクルをまわす



5.結果

結果

プラクティス		コスト	品質	情報共有	仕様書	モチベーション
インタラクション	要求管理	○	○	○	○	
	情報共有とコミュニケーション	△	○	○		
	プロジェクトの可視化	△	○	△		
	チーム全体アプローチ	△	○	○		○
開発方式	反復開発	○	○	△	○	
持続的な改善	KPTによる振り返り		○	○		○
	自主性（自律性）		○		○	○

品質効果

改善前不具合数558件　　**改善後不具合数は53件**

アジャイル開発導入後の変化

- 品質の変化
  - 初期の要求仕様が多少不明瞭でも、開発への影響はほとんど出ない
  - 急に新たな機能追加や要求変更があっても、対応できる
  - 受入検査NG時の対応が迅速になった

アジャイル開発導入後の変化

- 製品の変化(ユーザの声)
  - 動作が軽くなった
  - 直観的で使いやすい

開発者は自分の作品と認識し、プライドを持って、リファクタリングを行った

アジャイル開発導入後の変化

- 開発チームの変化
  - 開発するものは自分たちの「作品」と認識し始め、製品への愛着があった
  - 製品を良くなるために提案するようになった
  - 自律性の高いチームになった
  - 他のチームと比べ、離職率は一番低い

課題

- 情報共有とコミュニケーション
  - 定例会（毎週）用の資料を毎回作成している
- プロジェクトの可視化
  - 状況の確認に時間がかかる
- 反復開発
  - 納品時のテスト、受け入れテストが増えた
  - 指摘事項、改善要望の件数が増えた
  - 中間ビルドで、十分に動作確認できない場合がある

6.今後の予定

今後の予定

Scrumフレームワークにトライ

- 情報共有とコミュニケーション
  - バックログで一元管理
- プロジェクトの可視化
  - バーンダウンチャートを利用
- 反復開発
  - 継続的なインテグレーション
    - 自動テスト、毎日定時刻自動ビルドを実施
  - ペアプロ、テスト駆動開発を導入する

7.まとめ

うまく進めるために

- 工夫と努力は不可欠
  - オフショア側の自主的な活動をサポートする
    - 自主的な活動をサポートするルールづくり
      - 最低限のルールづくり
    - 開発するものは自分たちの「作品」と認識してもらう
    - 上流から参加してもらう
  - 対等の立場を意識する
  - ちょこちょこ顔を合わせ、信頼関係を築く

ご清聴ありがとうございました。