インド アジャイル西遊記

- V社 研修報告 -

株式会社オージス総研
ビジネスイノベーションセンター
山海 一剛

Items

- はじめに
- インドへ（研修概要と参加目的）
- アジャイル
  - なぜ今、アジャイル開発が注目されるのか
  - アジャイルへの期待
  - アジャイルを評価するときは
  - アジャイルとEA
  - XP vs SCRUM
- グローバル・デリバリ・モデル
  - グローバル・デリバリ・モデルとは
  - V社の考えるオフショア
  - GDM（オフショア）成功のポイント
  - オフショア×アジャイルの可能性
- V社における実践例
  - V社におけるアジャイルの歴史
  - 大規模×アジャイル×オフショア事例
  - XPとSCRUM適用事例比較
- まとめ
- 付録

はじめに

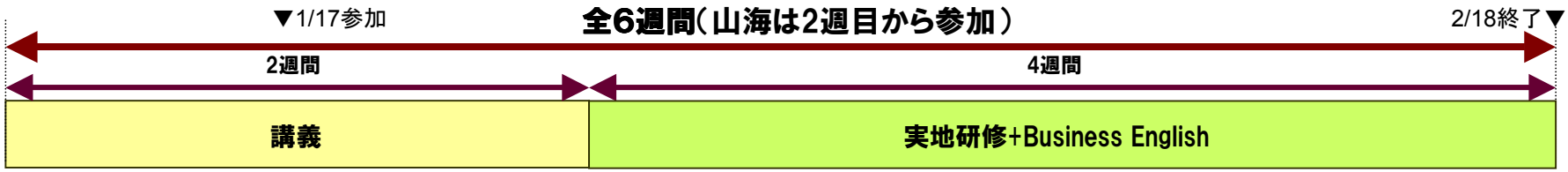
- 以前から付き合させて頂いているインドのIT企業(ここではV社と呼ぶ)が、日本向けにグローバル人材育成を目的とした研修コースを持っておられると耳にした。
- このコースは座学だけではなく、その会社で請け負っている実案件に直接参画するというOJT方式を取り入れているのが特徴。さらには欧米からの案件の大半はアジャイル開発を行っているという。
- 「オフショアでアジャイル？」、以前からアジャイルを勉強しつつも実践の機会が無かった私は、是非この研修に参画してみたいと考えた。
- 実際に行ってみると、彼らは実に見事に「オフショアでアジャイル」を実践していた。またエンジニア達の考え方にも一目置かざるを得ず、非常に刺激的で有意義な時間を過ごすことができた。
- 日本に帰ってみると、多くの人から興味を持っていただいたようで、「どうでしたか？」と、何人もの方に声をかけていただき非常に驚いた。
- その昔インドから仏経典を携えて長安に戻った玄奘三蔵法師もかくの如くか(笑)と思い、研修報告を「インドアジャイル西遊記」と名づけた。
- この資料では、インドで見聞きしたことに加え、それらの刺激をもとに自分なりのアジャイルに対する解釈をまとめたものである。

インドへ

- ✓ 研修概要と参加目的

日程とカリキュラム　カリキュラム（内容・期間）は都度アレンジされる

5



■講義

●Business English（ビジネス英語）　※毎日1時間程度（最初4日間集中講義）

・英語（会話）によるコミュニケーションを中心としたビジネス場面への応用力を強化する。（プレゼン、ディスカッション、ネゴシエーション）

●Cross Culture Communications（異文化コミュニケーション）　1日

・異なる異文化を理解し、国境／国籍に捉われることなく、国際的な領域で活躍するために必要な思考・発想・調整力を習得

●Global Delivery Model（GDM理解）　2日

・CDM理解（インドで成功したビジネスモデルについて学び、アウトソーシングの　ビジネス性に関する初期検討の実施）

●Project Management（プロジェクトマネジメント）　3日

・V社でのProject Management Processについて、V社 Projectsケーススタディ

■演習

●Hands on Assignments（実地研修）　4週間

・実践を通じてインド流のプロジェクトマネージメントタスクを理解
・インドオフショアモデルの実地研修

オフショア＋アジャイルの可能性

- アジャイルと組み合わせることにより、**オフショアの本質的な問題**を大幅に**軽減**できるはず！
  - ドキュメントではなく動くSWをベースにしたコミュニケーション
    - 異文化コミュニケーションの溝（ex行間を読めない…）の解消
    - オフショア先の業務理解促進
    - オフショア元は、仕様の表現の仕方、指示の仕方を学ぶ
  - 時差を利用することで納期短縮効果も期待できる
- つまり・・
  - エンドユーザの近くで開発する方がベターであることは言うまでも無いが・・オフショアでコスト削減を狙う前提なら、ウォーターフォールよりもアジャイルが圧倒的に親和性が良いはず
  - オフショアによって**コストダウン**しつつ、アジャイルによって**ユーザ満足度**の高いシステムを構築することが出来れば、生き残りの鍵になるのではないか？

V社J2P研修への期待

- そのためには
  - オフショア＋アジャイルならではの管理技法があるはず
  - 実案件を経験することでノウハウを吸収する以外に方法はない（体験あるのみ）。
- カリキュラムのポイントはハンズオン
  - V社の欧米向けオフショア案件の**半分以上はアジャイル**手法で行われている。
  - 研修にてV社のPMに補佐的につくことで、そのマネジメントスキルを吸収することができる。

アジャイル

- なぜ今、アジャイル開発が注目されるのか
- アジャイルへの期待
- アジャイルを評価するときは
- XP vs SCRUM

なぜ今、アジャイル開発が注目されるのか(1)

- 経済環境の悪化によるシステム投資の考え方の変化
  - プロジェクトの成否からプロダクトの効果へ
    - システム構築の成否はQCDではなく、経営戦略やユーザ業務にどの程度貢献するか
    - 「予算通り＋納期通り＋バグが少ない」であっても、業務に貢献しなければ「無駄な投資」
  - ウォーターフォールのオーバーヘッドを許容できなくなった
    - 要件変更を最小限に抑えることを前提としたマネジメントは、あまりにもオーバーヘッドが大きい(過大なドキュメンテーションと膨大なレビュー工数)
- 気づき:要件定義の困難さ
  - そもそもすべての要件を事前に定義すること自体が非常に困難であり、ユーザに求めるべきことではないことに気が付いた(要求開発もそのひとつの手段)。
- 戦略的で不確定要素の多い業務が増えてきた
  - 試行錯誤を許容しながらも、迅速に必要最低限の機能から作りこむ手法が必要になった。

- 変更を前提とした手法
  - 積極的に変更を受け入れる手法であるアジャイルが注目をあびるようになった。

なぜ今、アジャイル開発が注目されるのか(2)

- 開発技術の進化
  1. アプリケーションのWeb化
  2. 自動ビルド環境
  3. 自動テスト環境
  4. オープンなフレームワークの浸透
  - エンドユーザとの相互作用が容易に
    - 変更・追加された機能を、ほとんど工数をかけずにユーザに確認してもらえる環境
    - メインフレームやクラサバであれば、新モジュールのデプロイだけでも大きな工数がかかってしまう。
  - 反復的な開発環境を可能とする環境整備
    - すでにリリース済みのモジュールの改修を安全かつ迅速に行う環境
    - DIコンテナやGUIなどの既成フレームワーク活用によって、より業務ロジックに集中できる開発スタイル
- 3K、5K、7Kに代表される職場環境への反動

アジャイルへの期待

- 安い
- 早い
- 品質が良い
- 安心

Yes…では、何と比べて安いのか

Yes…いつまでに何が必要なのか

Yes…でも、品質とは何か

Yes…こまめに進捗と品質できる

アジャイルを評価するときは(1)

- ウォーターフォールと比較する場合は
  - 同じスコープを前提に比較しない。
    - 作りすぎを防ぐのがアジャイルの効果であり、同じスコープ(機能数)で比較することは無意味。
  - 生産性を指標にしない
    - リリース済みの機能でも積極的に改修していくのがアジャイル。生産性(ライン数/工数)で比較することは無意味
  - 開発期間だけでなくメンテナンス期間も意識する
    - スケジュールと予算どおりにリリースできても、ユーザ業務と合致しない部分が多く。リリース後の量が多いのがウォーターフォール。
    - アジャイルは反復しながらユーザ要件にアダプトしていく手法であり、リリース後の改修工数も意識して比較すべき。
  - 二元的な評価をしない
    - 単純に「WF vs Agile」という二元的な図式を描くと本質を見誤る。
    - アジャイルにはいろいろな流派があり、ユーザとの関わり、見積もりや計画に対する考え方は大きく異なる(後述)。

アジャイルを評価するときは(2)

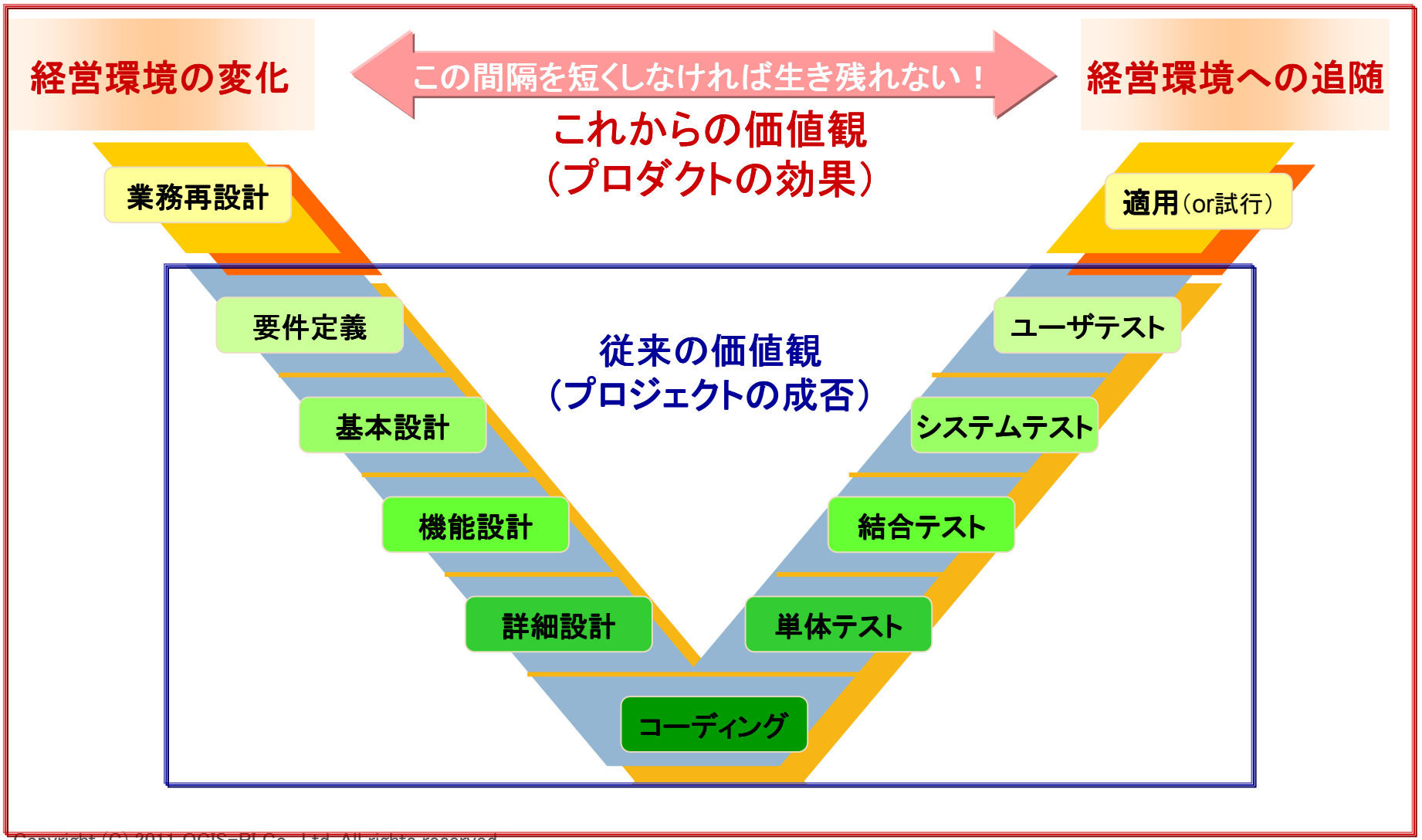
- アジャイルは
  - ✓ リスクを早期に回避する
  - ✓ エンドユーザにとってのバリュー(ビジネスへの貢献度)を最大化する

  ための手法。

- だから…
  - 単にQCDだけでウォーターフォールと比較すれば、アジャイルは同じかむしろ悪くなる。
  - 言い換えれば、現状のQCD(この場合のQ指標は、狭い意味での不具合率)は、アジャイルの評価には役立たない。
  - しかしそれは、「アジャイルは評価出来ない」という意味ではない、
  - 「プロジェクトの成否ではなく、プロダクトの効果へ」というムーブメントに対して、我々が有効な指標をもっていないということである。

アジャイル開発における価値

プロジェクトの成否からプロダクトの効果へ



アジャイル開発とEA(1)

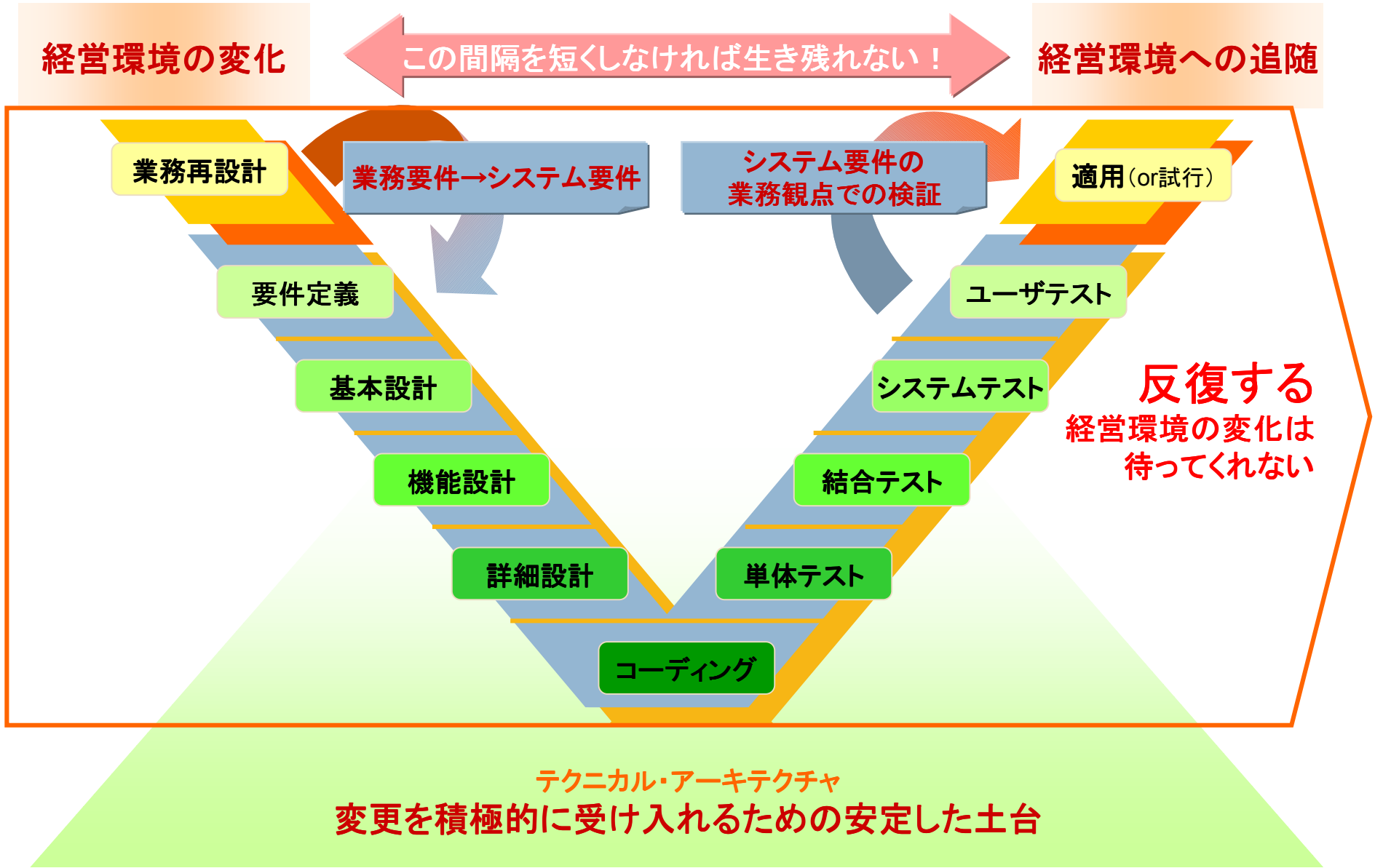
- 「変更を積極的に受け入れる手法」であるが、手法だけでは実現できない
- 企業全体の観点からの指針が必要
  - 反復しながらアプリケーションを成長させていく過程で、企業の中でのアプリの位置づけを見失わないための指針
    - BA (Business Architecture)
    - DA (Data Architecture)
    - AA (Application Architecture)
  - 他アプリケーションとの役割分担、データや機能配置を意識して、機能を追加/変更していく必要がある

アジャイル開発とEA(2)

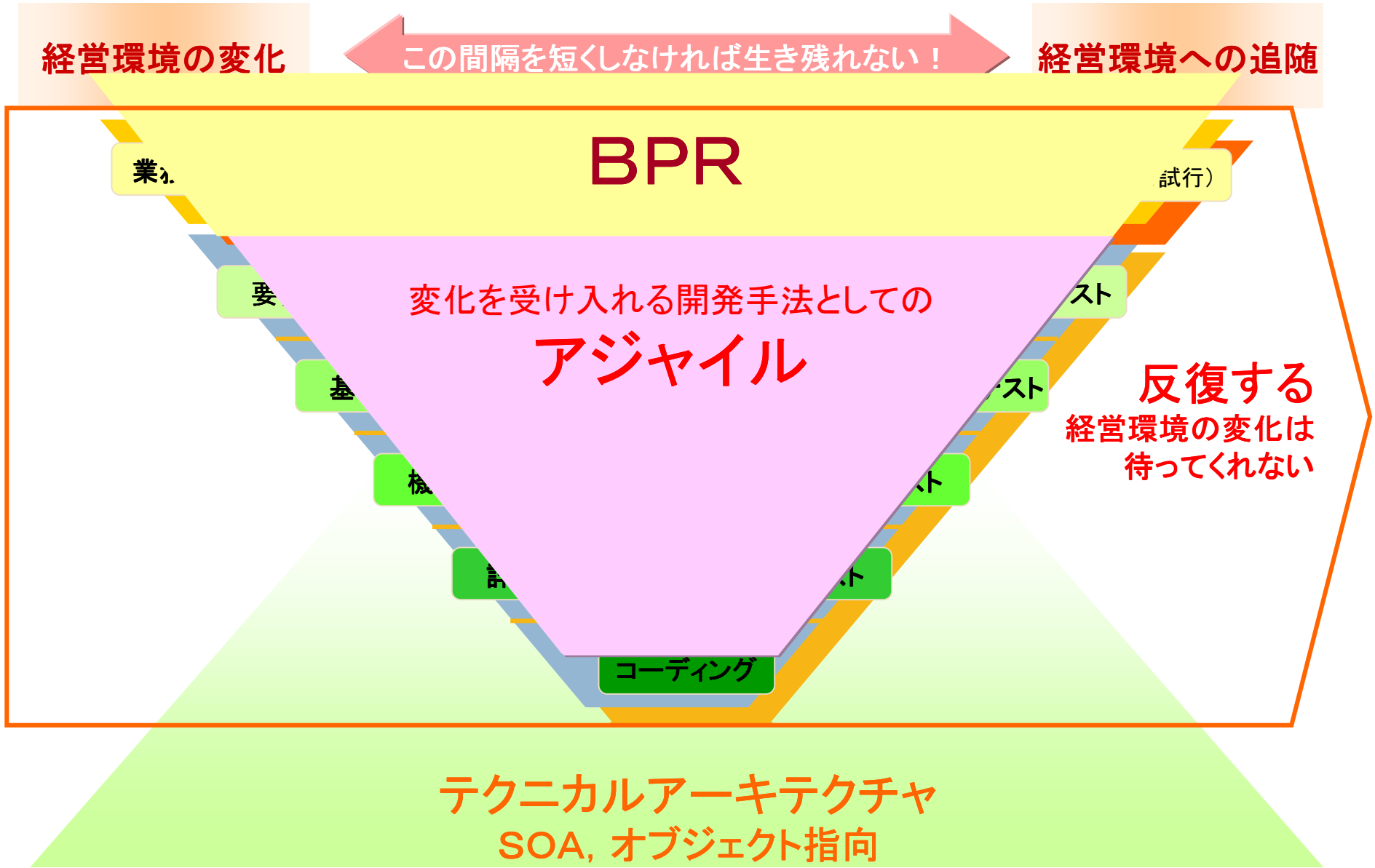
- 体系化された技術基盤上で開発する必要がある
  - アプリ毎に異なる作法や基盤で作らない(アジャイルでなくてもそうだが…)
    - **TA** (Technology Architecture)が必要
  - 安全かつ迅速に反復できる環境が必要
    - 変更箇所を局所化する技術　　SOA/オブジェクト指向
    - デグレードを自動的に検出する技術　　自動リグレッションテスト
    - 動くソフトを素早くユーザに提供する技術　　自動ビルドツール

      →これらもTAとして定義されているのが理想

アジャイル開発とアーキテクチャ(3)



アジャイル開発とアーキテクチャ(3)



XP vs SCRUM

- 大きな二つの潮流
  - XP
    - 多くのプラクティスを提案
    - ユーザとの継続的なインタラクションを重視
      - ユーザと開発者は一体のチーム
      - 見積もりと計画よりも、目前の要求に迅速に対応することを重視
  - SCRUM
    - プラクティスはなく、プロジェクト運営手法が中心
    - メリハリをつける。
      - スプリント中は開発チームに全権を委ねる
      - スプリント間は積極的にユーザとかかわり、要求開発を行う。
      - 見積もりと計画を重視
    - 反復開発することで実測をベースにしたプランニングにより、計画性のある反復開発を可能とする。
  - V社ではプロジェクトに応じて、使い分けている(後述)。

グローバル・デリバリ・モデル

- ✓ グルーバルデリバリモデルとは
- ✓ V社の考えるGDM
- ✓ GDMのパターン
- ✓ GDM(オフショア)成功のポイント
- ✓ オフショア×アジャイルの可能性

グローバル・デリバリー・モデルとは

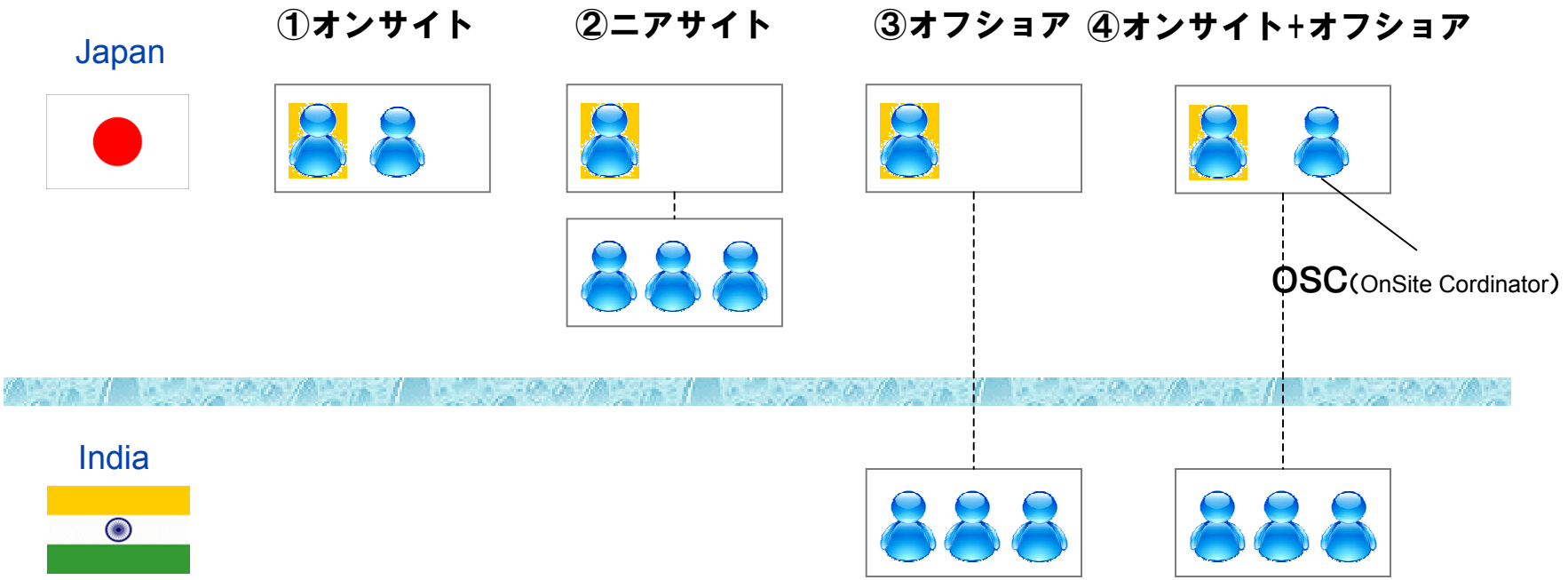
- グローバル・デリバリー・モデル（Global Delivery Model）
  - 顧客拠点であるオンサイト、同地域のオンショアと、低コスト地域であるニアショア、オフショアを組み合わせて、ITサービスを提供するビジネスモデル
  - グローバル・ソーシング（Global Sourcing）とも呼ばれる。
- メリット
  - コストメリット
  - 人材調達の容易さ
    - 調達規模と調達スピード
    - 業務知識を持つSE
  - 時差の利用
    - ヘルプデスクサービスなど
    - フォロー・ザ・サン開発
  - 顧客のグローバル化に対応

➢ 体制づくりの柔軟性を獲得することで、案件受注力、案件遂行力を向上

V社の考えるGDM

- オフショアは手段のひとつ
  - メリットとデメリットを理解して使い分ける、組み合わせる
    - ┌オンサイト
    - └オフサイト┬ニアサイト(ニアショア)
    - └オフショア
  - ODC(Off-share Development Center)
    - 特定顧客、特定プロジェクトのために、一定期間オフショア先が人材を確保する
- 単独のプロジェクトでは考えない
  - 自社ビジネスのどの部分をアウトソースするか
  - アウトソースする場合、どこにオフショアにするか

GDMのパターン比較



	①オンサイト	②ニアサイト	③オフショア	④オンサイト+オフショア
コミュニケーション	○	△	×	△
コスト	×	△	○	△
人的リソース	×	×	○	○
カントリーリスク	○	○	×	×

GDM(オフショア)成功のポイント

- **準備**
  - いきなりGDMの実践は無理。周到な準備が必要
  - 準備は複数の部門に関連する
    - 組織構造、責任分担、調達プロセスの見直し、契約形態
    - 監視と評価の枠組、リスク評価の基準、チーム編成の考え方
- **業務理解**
  - 開発チームに、業務に関する情報を十分に与えること
  - 出来ていない場合は、失敗するケースが多い
- **OSC(OnSite Cordinator)が重要**
  - ユーザの立場で要件を理解、開発チームに伝える
  - 開発チームの状況をユーザに伝える
  - OSC:開発者=1:5が理想(ウォーターフォールの場合は1:10まで許容)
- **ツール類の環境整備**
  - TV会議の活用
  - デスクトップ共有
  - プロジェクト管理用ソフトウェア(Webベース)

オフショアのリスク

- リスク
  - コミュニケーション・カルチャリスク　→次項
  - 社会的/政治的リスク
    - 戦争、テロ、政情不安
    - 不安定な社会インフラ
    - 経済活動への規制
  - セキュリティリスク
    - セキュリティに関する法制度が未整備
    - 知的財産保護に関する法制度が未整備
    - 争議解決の枠組み（制度）が未整備
    - 契約破棄
  - ファイナンシャルリスク
    - 為替レートの変動
  - オフショア先に依存することのリスク

- 低減策
  - 関連する法整備状況を事前に確認
  - 契約の枠組み整備
  - 問題発生時のエスカレーション、解決、調停の枠組み整備
  - オーディットの取得
  - 再委託の禁止
  - 物理的な隔離（ODC）
  - セキュリティ認証制度の確認
  - 低減策
    - 対象地域の選定
    - 経済的に安定している地域を選択
    - 社会インフラの事前調査

オフショア×アジャイルの可能性

- オフショアのリスクを低減する手段としてのアジャイル
  - 動くソフトウェアをベースにしたコミュニケーション
  - 反復を通しての学習効果によるコミュニケーションの向上
    - オフショア先：ユーザの求めるもの、言葉ですら表現しにくい部分（行間）を、学習できる。
    - エンドユーザ：オフショア先の理解度を確認できる。どのように伝えればよいかを学習できる。
- 当初のアジャイルは、エンドユーザのそばでの開発に価値をおいていたが、最近は違ってきている
  - コミュニケーション・ツールの発達
  - アジャイル・プロジェクトにおけるユーザ役割が明確になってきた
  - OSCがユーザを代行するスタイルの確立
- V社は多くの案件で既に実践、学ぶところが多い。

V社における事例

- ✓ V社におけるアジャイルの歴史
- ✓ 開発手法に対する考え方
- ✓ アジャイル＋オフショア成功のポイント
- ✓ アジャイル＆オフショアを支えるツール
- ✓ 大規模＋アジャイル＋オフショア事例
- ✓ XPとSCRUM適用事例比較

V社におけるアジャイルの歴史

- 2004年(頃)M社案件スタート
  - 大手ポータルサイトの広告事業管理
  - M社がアジャイルの適用を要求した。
  - M社メンバ10人に対しV社4人のオンサイト体制でスタート
- アジャイルの有効性を認識した初期メンバが社内で普及活動
  - コンセプトの啓蒙・布教
  - 支援ツールの発掘、評価、紹介
- 並行してM社案件は、オフショア＋大規模化
  - 複数のアジャイルチームによる最大60名体制

➢ 当初はアジャイルを疑問視する声も多かったが、実績がものを言った。

V社のアジャイル手法に対する考え方(1)

- プロジェクトに応じてXPとSCRUMを使い分けている

	XP	SCRUM
プロジェクト管理	特に言及していない	体制、役割、日々の作業 要求からタスクへのブレークダウン
反復期間	1W～2W	3W～4W
実践	ペアプログラミング、TDD(Test Driven Development),CI (Continuaus Integration)など	特に言及していない
コスト	△ ペア・プログラミングによる工数増や、自動ビルド＆テスト環境構築などのためにコスト増	○
品質	◎ ペアプログラミングにより、単なる欠陥率の低下だけではなく、より良い設計が得られる。	○

- 高品質が求められるアプリケーションや、複雑なアプリケーション構築(要件確定に試行錯誤的要素が必要)には、XPが適している。

V社のアジャイル手法に対する考え方(2)

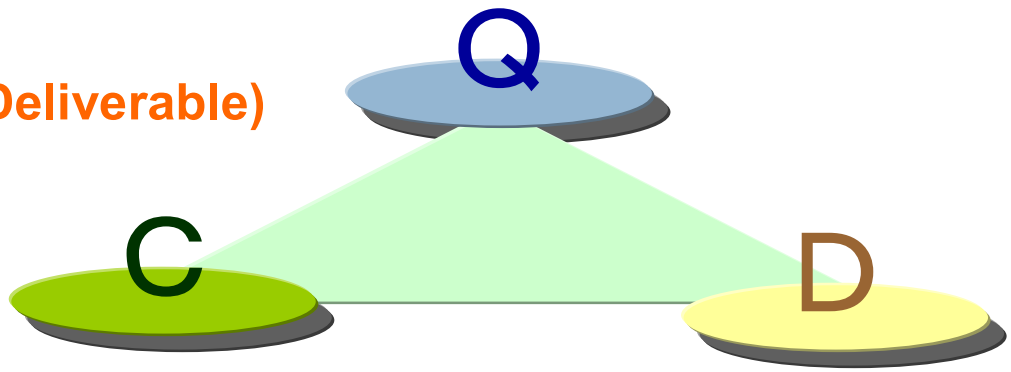
- アジャイルは品質が良い
  - アジャイルとは「**品質を最優先にする手法**」であるとの考え方が、社内のほぼ全てのエンジニアで共通認識となっている。
    - 「品質を妥協しないために、スコープを可変にする」という考え方であるとも言える。
    - 日本でアジャイルというと、「品質面では多少目をつぶって、安く早くつくる手法」というイメージを持つ人が多いのでは？

V社のアジャイル手法に対する考え方(3)

Q：品質 (Quality)

C：費用 (Cost)

D：成果物/スコープ (Deliverable)



	価値観	QCDの全てを守れない場合は
従来手法	全てが優先(Dも優先)	D(成果物、スコープ)を優先する →Q(品質)が悪化する →C(コスト)が超過する
反復開発	QとCが優先	D(成果物、スコープ)を変更する (つまり優先度の低い機能は実現しない)

V社の考えるアジャイル＋オフショア成功のポイント

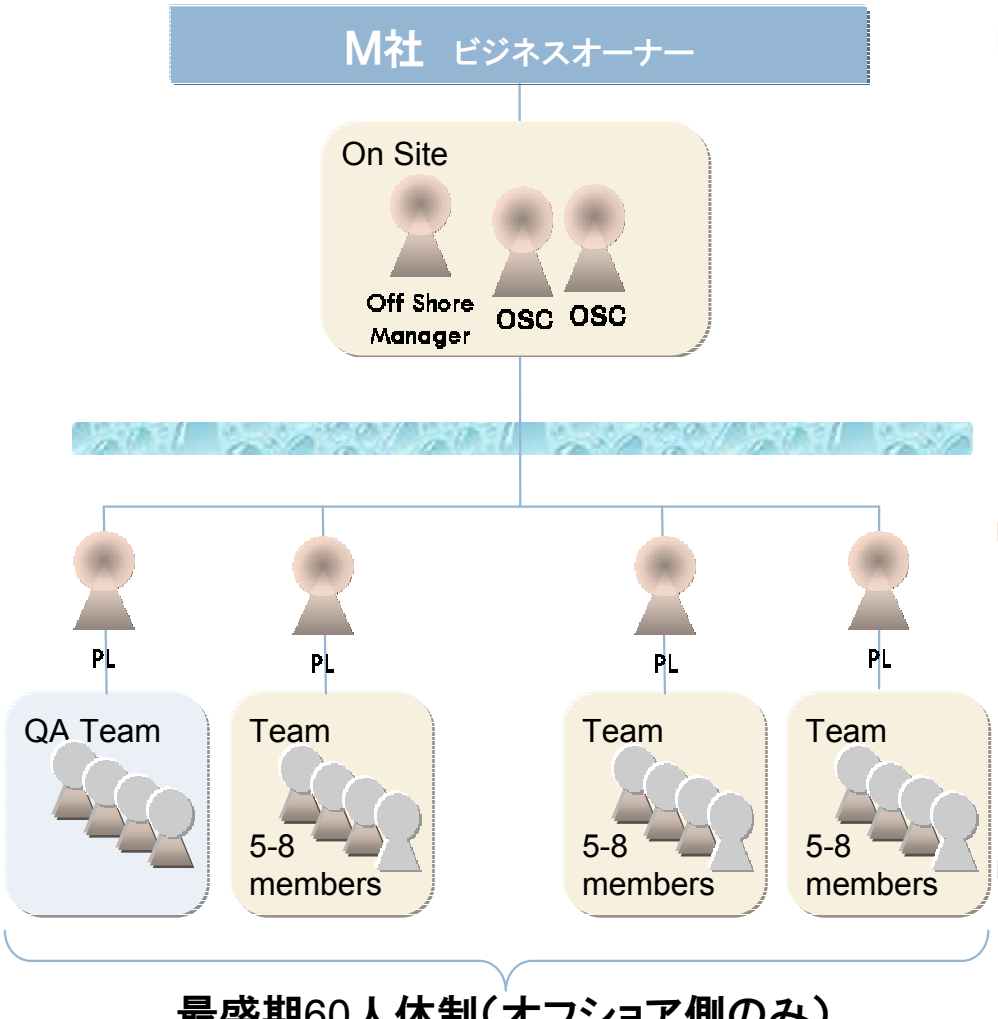
- オフショアチームの業務理解
- エンドユーザの積極的な関与
  - V社でもうまくいかなかったことがある（エンドユーザの関与率が低い→ユーザがフェードアウトしてしまった場合など）
  - リリース計画立案などエンドユーザには無理なタスクもある
  - OSCがエンドユーザを代理するプロジェクトもある。
- 重要なポストに優秀な人材を重点配置
  - 優秀なアーキテクト
    - ユーザ要求を相互に依存性の低い要件にブレークダウンできる能力
  - 優秀なOSCの確保
- ツール類の整備
  - 次項

➢ 前者3つはウォーターフォールであっても必要（やり直しがきかない分、むしろもっと重要）

アジャイル＆オフショアを支えるツール

- アジャイル・プロジェクト管理
  - versionOne
  - Rally
- 自動ビルド
  - CruseControl
- テストオートメーション
  - MS Test
  - RhinoMock
  - Quick Test Pro
  - Silk Test
  - Test Partner

大規模＋アジャイル＋オフショア事例



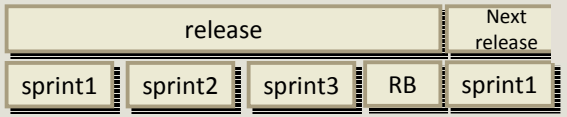
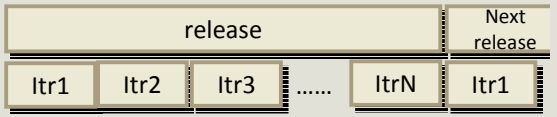
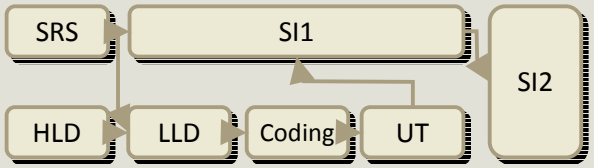
最盛期60人体制（オフショア側のみ）

- M広告事業管理
  - OSC一人あたり、1～2チームを担当
  - 1チームは5～8名（8名程度が限界）
  - 各チームに1名のテクニカルリーダー
  - チーム単位で毎朝スタンドアップミーティング
  - その後リーダ間でスタンドアップミーティング
- 機能分割がポイント
  - × 開発効率で分割
  - ✓ メンテナンス性で分割
    - プラガブルな状態（依存性が低い）
    - 何かあれば、機能単位で反映をオフ/オン
- 顧客事情で体制が増減
  - プラクティスによりナレッジトランスファが容易
    （ペア・プログラミング、ソースの共同所有）

XPとSCRUM適用事例比較(1)

- 現地では、主に2つのプロジェクトに参画した。
  - CCVプロジェクト
    - Amol氏がリーダを努めるSCRUMを適用したプロジェクト
    - 3～4週間単位の反復
  - Feedback Systemプロジェクト
    - もうひとつはAnjali氏がリーダを努めるXPを適用したプロジェクト
    - 1週間単位の反復
- 疑問
  - なぜこの2つのプロジェクトが、異なる手法を採用しているのか？が疑問であったため、2プロジェクトの相違点を表にまとめ、2人のリーダを呼び出して、説明してもらった。
  - その資料を次ページ以降に示す。
  - ➢ 結果として2人のリーダの考え方に大きな違いはなく、プロジェクトの特性に応じて、反復期間や作業内容を工夫しているとのことであった。

XPとSCRUM適用事例比較(2)

	CCV	Feedback System
Method	SCRUM	XP
Term of Iteration	3-4W	1W
		
Communication w/ User	Once in 2 or 3 releases	Weekly Demo every Friday
Definition of Activity in sprint/iteration	Defined  *Defined through project and it may be changer depending on backlog *UT and SI are not automated but should be. *some time SRS, HLD, LLD might be shortened or omitted. Another activity might be append. Process and activity is also agile.	Not defined *Defined as a backlog when any activity identified.

XPとSCRUM適用事例比較(3)

	CCV	Feedback System
Structure and roles of team	Product Owner SCRUM Master ├Architect ├Dev. Team(AddIn) ├Dev. Team(PWA) └Test Team *Each team has a leader. Leader allocate evry task in team.	Product Owner Team *each member pick up task by ones self.
Breakdown to activities from requirement	Requirement -> product backlog -> choose PB for next sprint(sprint backlog) *Product Backlog List contains story.	Product Backlog = Story -> choose PB for iteration ->break down to activity *story is described in one sentence, complex story may have story description.

XPとSCRUM適用事例比較(4)

	CCV	Feedback System
Documentation	User requirement. feature list(excel) product backlog list(excel) SRS document(excel) Design design document(word) Test SI test case and evidence(excel) *"Just enough document is policy". *product backlog list,SRS documentation and SI test cases are per customers request. *all these is need for tracking.	All stories are as product backlogs in versionOne. *User manual is best document for WHY. *Code is best document for HOW *"make some more documents if we need like 'high level architecture document', not for how but why". *"test as code" *Iteration is short, so we can manage without documents.
Practice of XP	× Pair Programing × Test Driven Dev.(Automation) ✓ Continuous Integration	✓ Pair Programing ✓ Test Driven Dev.(Automation) ✓ Continuous Integration /w tool

XPとSCRUM適用事例比較(5)

	CCV	Feedback System
Do modeling?	No *Use partial class diagram in design document	No
Tools	Excel : Prj management, JIT management Wizible : Issue management CruseControl : Continuous Integration	versionOne : project management RhinoMock, MS Test : Test Automation CruseControl : Continuous Integration

まとめ

- ✓ 実感
- ✓ 日本のSIerの課題
- ✓ アジャイル普及のために

実感(1)

- アジャイル(反復型)開発に関して
  - インドは先進国(他の会社を知っているわけではないが、少なくともV社は進んでいる)
  - 特に管理ツールや自動テストツールを駆使して、一週間単位のアジャイル開発を、地理的・時間的壁を越えて実現している点は驚嘆に値する。
  - 日本は遅れている
- アジャイルと品質
  - 開発者だけではなく、品質管理部門のヘッドも含めて、アジャイル開発の方が品質が良いという認識であることに、驚かされた(品質を優先するならアジャイルという発想)。
  - エンドユーザあるいは発注側の観点からも、進捗と品質が、毎週目に見えるかたちで把握できるため安心感がある。

実感(2)

- アジャイルは"現場の工学"であることを実感
  - V社では実案件を通して、アジャイルのコンセプトも、ツール適用技術もうまく継承・伝播されており、組織としてのナレッジを構築できている。
  - 自分の身の回りを見る限り、一部の先進的なエンジニアによる事例があるものの、少なくとも"組織知"になっていない。
- おまけ
  - アジャイルはXPから入るべき
    - XPのイテレーションが1週間であるのには理由がある
    - XPのプラクティスは相互に補完しあい、1週間のイテレーションを実現する
  - XPはトヨタ生産方式に酷似している
    - 限りなく在庫持たずに生産するために捨てるもの/補うもの
    - 1週間でイテレーションを廻すために捨てるもの/補うもの

国内のSIerの課題

- アジャイル（反復型）開発という単品ではなく、BPRやSOAと組み合わせてこそ、経営環境変化に迅速に対応できる情報システムを提供できる。
  - BPR
    - 単発ではなく継続的な業務改革活動と、それを受けての反復開発
  - SOA
    - 継続的な業務改革を支える柔軟性の高いアーキテクチャ
  - オブジェクト指向、モデルベース開発
    - 柔軟性の高いソフトウェア構造を提供
- まずはアジャイルを組織知とし、その後オフショア経験を積むべし
  - オフショアによって**コストダウン**しつつ、アジャイルによって**ユーザ満足度**の高いシステムを構築する
  - 単価競争力が激化し、「高品質だから高い」では通用しなくなってきた昨今、これは**生き残りへの道**

アジャイル普及のために

- 「現場の工学」であるアジャイルの普及には
  - 日本のSIerの大半は後発であり、それゆえ机上の議論や評価（象牙の塔的なアプローチ）は不要
  - どの手法が良いか？の議論もそれほど重要ではない（V社は経験をとおして熟知し、使い分けている）
  - 現場で経験を積ませることによるエンジニアの育成こそが急務
  - 経験者がいなければ、ユーザ企業に提案すらできない
  - 日本では得がたい"現場の代替"として、この研修は貴重な場

付録

- ✓ インドと中国の違い
- ✓ 品質管理部長インタビュー

インドと中国のエンジニアの違い

- 私にとって、これまでのインドのエンジニアの印象は、個人主義であり"助け合う"というイメージには遠かった。
- 今回スタンドアップ・ミーティングに参加して互いに助言しあう光景をみたり、V社のPMたち間でアジャイルの技術や思想がうまく受け継がれているのをみて、考え方を大きく変えた。
- 以下は、同じ研修生で中国オフショアの窓口を担当しておられた方とのディスカッションから、まとめた比較表である。
- 帰国後寄せられた意見に「オフショア先のエンジニアの違いではなく、発注側の"使い方"の違いではないか」との指摘があった。確かにそうかもしれない。

	インド	中国
エンジニアの価値観	技術的なスキル・知識を持って活用することが、エンジニアの価値	開発する(コーディングを書く)ことがエンジニアの提供する価値
SEの評価	より高度なサービスを提供することで評価される	指示されたものを早く or 大量に作ることで評価される
同僚とは	同僚は仲間	同僚はライバル

V社品質管理部長インタビュー

47

品質管理部長という立場からみたアジャイルについてご意見を伺うべく、インタビューさせていただいた。

Q　アジャイルとウォーターフォルで品質管理の考え方に違いはあるか？

A　タスク管理（工数管理）と（狭義の）品質管理で答えは異なる。工数管理という観点では、アジャイルはウォーターフォールと大きく異なる。例えばアジャイル では"ベロシティ"という考え方でタスクアサインを行う。狭義の品質管理（バグ管理）という観点では、両者の考え方は非常に近いが、アジャイルの方が品質は良い。そもそも沢山の機能を一度に作るウォーターフォールは本質的に無理がある。重要な機能から少しずつ理解を深めて、開発できるアジャイルの方が品質面では有利である。

Q　欧米と日本で、顧客の品質に対する考え方に違いはあるか？

A　それはもう大きく違う。そもそも欧米顧客は、バグ分析などのレポートを提出しても、あまり見ない。それよりも必要な時期に製品がマーケットインできるかの方を重要と考えている（プロダクトアウト・ファースト、クオリティ・ネクスト）。日本は非常に深い分析を求める（クオリティ・オリエンテド）。ドキュメントの内容をとっても、非常にたくさんのグラフや表を求めるし、そのレビューにも非常に時間をかける。下手をすると、バグ1つあたり1ページの分析報告資料を求められることもある。またバグフィクスについても、すぐに対応することを求められ、さらに即刻ドキュメントに反映することを求める。それゆえ日本のプロジェクトのメンバーは非常にストレスを感じる。

Q　日本はアジャイルの普及が5年程度は遅れていると思うのだが？

A　明らかに5年以上は遅れている。10年近い差があるのではないだろうか。欧米のユーザ企業にとってウォーターフォールはもはや時代遅れ（out of date）の開発手法とみなされている。

Q　日本でアジャイルの普及が遅れている理由は何だと思うか？

A　まずはリスクテイクの考え方の違いだろう。欧米のユーザ企業は自らリスクをテイクするが、日本はそうしようとしない。また欧米は開発サイクルが早い。やはりいかに早くマーケットに出すかを最優先に考えればアジャイルという選択肢になるのだろう。一方日本は、プロジェクト開始時にはリスク分析に時間をかけるし、リリースにあたっても、品質を確保して（バグフィックスに時間をかけて）からリリースしようとする。

Q　V社としては、ウォーターフォールとアジャイルが混在し、またアジャイルでもXPとSCRUMの両方を使用している。V社の品質管理担当としては、社の標準として統一すべきと考えているか？

A　統一できれば良いが、実際にはお客様の要望などもあり難しい。もし統一できるとすれば、すべてのプロジェクトをアジャイルで統一したい。
手法がどうであれ、どのようなプロジェクトでも要求変更は必ず発生する。であれば変更を積極的に受け入れる手法であるアジャイルを使うべきだ。

Q　XPとSCRUMについては、どのように評価しているか？

A　XPはイテレーションの間隔が短く、ペアプログラミングを行うことから、より不確定要素の多いプロジェクトや、高い品質を求められるプロジェクト、複雑な構造のソフトウェア等に適している。

Q　XPとSCRUMのどちらか一方を選ぶとしたら？

A　あえて言うならSCRUMである（あえて一つに絞り込む必要を感じて無さそうな雰囲気）。XPには利点が多いが、ペアプログラミングによる工数増や、テストドリブン環境や自動インテグレーション環境などもしっかりと構築する必要があり、コスト増になる傾向がある。

ありがとうございました

- あなたも行きませんか？インド研修！