UMLモデリング推進協議会
アジャイル開発部会
アジャイル開発事例セミナー

エンタープライズ・アジャイル開発手法の導入への取り組み

株式会社 中電シーティーアイ
ITソリューション事業部
保安業務基幹システム開発室
chuden CTI 大橋 正敬

本日の目次

1．会社紹介
2．アジャイル開発手法への取り組みの背景
3．アジャイル開発手法適用へのこれまでの挑戦
4．エンタープライズ・アジャイルへの期待と準備
5．エンタープライズ・アジャイルの挑戦
6．本日のまとめ

chuden CTI

1. 会社の紹介

会社紹介

中部電力グループは，愛知・岐阜（一部を除く）・三重（一部を除く）・長野・静岡（富士川以西）中部5県に，ライフラインである電気を中心に，ガス・LNGやオンサイトエネルギーなど，お客さまの多様化するニーズに対応した「安定」にかつ「安価」なエネルギーサービスを供給しております。

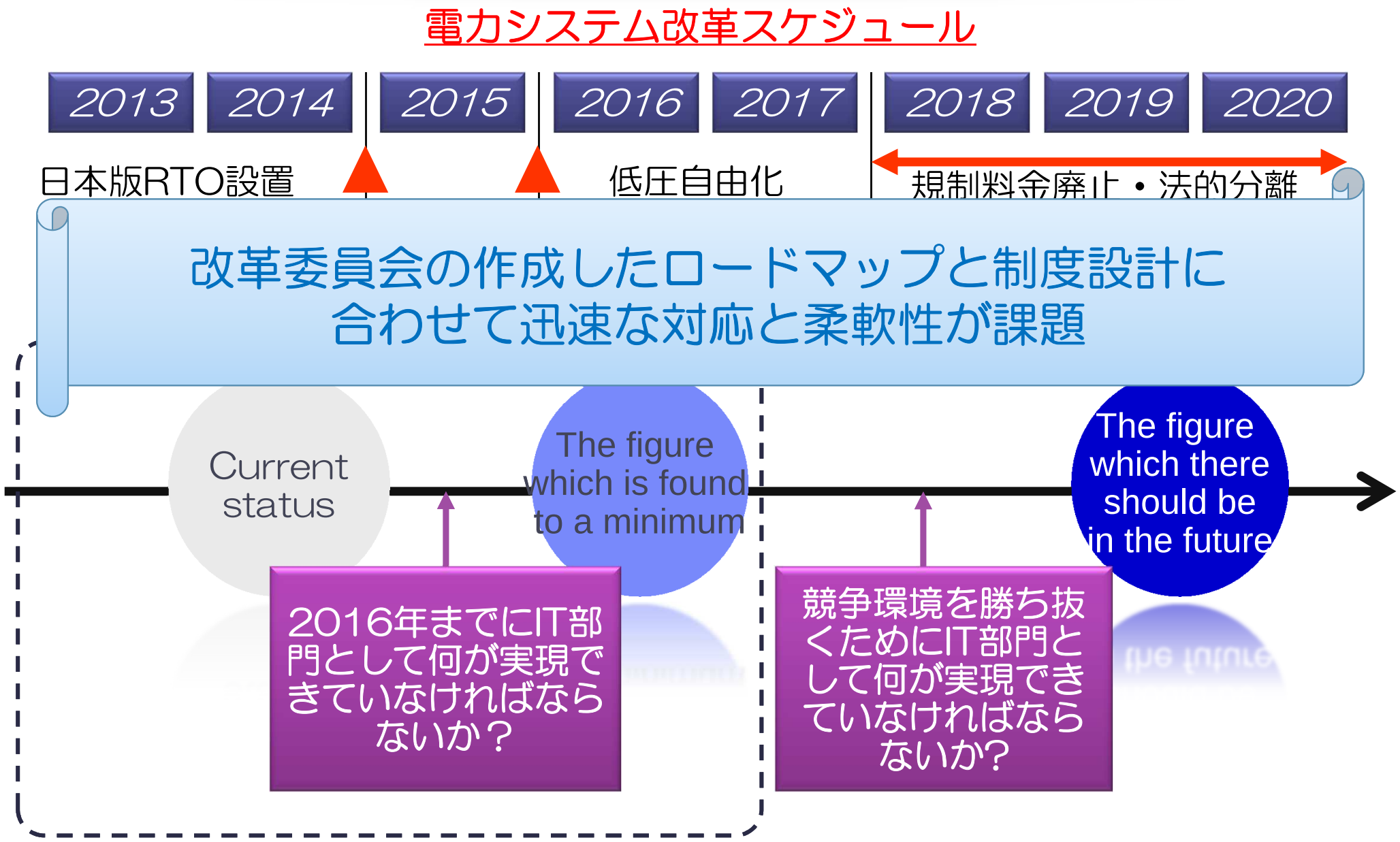
株式会社中電シーティーアイは，中部電力株式会社100%出資による情報系会社です。平成53年に設立された中電コンピュータサービス(株)と，平成元年に設立された(株)コンピュータ・テクノロジー・インテグレータ(CTIに改称)が，平成15年に合併して設立いたしました。以来，中部電力株式会社および中電グループ企業を中心に，IT分野をサポートしております。当社事業としては以下の分野があります。

- システム開発・システム保守
- ネットワークシステム・インテグレーション
- 科学技術(環境情報，技術開発・解析)
- データセンター
- システム運用管理
- データエントリー出力サービス

2. アジャイル開発手法への取り組みの背景

- なぜ，アジャイル開発手法に取り組んだのか -

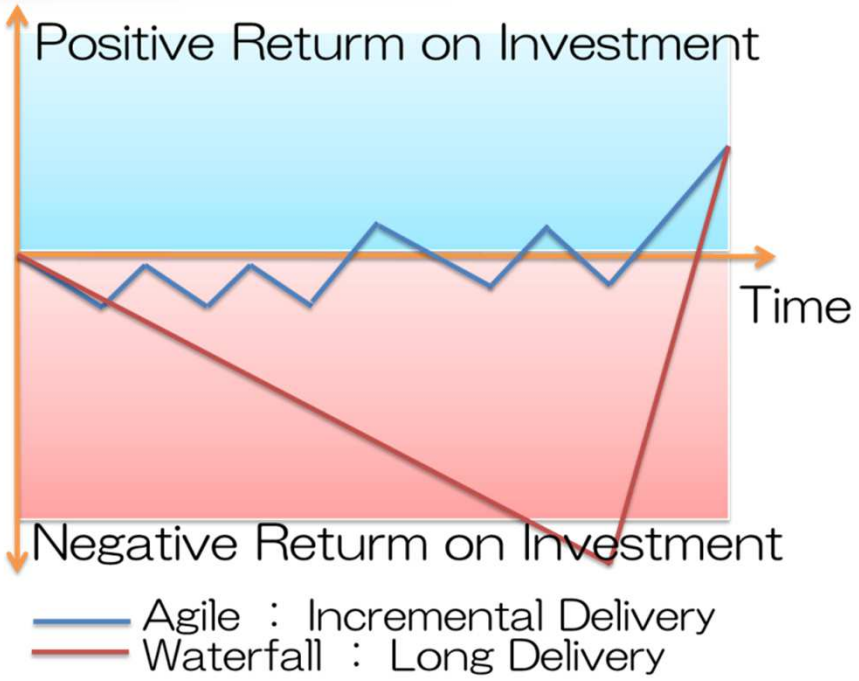
電力システム改革への対応スピードと柔軟性



求められるITサービス

(1) 従来のシステム開発

①決められた要求に対して工数・期間を設定
　ウォーターフォール型開発
②要件定義の後で設計・開発を開始
　プロジェクト全体の予算・スケジュール



Change of the IT service

(2) 今後求められるITサービス

①新たな業務・業務改革に対して必要な機能を迅速に提供
　要求に対する優先度を明確化してスモールスタート
　パッケージ適用，アセット活用，クラウドサービス
②限りある期間・予算の中でシステムを改修
　ビックバン型の再構築プロジェクトを極力回避し，既存システムの継続利用

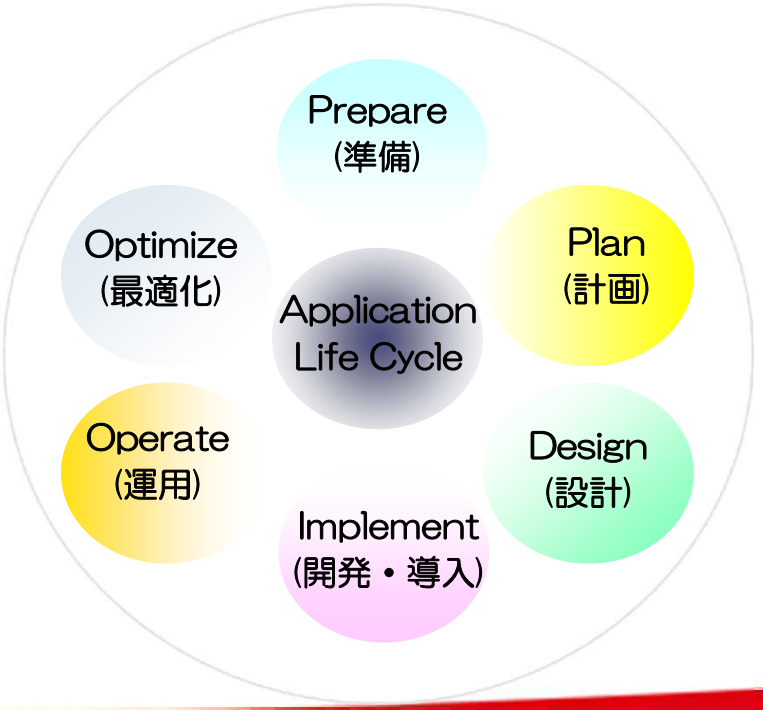
これを実現するためには？

求められるITサービス

「開発」「保守」「運用」を一体のシステムライフサイクル・プロセスとして捉え，その中で「安定」と「変化」を両立させる「短期・小規模」開発（改修）を繰り返し行うことで，お客さまの日々変化する業務にも確実に対応することも含め，業務継続をサポートすること。

①安定的かつ継続的なサービス提供

②変化への確実かつ迅速な対応



ITサービスの必須条件と対応施策

①安定的かつ継続的なサービス提供

②変化への確実かつ迅速な対応

【テーマ1】既存システムを活用した強みのさらなる強化

施策1：業務とITのモデル化とその活用スキル向上

施策2：既存システムの変化対応力強化の為のリファクタリングと再利用

【テーマ2】「保守」と「開発」を統合した新しい開発スタイルの確立

施策3：小ロット短期開発

施策4： ITサービス提供基盤

【テーマ3】システム・リノベーション手法の確立

施策5：アプリケーション最適化を含むリノベーション手法

施策6：必要とされる技術の選定とソリューション

【テーマ4】プロセス・マネジメント手法の確立

施策7：継続改善を実現するためのプロセス・マネジメント手法の
システム開発・保守への適用

アジャイル開発手法の導入

3. アジャイル開発手法適用へのこれまでの挑戦

- 保守案件へのアジャイル開発手法の適用 -

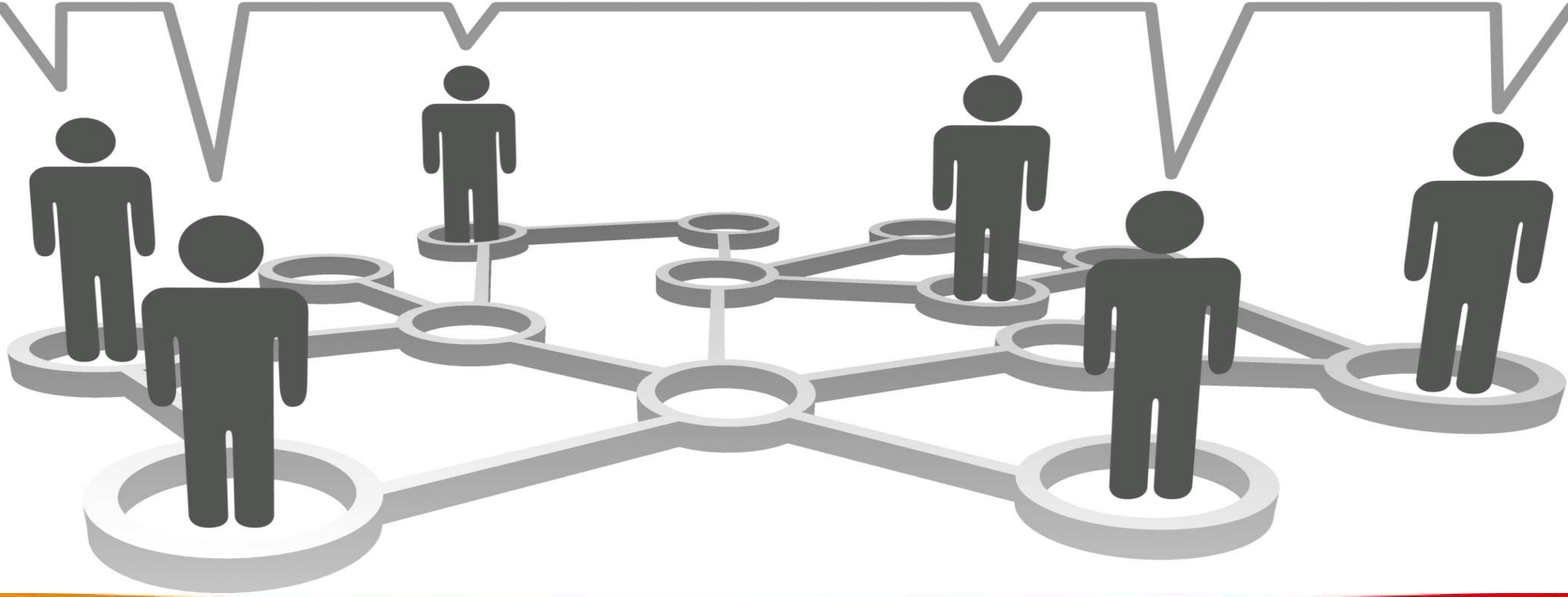
3. アジャイル開発手法適用へのこれまでの挑戦

3.1　アジャイル開発手法適用結果について

アジャイルへの拒否感と打開策

- 現場の拒否感

- なぜ今までのやり方を変える必要があるのか？
- 新しい事を取組むのはリスクが高い。
- やり方を変えることでオーバヘッドが大きい。
- アジャイル開発は，基幹系システムには適用できないのでは。



アジャイル導入の効果ある打開策

現場の拒否感を打開するにはメンバーの意識改革が必要

この目的を達成する[illegible]なプラクティスとして3Cリズム

- 3層による“調整(Coordi[illegible]laborate)”-”完成(Conclude)”リズム　を適用

各階層にPDSAサイクルを入れる

デイリーリズム	[illegible] デイ[illegible]ップ	collaborate 設計，PG，テスト	conclude コミット／ふりかえり
イテレーションリズム (スプリント)	[illegible]ordinate イテレーション計画	collaborate イテレーション	conclude リリース／ふりかえり
リリースリズム ※DADで拡張	coordinate 方向付け	collaborate 構築フェーズ	conclude プロジェクト完了／ふりかえり
PDSAサイクル	Plan 計画	Do/Act 実行/改善	Study 点検・評価・学習

DAD・・・Disciplined Agile Delivery

さらに，「ふりかえり」において，タスク，バックログの消化状況やリリース後の評価以外に，通常業務においてさらにムリ･ムダが無いか，改善点を提言するサイクルを併せる。

Checkではなく，学習し次につなげることを意識し，Studyとした。

アジャイル開発導入から定着・継続の道のり

チーム成熟度

不安，心配，
変化への拒絶

ステップ１

意識変革のために
アジャイルリズムの導入⇒定着へ
4か月

要求仕様の品質に
問題あり

チームの課題を
解決しよう

ステップ２

アジャイルリズムの継続
・要求仕様改善のための活動
3か月

テストが自動化
されると安心

チームのために
貢献しよう

ステップ３

アジャイルリズムの継続
・更なる効率化のためにテスト自動化
3か月

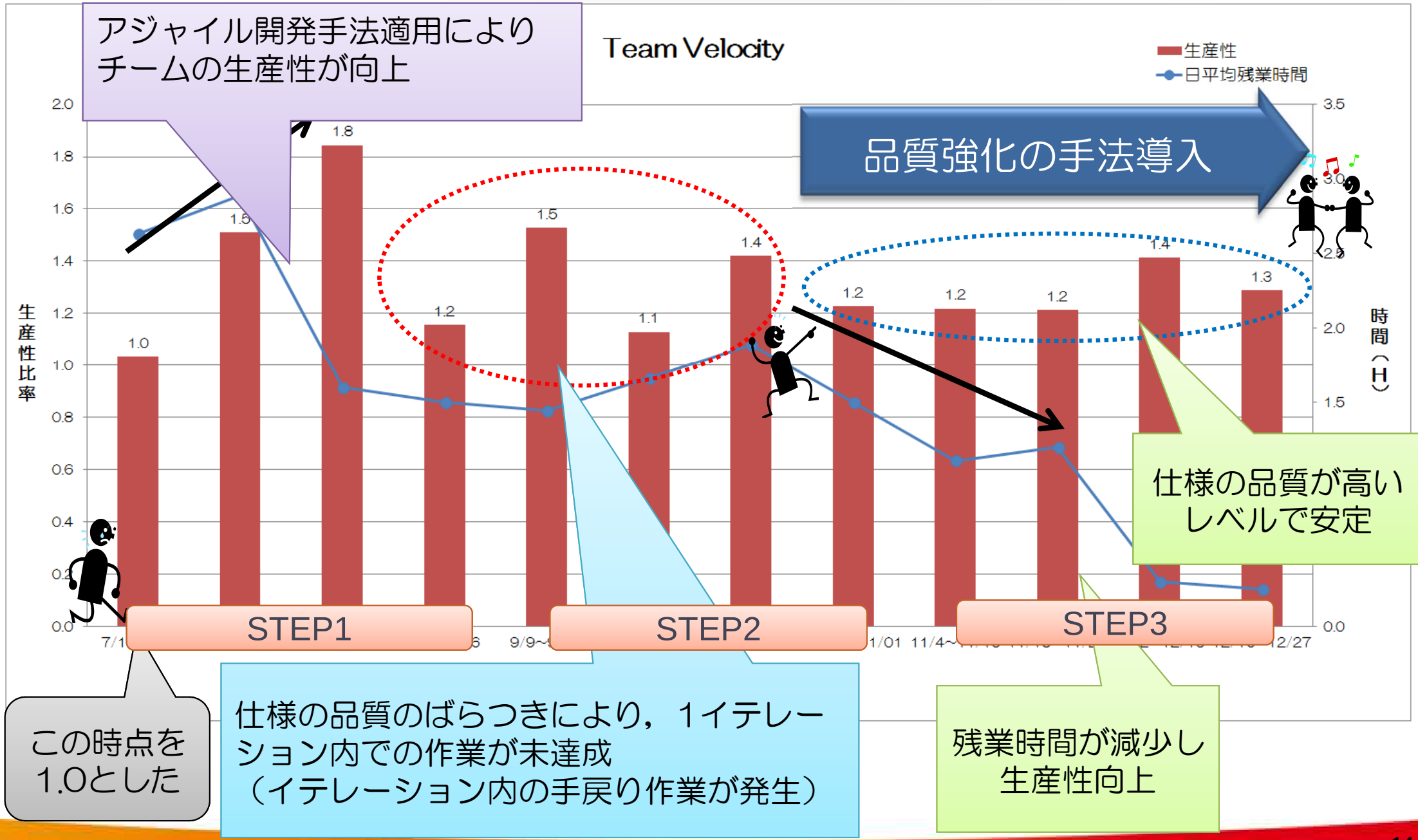
更なる改善に
挑戦しよう

課題は全てチーム
から提言

意識改革
効果の可視化
実践できる舞台を準備

品質・生産性

アジャイル開発手法導入による生産性向上への道のり



3. アジャイル開発手法適用へのこれまでの挑戦

3.2 アジャイル開発手法導入のステップ

保守案件アジャイル導入の概略スケジュール

ステップ１：アジャイルの基本の型をそのまま取り入れて実践する。
合わせて，アジャイルの基盤ツールとして共通作業基盤を導入する。
ステップ２：改善のサイクルにより，アジャイル基本の型を自分たちのやり方に改良する。
（品質改善活動の導入など）
ステップ３：アジャイル開発のテスト，品質の領域にツールを含めた改良に着手する。

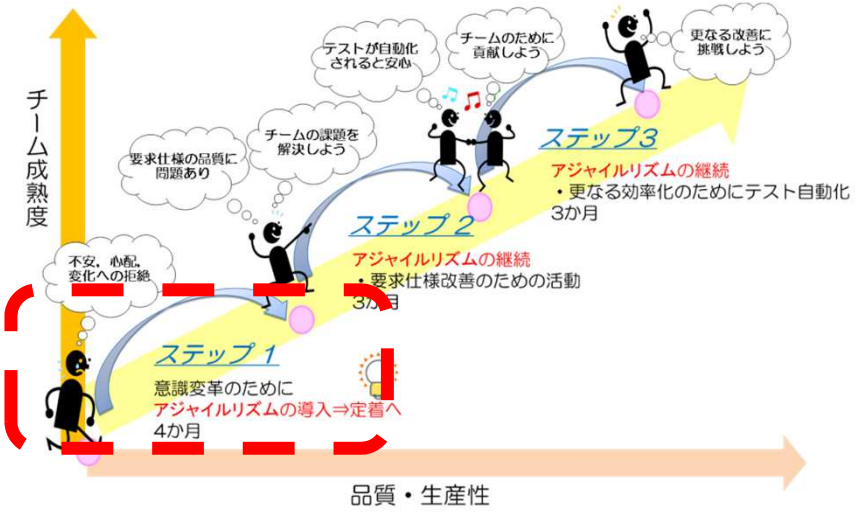
	6月	7月	8月	9月	10月	11月	12月	1月	2月	3月
	アジャイルの立ち上げ	共通作業基盤による計画づくりと進捗管理	共通作業基盤によるソース管理を追加	共通作業基盤によるビルド自動化	品質改善活動の実施・欠陥分析	品質改善活動の実施・欠陥分析	品質改善活動の実施・欠陥分析	回帰テスト，テスト自動化の導入	回帰テスト，テスト自動化の導入	回帰テスト，テスト自動化の継続
ステップ1 アジャイル導入の評価 ・アジャイル開発	ワークショップ	構築1	構築2	構築3	構築4	構築5				
ステップ2 アジャイル継続 ・アジャイル開 ・品質改善							構築11			
ステップ3 アジャイル展 ・アジャイル開 ・テスト自動化								構築12	構築13	構築14	構築15	構築16	構築17

Step2以降はアジャイルチームのPDSAから提言された施策を実施するように運営。

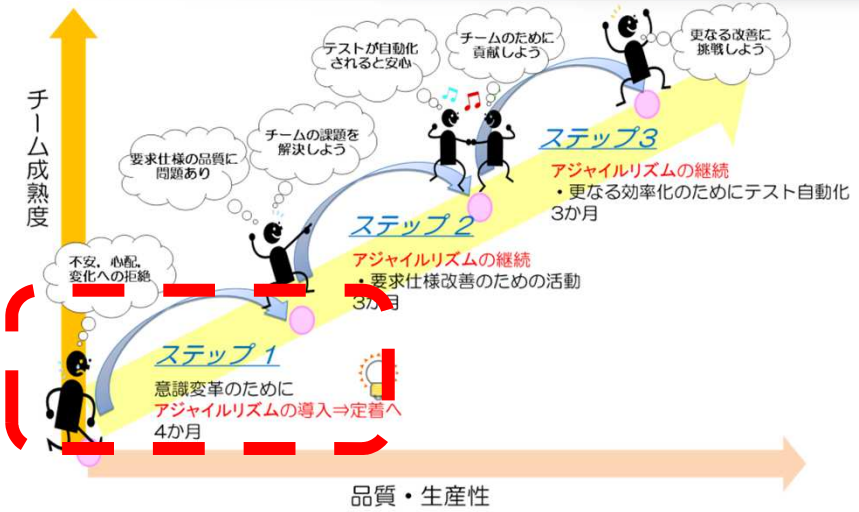
選択するプラクティスにかかわらず，アジャイルの定着のためには意識改革のためには長期の導入計画が必要。
メンバーのモチベーション維持のためにも自主性を尊重。

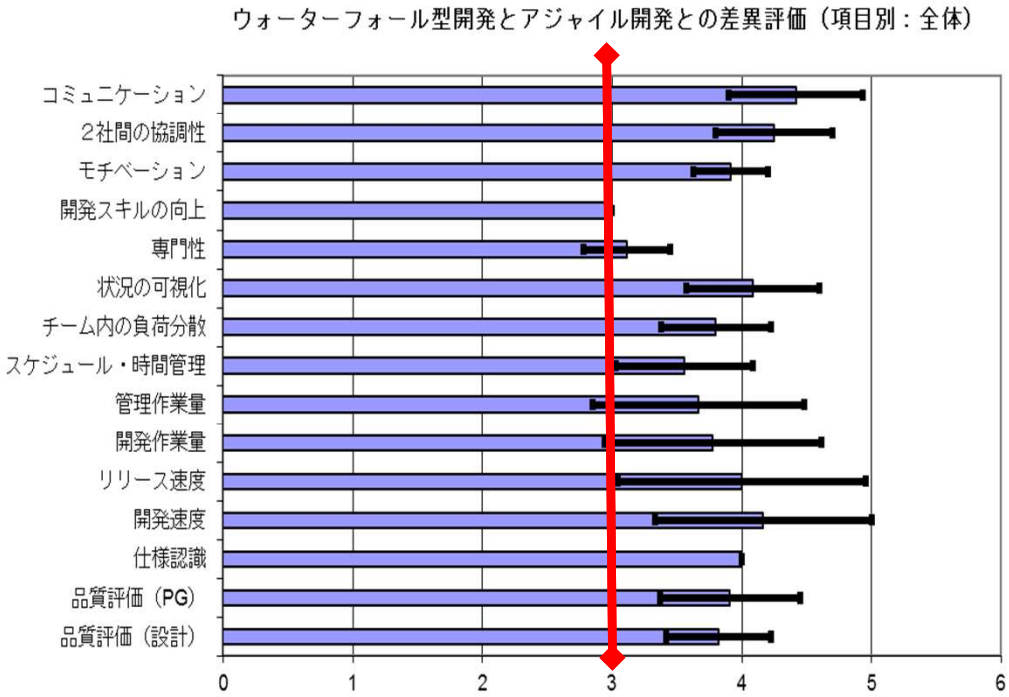
3. アジャイル開発手法適用へのこれまでの挑戦

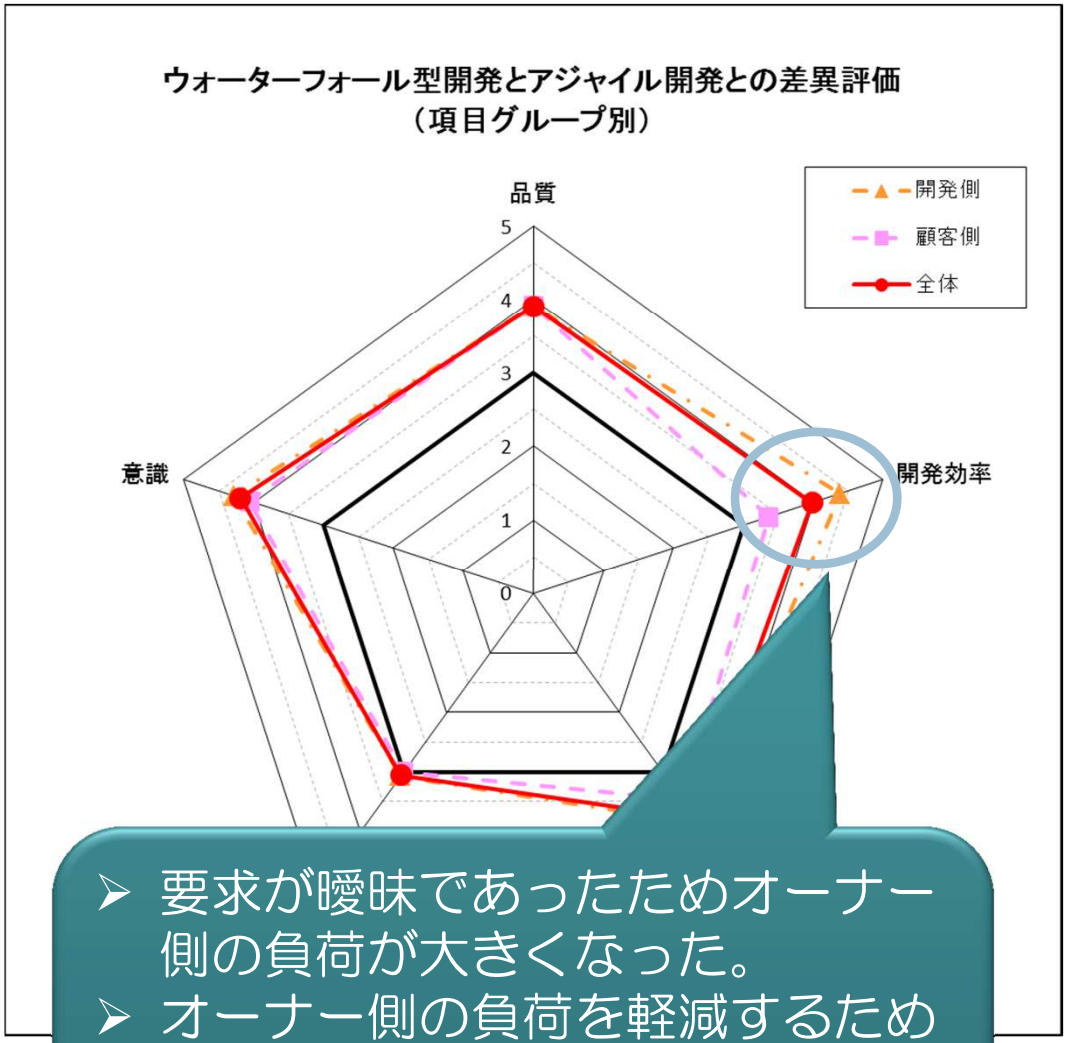
3.3　導入ステップ1について



STEP1のアジャイルとウォーターフォールの比較

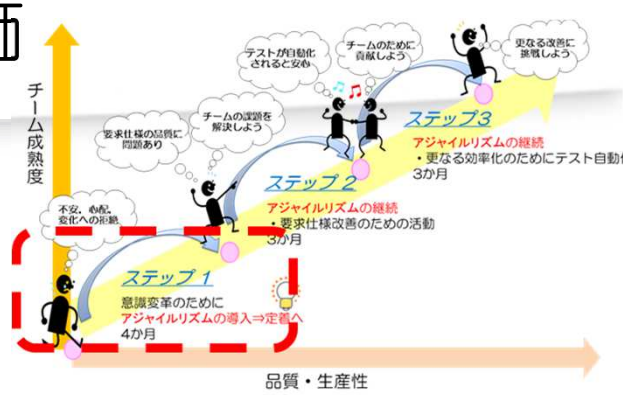






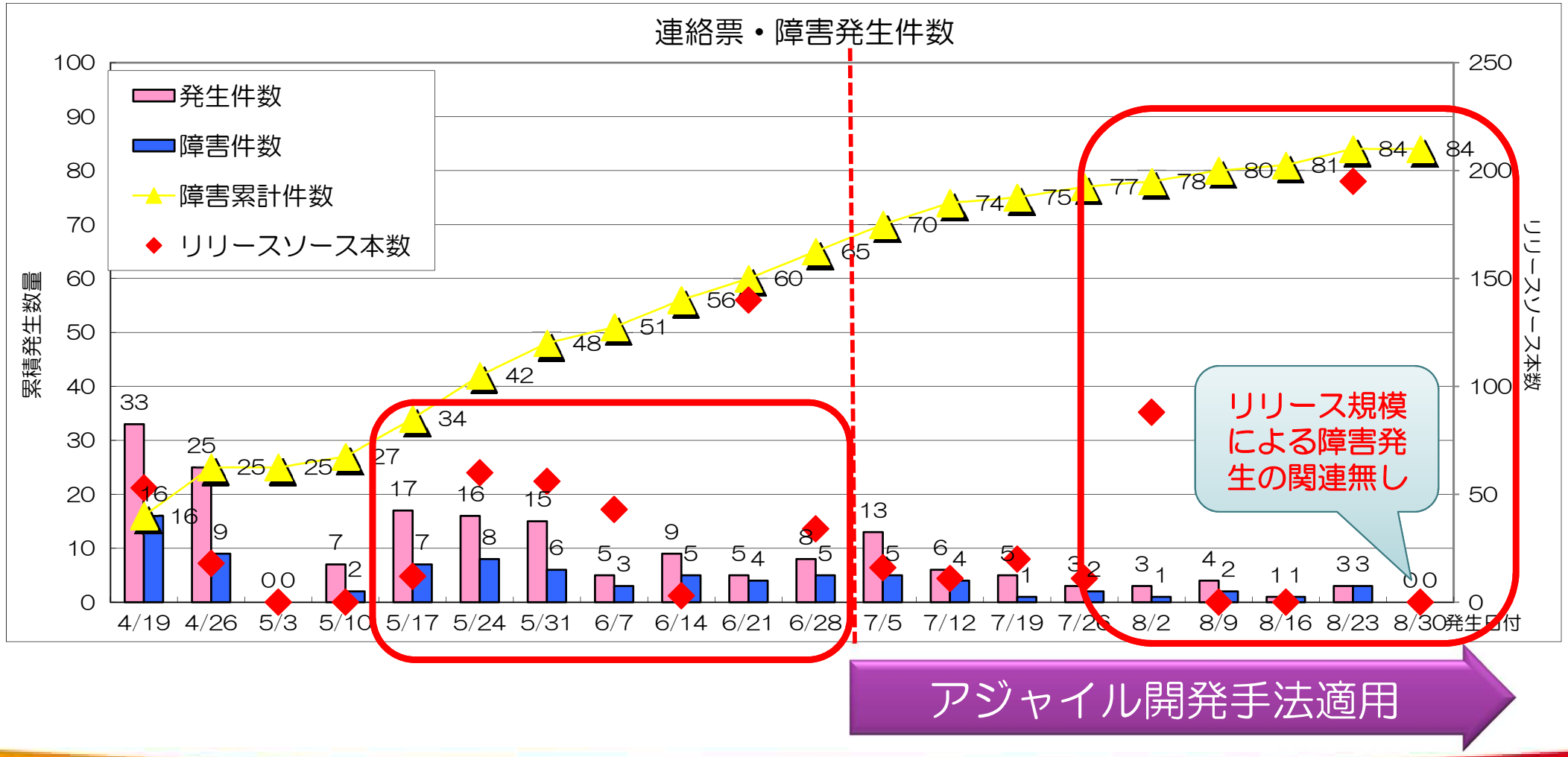
- 要求が曖昧であったためオーナー側の負荷が大きくなった。
- オーナー側の負荷を軽減するために要求提示段階で品質を確保する取り組みが必要。
  ⇒STEP2のテーマ

STEP1のアジャイル開発手法適用における品質評価



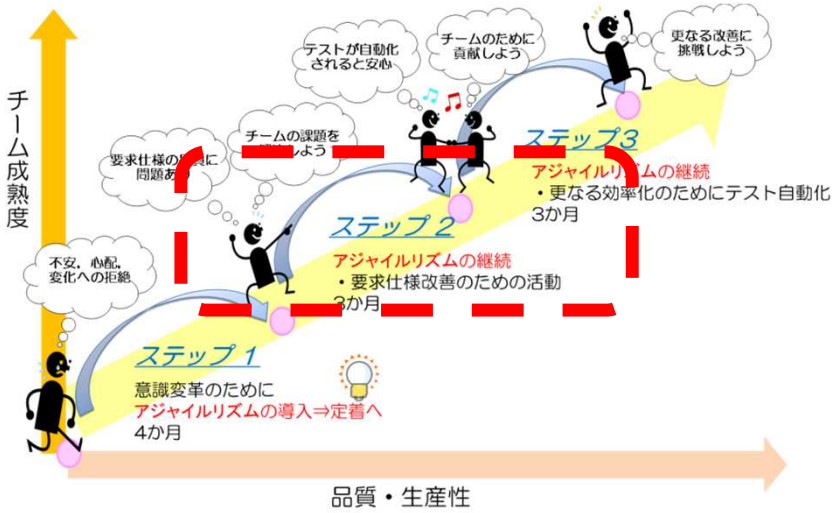
本番リリース後の障害件数を評価値としました。

一般的にはプログラム本数が多いほど，障害が多く発生する傾向となります。



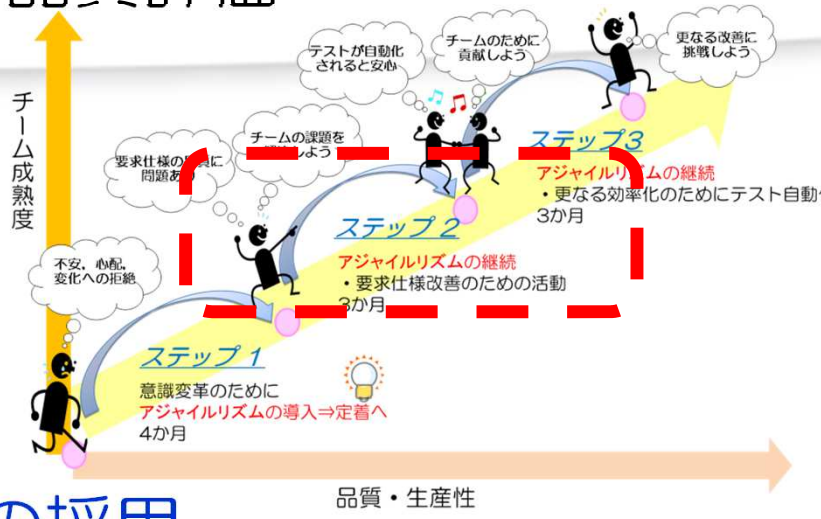
3. アジャイル開発手法適用へのこれまでの挑戦

3.4　導入ステップ2について



STEP2のアジャイル開発手法適用における品質評価

要求提示段階で品質を確保する取り組みが必要
⇒コストはかかるが最も欠陥検出・予防効果の
高いフォーマルインスペクションを採用



Formal Inspectionの採用

High
Low
Cost

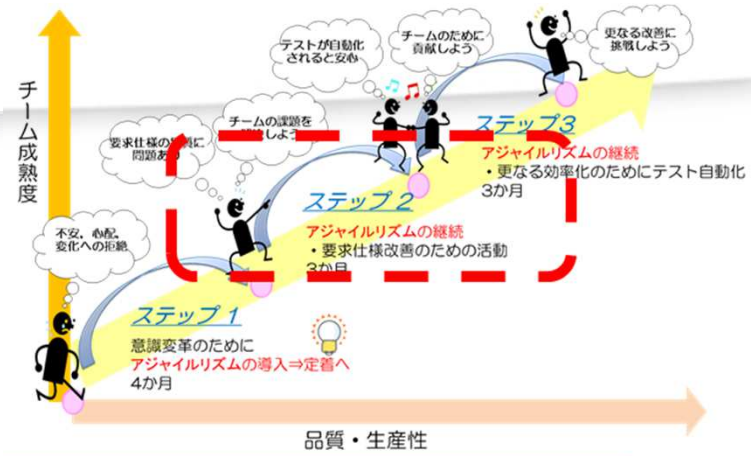
Formal Inspection
Team
Waikthrough
Pair Programming
Per Deskcheck / Passaround
Ad hoc Review / Casual-Leaming

High
Low

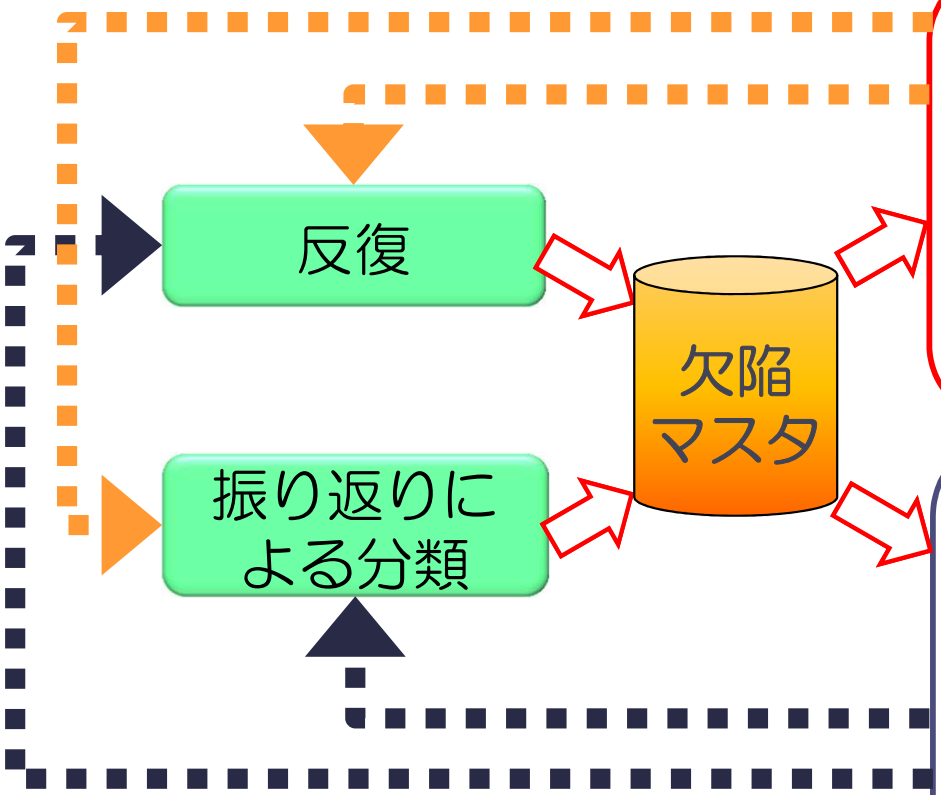
コスト軽減のために、欠陥マスタを使用
⇒ふりかえりにて欠陥マスタの共有と蓄積を実施

orporate sales Division. 2002

品質強化における効率化施策について

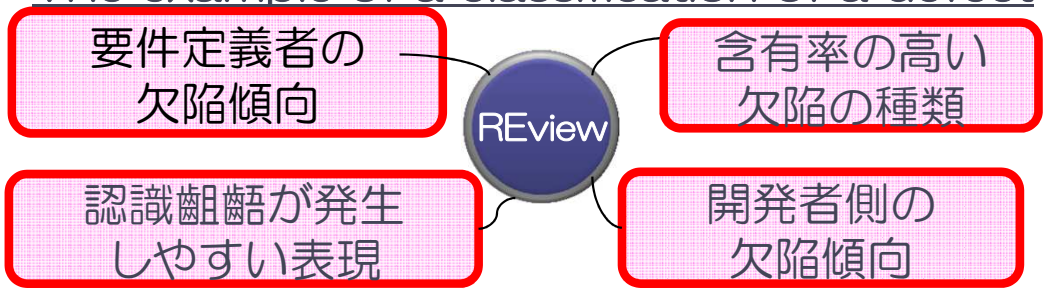


フォーマルインスペクションの効率化を図るために，欠陥マスタを整備・活用
⇒レビュー速度と品質の向上，およびメンバーのレビュー能力向上と均一化



欠陥の分類

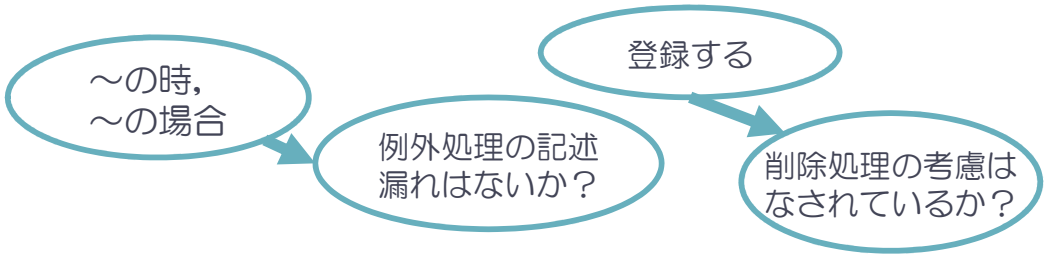
The example of a classification of a defect



A defect with many hand return man days in a back process is detected preferentially.

ガイドワード

The example of the collection of guide words



ガイドワード集を増やすことで，レビュー効率の向上と作業者への意識付けを図る

STEP2のアジャイル開発手法適用における品質評価

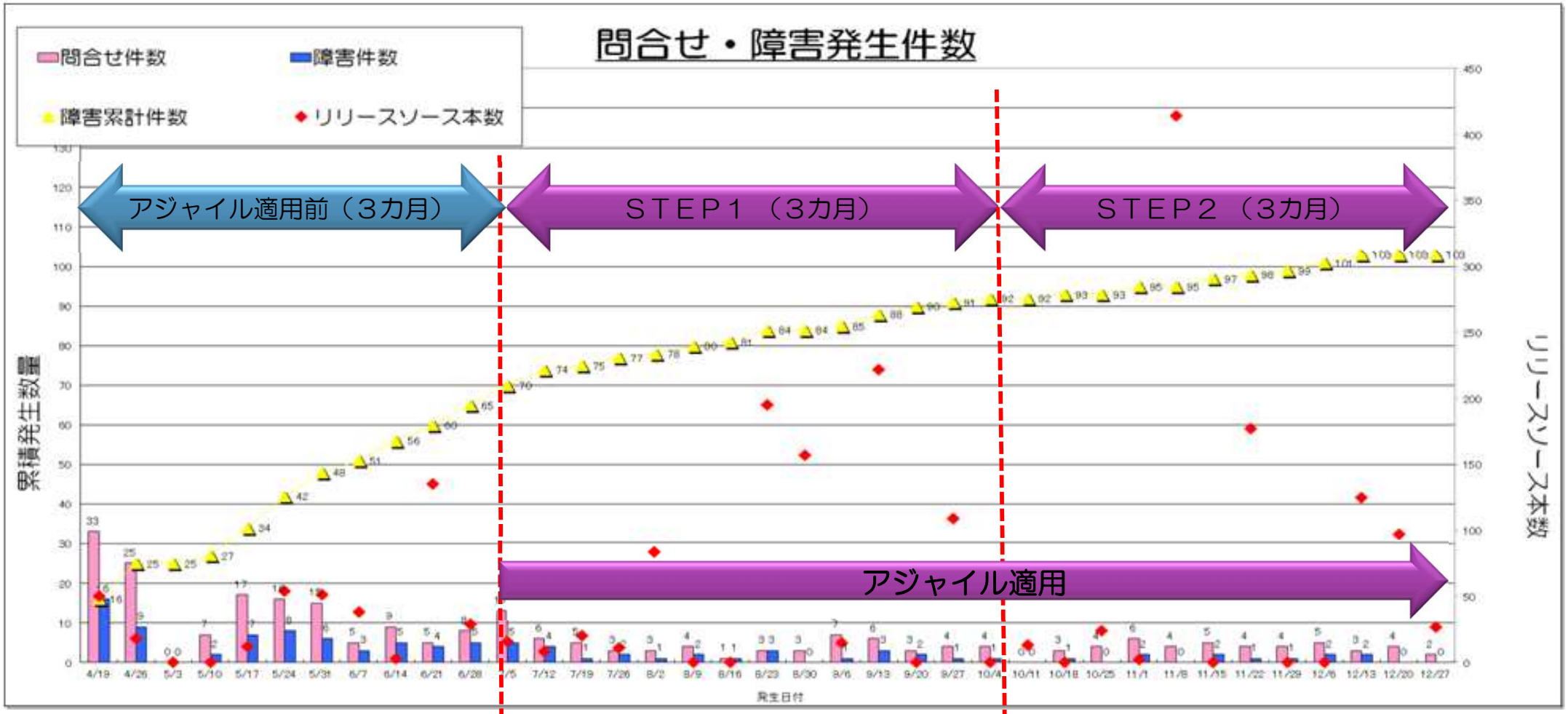
欠陥マスターを利用し，フォーマルインスペクション手法によるビューを実施したことによる，仕様の欠陥検出結果を以下に示します。

検出欠陥数：　1.6 倍

	従来のレビュー	フォーマルインスペクション レビュー
実施人数	４名	4名
総工数	80人・時間	76人・時間
検出欠陥数	26 箇所	41 箇所
レビュー密度	１．３ 個 / 時	２．２ 個 / 時

STEP1とSTEP2の品質評価と生産性

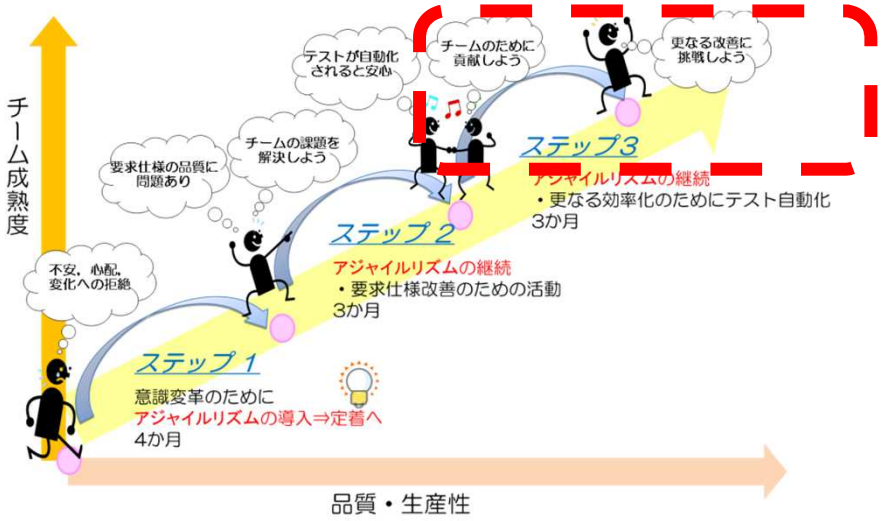
- ３カ月の平均値として比較するとアジャイル開発手法の適用後は品質が安定
- 生産性としてはSTEP２はアジャイル適用前の1.6倍



アジャイル適用前（３カ月）	ＳＴＥＰ１（３カ月）	ＳＴＥＰ２（３カ月）
リリースソース本数：390（本） 開発者：２（人） 生産性：１８０（本／人）	リリースソース本数：836（本） 開発者：４（人） 生産性：209（本／人）	リリースソース本数：879（本） 開発者：３（人） 生産性：293（本／人）

3. アジャイル開発手法適用へのこれまでの挑戦

3.5　導入ステップ3について



STEP3時点でのアジャイル開発手法に対するオーナー側の意見

<＜アジャイル導入時点＞

- 新しい取り組みを，先行してやる必要はない。
  （他社の導入実績を確認したうえで取り組みたい）
- 新しい取り組みにより，新たな問題が増えそう（不安）
- アジャイル開発により遅延が発生するようであれば，
  即時に中止したい。
- オーナー側（情報システム部）の作業が増えそうなので取り組みたくない。

＜STEP1完了時点＞

- 開発・進捗状況が明確になり，合理的な進め方が可能になった。
- 確認時点での不具合が減り，品質が向上した。
- 無駄な開発が削減され，開発費用が抑制された。
- 以前よりも作業と責任範囲（仕様の責任）が広がったように感じる。
- オーナー側と開発側の連携が密になり，いち早く価値を享受できることが実感できた。

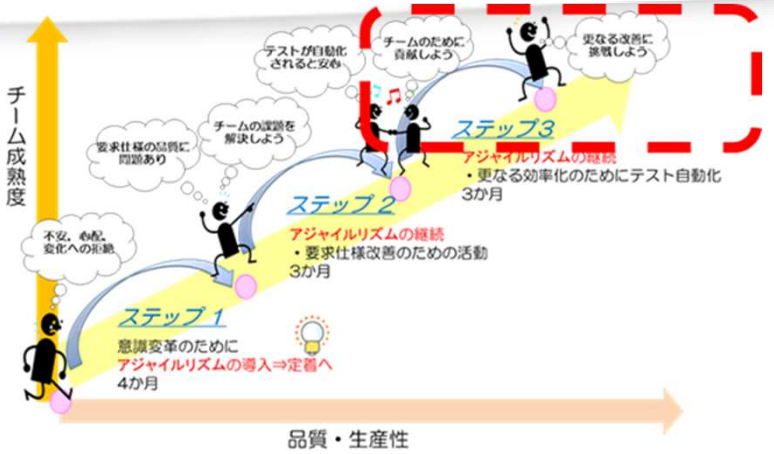
＜STEP3時点＞

- コミュニケーションが密になり，リリースまでの期間が短くなった。
- アジャイル開発に対して拒絶感がなくなった。
- 作業量は増加したものの，主管部署との仕様調整の重要性が明確となった。
- 今の開発体制を維持したい（維持するために予算等の問題はある）。

STEP3時点でのアジャイル開発手法に対する開発側の意見

＜アジャイル導入時点＞

- 運開直後であるため，新しい運用ルールやツール導入はリスクが高い。
- 突発作業がある状況下で，新しい取り組みをすると遅延が発生する。
- アジャイル開発は，要望が発散し作業量が増えそう。
- アジャイル開発を取り入れたことによる失敗が怖い。



＜STEP1完了時点＞

- 構成管理等の負荷が下がり，作業時間（設計，PG）を十分に確保することができる。
- 短い間隔でリリースができるため，短期間で達成感が得られる。
- 以前よりも顧客要望が理解度が上がり，修正方法・箇所が明確になった。
- 顧客側が，開発側の状況をリアルタイム確認できるため，言い訳ができない。
- お互い優先度に関して共通認識があるため，無理な要求が抑制できる。
- アジャイル開発への不安はなくなった。

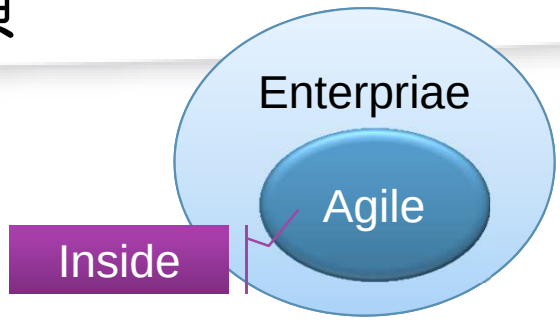
＜STEP3時点＞

- 品質に対して，結果の評価だけでなく，プロアクティブな取組みに意識が変わった。
- アジャイル開発をすることが当たり前になった。（改善活動への拒絶感がなくなった）
- 仕様レビューのやり方，考え方が大幅に変わった。
- 各々の要求の目的を考えるようになった。
- 今回のアジャイル開発に参加（経験）できたことを感謝している。

3. アジャイル開発手法適用へのこれまでの挑戦

3.6 アジャイル開発手法導入時の留意事項

アジャイル開発手法導入における留意事項



ウォーター・スクラム・フォールの蔓延

ウォータースクラム＋スクラムフォールへ陥らないこと。

「要件検討で時間をかけて構築に着手しない」，「スクラムで開発しているがデリバリーしない」では，お客さまに価値が届かない。これを如何に無くすかが進め方のポイントとしました。

単なるスクラムではなく，Enterprise Agileであること

「Enterprise＋Agile」というわけではなく、AgileをEnterprise(事業・会社・企業レベル)の中に取り入れる必要があります。つまり，無秩序でスクラムを実践するのではなく，計画・規律のあるアジャイル開発手法が必要です。

ウォーターフォールで取り入れられる成果物駆動が昨今のEnterpriseにそもそも合わないという考え方に立ちます。

また，Agileは，“Agility“ではなく，”Just in Time”として捉えます。

※成果物駆動：要件定義からの各工程で成果物として100点を取ってから次工程に進むことになります。
その間にレビューやルールの強要が行われることになります。何が100点かという基準がない
要件定義まで100点を取るようにして，終わらないということに陥る可能性があります。

アジャイル開発手法導入における留意事項

重要な工程：方向付けフェーズ

アーキテクチャーの指針と実現性確認や実施計画と予算の割り当て，それらを合意するフェーズです。この時に「BxUF」に注意する必要があります。ウォーターフォールからアジャイルに移行するときは，この工程が「BxUF」となり，進まないことが見られます。

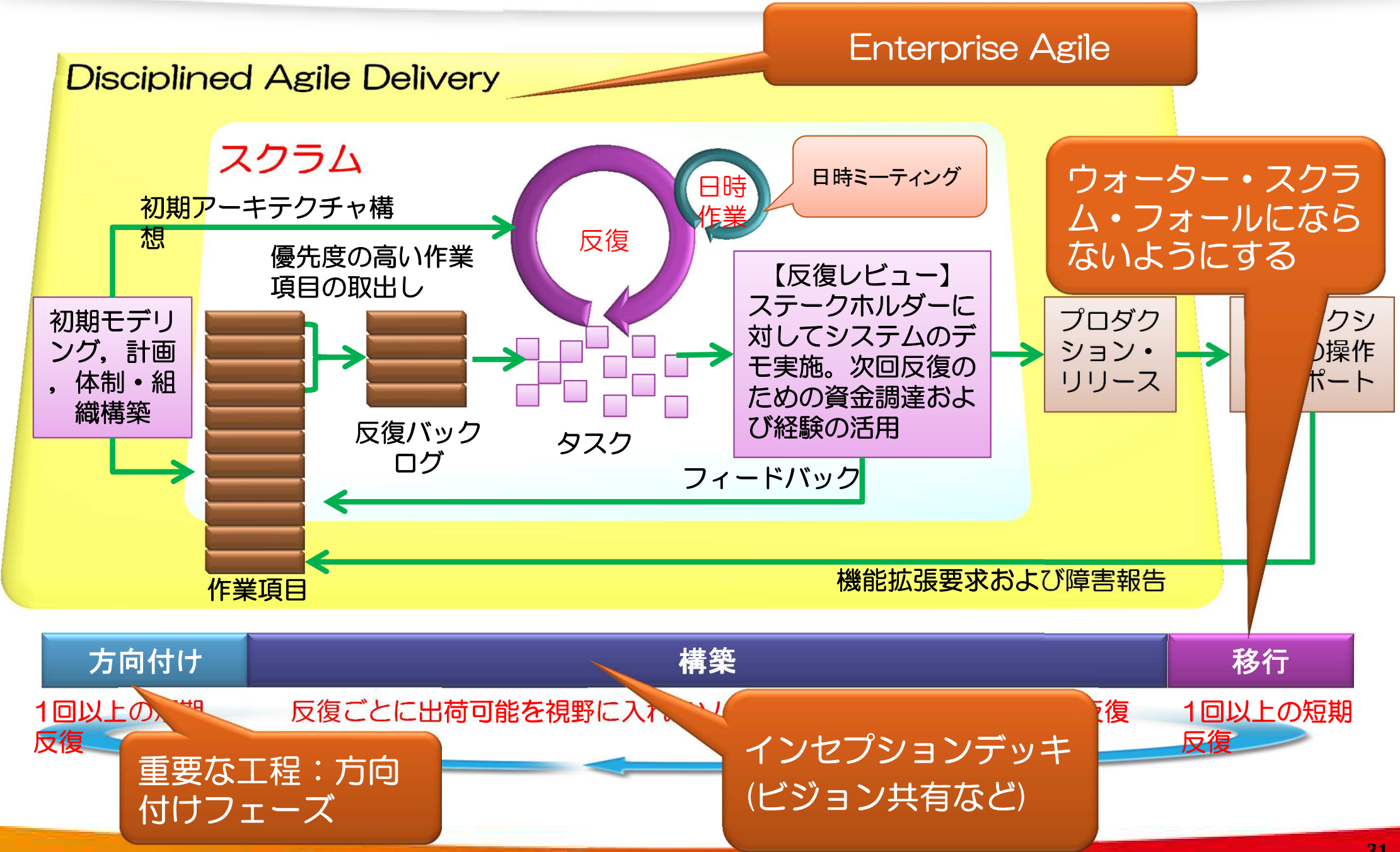
「初期に定義された詳細な要求や緻密な計画も憶測にすぎない。」

やりすぎない、作り込み過ぎない、実際に動くものを早く作ること。作ったものを検証するというプロセスに早く移行することは必要です。

意識の共有：インセプションデッキ

プロジェクトの目的・方針など(＝“ビジョン”)を共有しない場合は，素早く着手できることになりますが，構築フェーズでのイテレーションを無駄にしたり、間違った方向に進むこともあります。つまりビジョンは共有しておくことが必要であり，そのためのインセプションデッキが重要になります。

アジャイル開発手法導入における留意事項



4. エンタープライズ・アジャイルへの期待と準備

- エンタープライズ・アジャイル適用における準備 -

エンタープライズ・アジャイルで達成したいこと？

エンタープライス・アジャイルで達成したいことは・・・

ターゲット

大規模基幹系システム開発および大規模機能改修プロジェクト。

制約

あらかじめコミットした規模。

制約条件として与えられた費用およびお客さまの希望納期。

発生する課題への対応

仕様変更，品質悪化，認識相違による大幅手戻り，仕様漏れなどのリスクは排除できない。排除できないなら許容する。しかし計画通りにサービスインしなければならない。⇒リスク・テイクで乗り切る。BxUFに注意。

ゴール

期待される期限に確実にITサービスを提供すること。

その後のシステムにおいても都度発生する変更要望にJust In Timeで応えることができるシステムを構築すること。

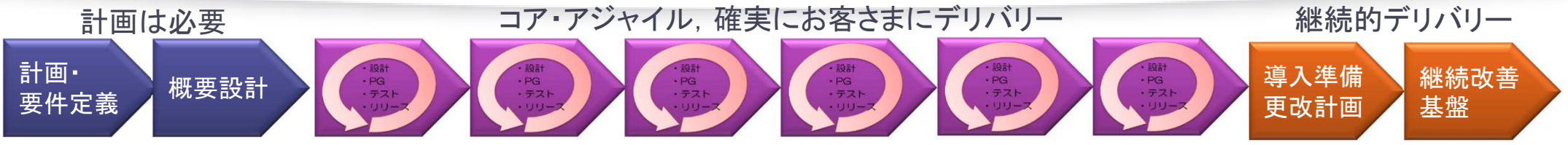
エンタープライズ・アジャイルで実施すべき主なプラクティス

＜要件定義（方向付け）＞
- 主要成功要因の合意（要件の明確化）
- パイロット開発（アーキテクチャの確立）
- 3Cリズムの導入と継続的な改善の取組み（チームビルディング）

＜製造・テスト（構築）＞
- 技術的課題へのTry & Catch
- 分業制ではなく全員参加による製造
- 標準化，共通化，再利用による効率化策
- DBの独立性確保によるデータ整合性の担保
- テスト結果，不具合発生・対応状況の可視化

＜リリース（移行）＞
- ビルド・デプロイ自動化による作業の効率化
- 作業タスクとPGの紐付けおよび的確な構成管理

エンタープライズ・アジャイルで実施すべき主なプラクティス



1回以上の短期反復　反復ごとに出荷を視野に入れたソリューションを生成する多数の反復　1回以上の短期反復

方向付け　**構築**　**移行**

- ゴールの共有
- チームビルディングとロールの明確化
- リズムの慣例化（3Cリズム）
- 要件の可視化とトレースの自動化
- アーキテクチャの確立と可視化
- プロダクト・パイロット開発
- プロセス・パイロット開発
- 成果の可視化（定量化）
- リリースの自動化
- 構成管理・変更管理の自動化
- プロダクトアウト中心の計画

- インスペクション技法
- 検証プロセスへの早期移行
- 作業の細分化(1バックログは5人日以内)
- 再利用・流用技術
- 計画の見直しへの柔軟性
- コア・アジャイルスキル

- 3つの移行（システム移行，データ移行，業務移行）
- 教育の充実
- DevOpe基盤
- アーキテクチャの更改計画
- 顧客からのフィードバック

5. エンタープライズ・アジャイルの挑戦

- 大規模基幹システム開発へのアジャイルの適用 -

【方向付け】
- ゴールの共有
- チームビルディングとロールの明確化
- リズムの慣例化（3Cリズム）
- 要件の可視化とトレースの自動化
- アーキテクチャの確立と可視化
- プロダクト・パイロット開発
- プロセス・パイロット開発
- 成果の可視化（定量化）
- リリースの自動化
- 構成管理・変更管理の自動化
- プロダクトアウト中心の計画

【構築】
- インスペクション技法
- 検証プロセスへの早期移行
- 作業の細分化(1バックログは5人日以内)
- 再利用・流用技術
- 計画の見直しへの柔軟性
- コア・アジャイルスキル

【移行】
- 3つの移行教育の充実
- DevOpe基盤
- アーキテクチャの更改計画
- 顧客からのフィードバック

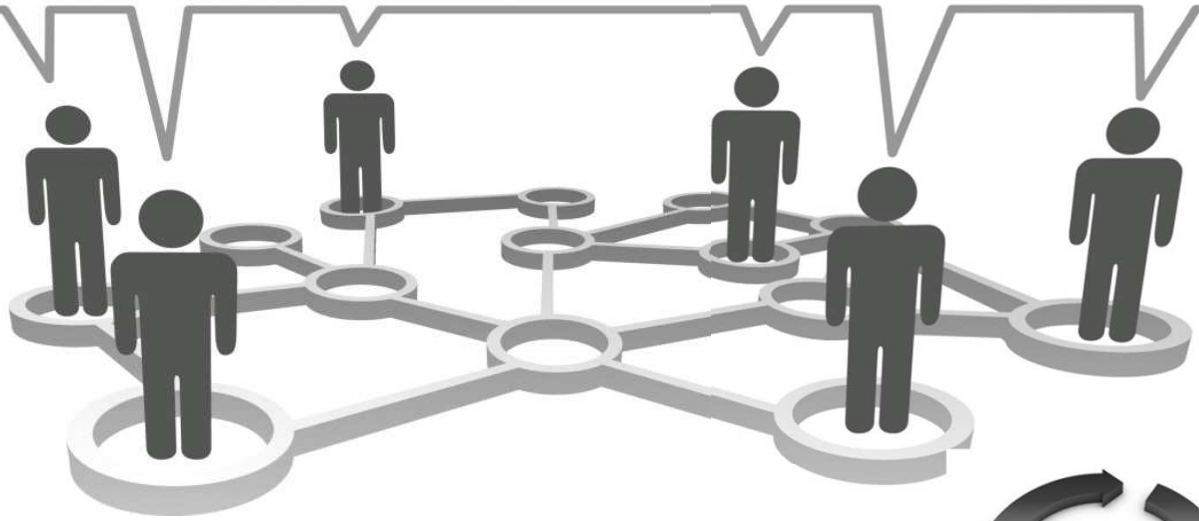
主なアジャイルプラクティス

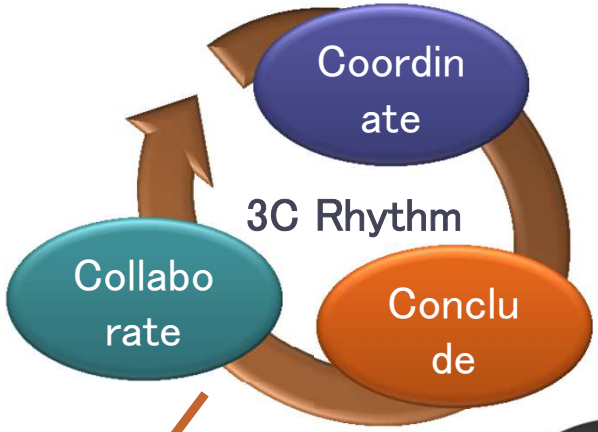
5.1　エンタープライズ・アジャイル導入のポイント1

エンタープライズ・アジャイルの導入　ポイント1

＜現場の拒否感への対応＞

- なぜ今までのやり方を変える必要があるのか？
- 新しい事を取組むのはリスクが高い。
- やり方を変えることでオーバヘッドが大きい。
- アジャイル開発は，基幹系システムには適用できないのでは。





PDCA

●プロジェクトの課題可視化
●チーム間のタスク意識付け

PDCA

●日々の活動から課題を抽出
●発表形式による情報共有

PDCA

●イテレーションリズムの構築
●「朝会」、「ふりかえり」
による軌道修正

チーム成熟度

あるプロジェクトのアンケート結果

【イテレーション計画】

ESVP	人数
探検家	11
買い物客	18
行楽客	1
囚人	0

主なメリット	主なデメリット	主なコメント
・直近の作業について，柔軟性，具体性の高い計画となる。 ・メ…本性を把…納得感…。	全体進捗が見えない。プロジェクトの全体像を把握できない。	品質計画，品質の担保という観点では何らかのアクションが必要。

3Cリズムに意識が偏りすぎ

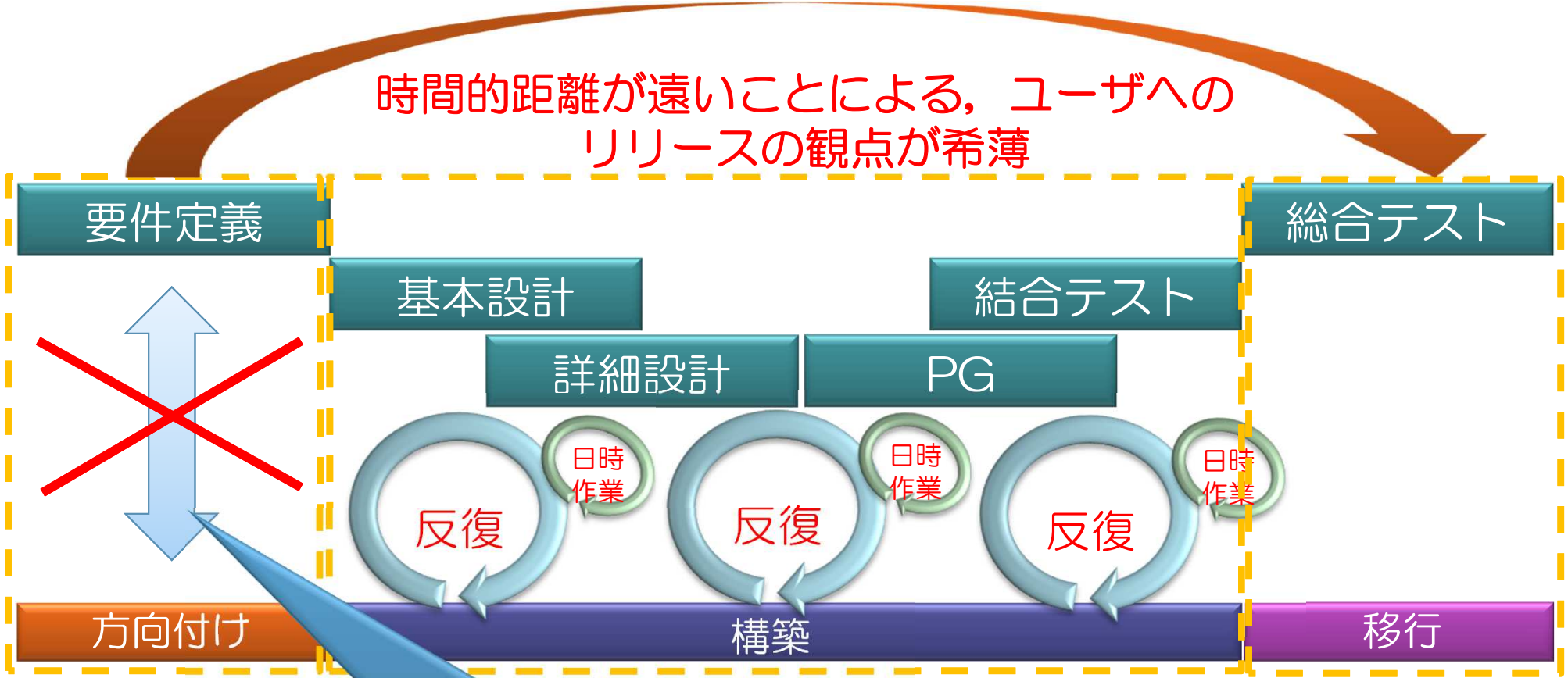
【朝会・ふりかえり】

ESVP	人数
探検家	11
買い物客	17
行楽客	0
囚人	2

主なメリット	主なデメリット	主なコメント
・情報・課題の共有を行う場として有効 ・チーム全体の作業内容，作業量が把握可能	・課題がない場合，時間が無駄 ・重要な情報を記録に残したり，課題をタスク管理する部分まで繋がっていない。 ・ツールの活用が必要。	品質計画，品質の担保という観点では何らかのアクションが必要。 ・人数が多くなると時間がかかるため，各々が時間をかけすぎない意識が必要

定量的な効果測定方法が無い

ユーザへのリリース視点の欠如



要件定義と方向付けは同じではない。

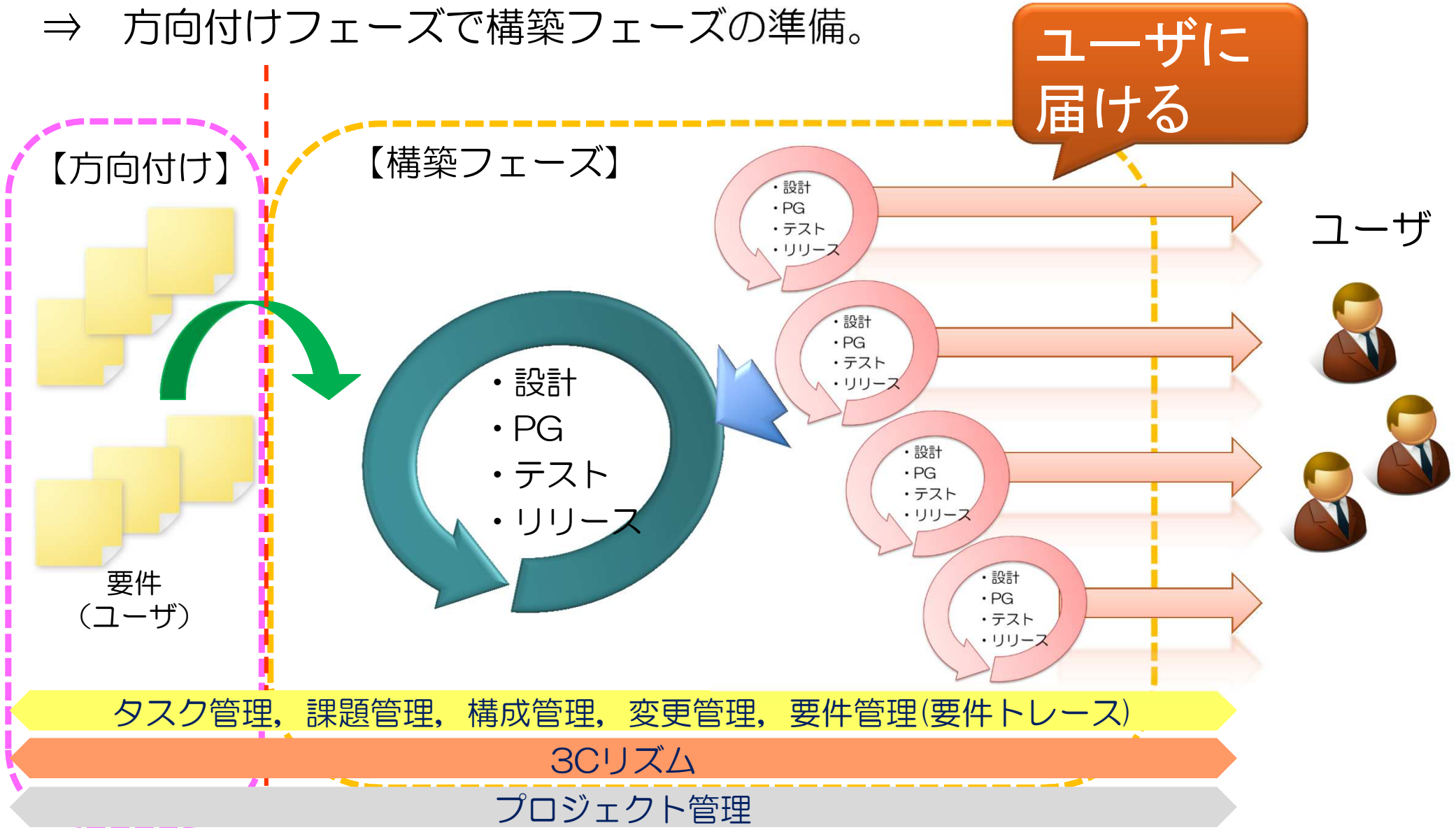
ユーザーへのリリース視点の重要性

5.2　エンタープライズ・アジャイル導入のポイント2

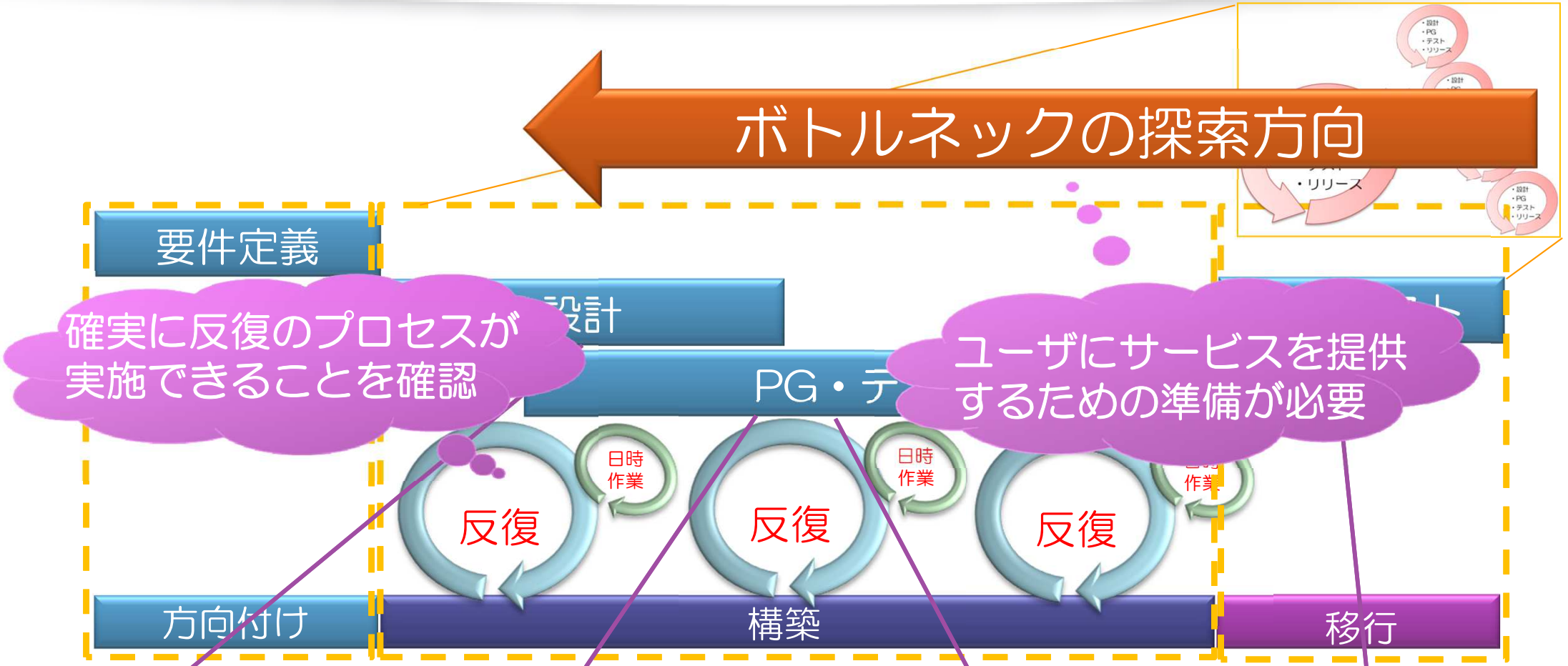
エンタープライズ・アジャイルの導入　ポイント2

構築フェーズでは，イテレーション毎に確実にデリバリーすることを重要視

⇒　方向付けフェーズで構築フェーズの準備。



構築フェーズに向けて必要な準備-ボトルネックを探す-



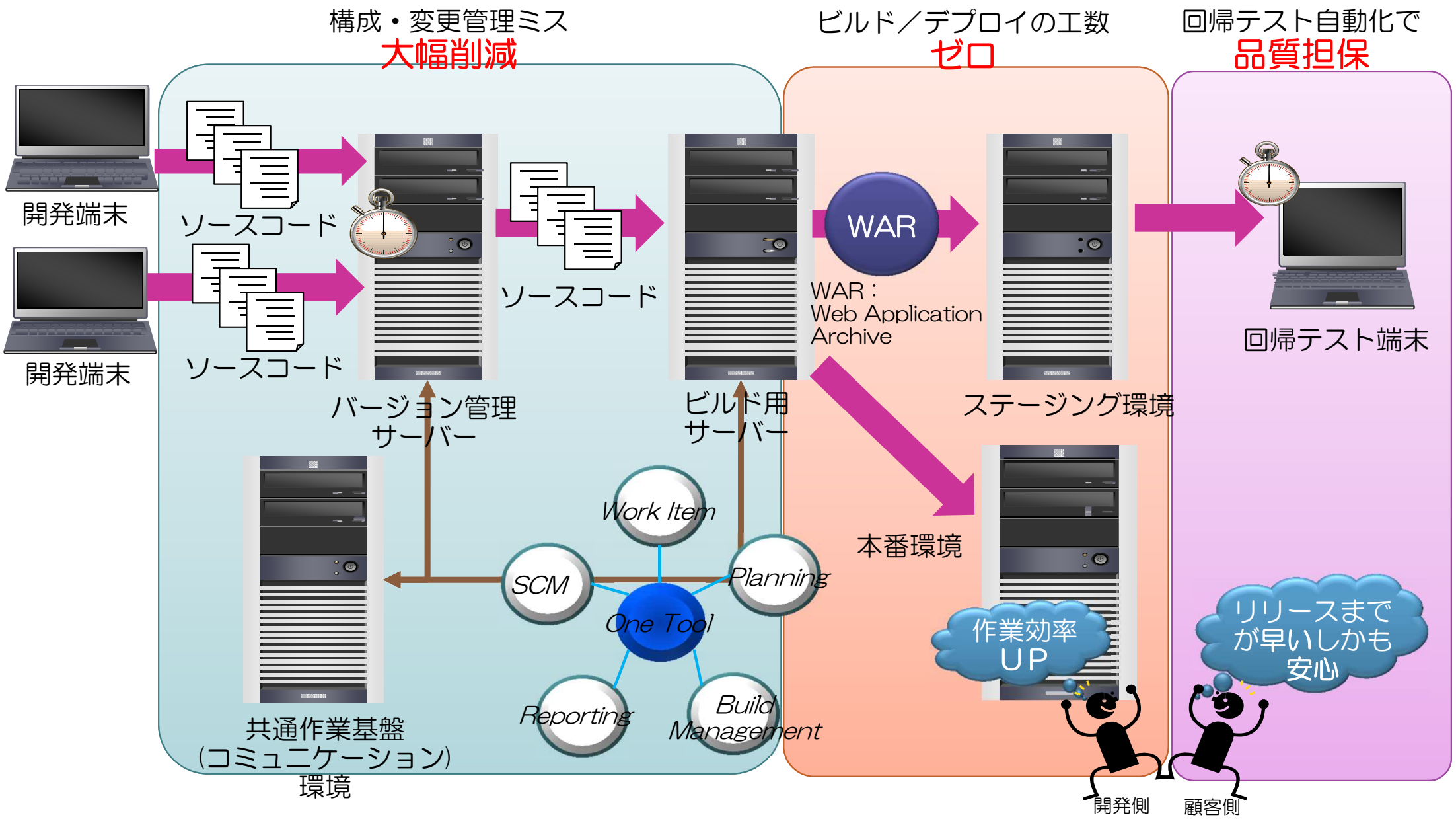
コーディング環境の整備
- コーディング規約
- フレームワーク
- ＤＢ配布
- 標準化・共通化

テスト環境の整備
- テスト計画
- テストケース
- テスト状況可視化
- 要件とのトレース管理
- テスト環境
- リリース・デプロイ自動化

ビルド環境の整備
- 継続的統合
- 単体テスト自動化

受入環境の準備
- 受入テスト計画
- 受入テスト環境
- 要件とのトレース管理
- リリース・デプロイ自動化

アジャイル開発を支えるなツールの例

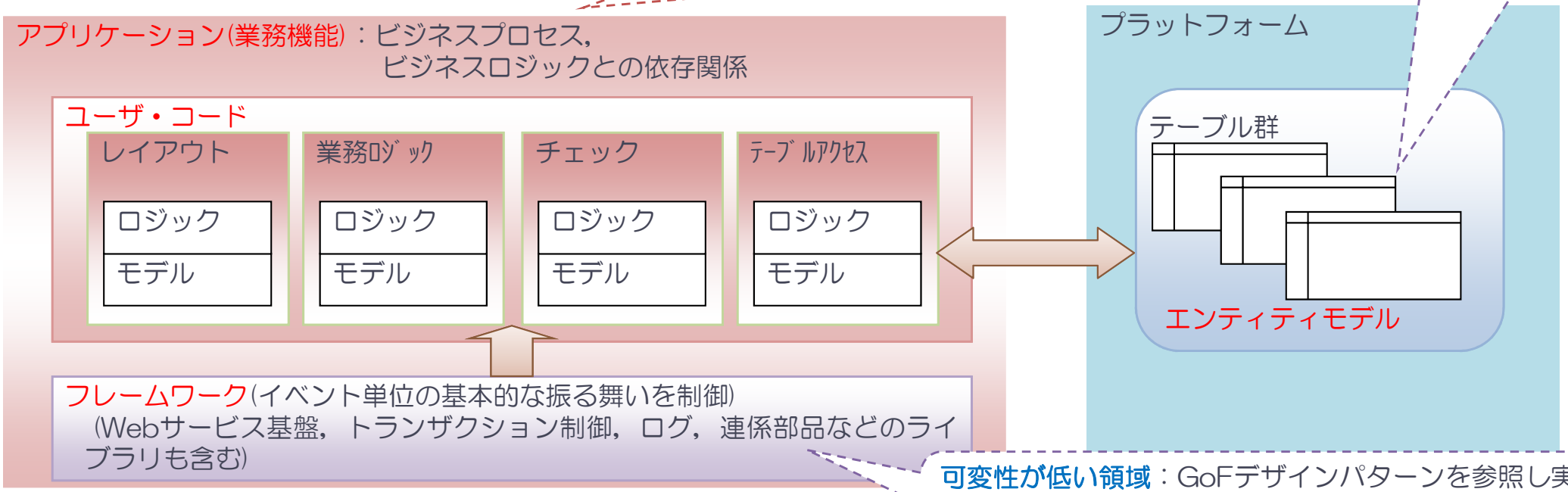


5.3　エンタープライズ・アジャイル導入のポイント3

エンタープライズ・アジャイルの導入　ポイント3

可変性が高い領域：開発では，標準化，自動生成，テンプレート可で対処する領域。この中でも可変性が低いものと高いものに分けられるものの，保守においては，変更要望への対応として従来より対応している範囲。ただし，変化対応が可能な仕組みになっている必要がある。

可変性が低い領域：ドメインモデルを活用し対処する領域(新規事業など対応が無い場合で，リプレイス案件の場合)。



可変性が低い領域：GoFデザインパターンを参照し実装され，IoCのフレームワークとして構造を定義した領域。構造を規定しつつ，可変性を支えるものとして定義。

非機能要件モデル
性能，356日24時間稼動，ダウンタイム許容時間などのサービスレベル

プラットフォーム
OS，DBMS，ApplicationServer，WebServer，セキュリティ，他システム連係基盤など

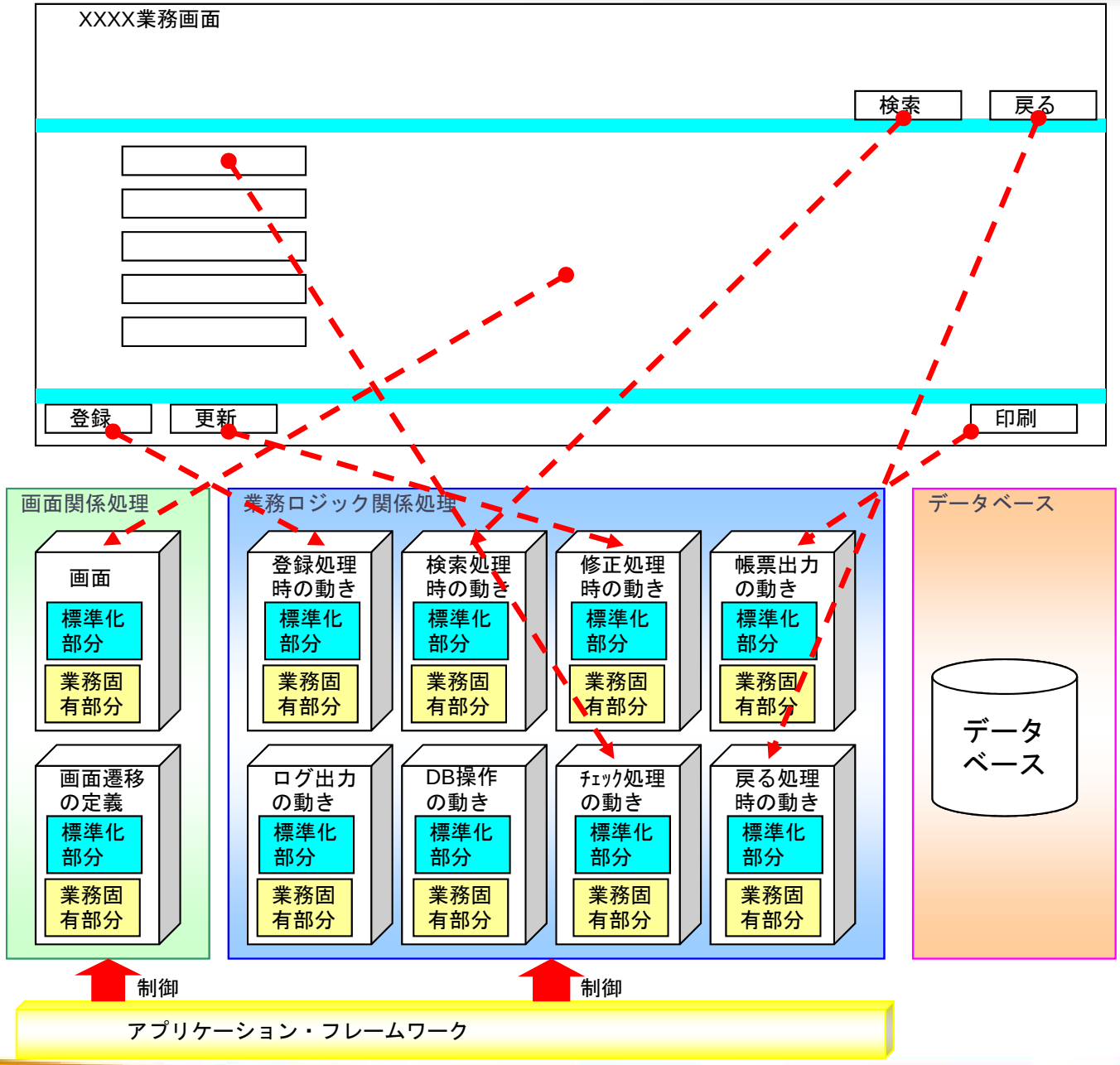
インフラ配置モデル

可変性が低い領域：既存の配置モデルを活用し対処する領域。

可変性が高い領域：標準化で対処する領域。

可変性が高い領域：標準化で対処する領域。

アジャイル開発に適したアプリケーションのモデル例



運用開始後の変更があるため，柔軟に対応したい。

イベント別にオブジェクトを作成

各オブジェクトは，共通部分と固有部分(ホットスポット)で構成

外部より各オブジェクトを制御(独立性)

7. 本日のまとめ

エンタープライズ・アジャイル導入の3つのポイント

POINT1：意識改革のためのアプローチ

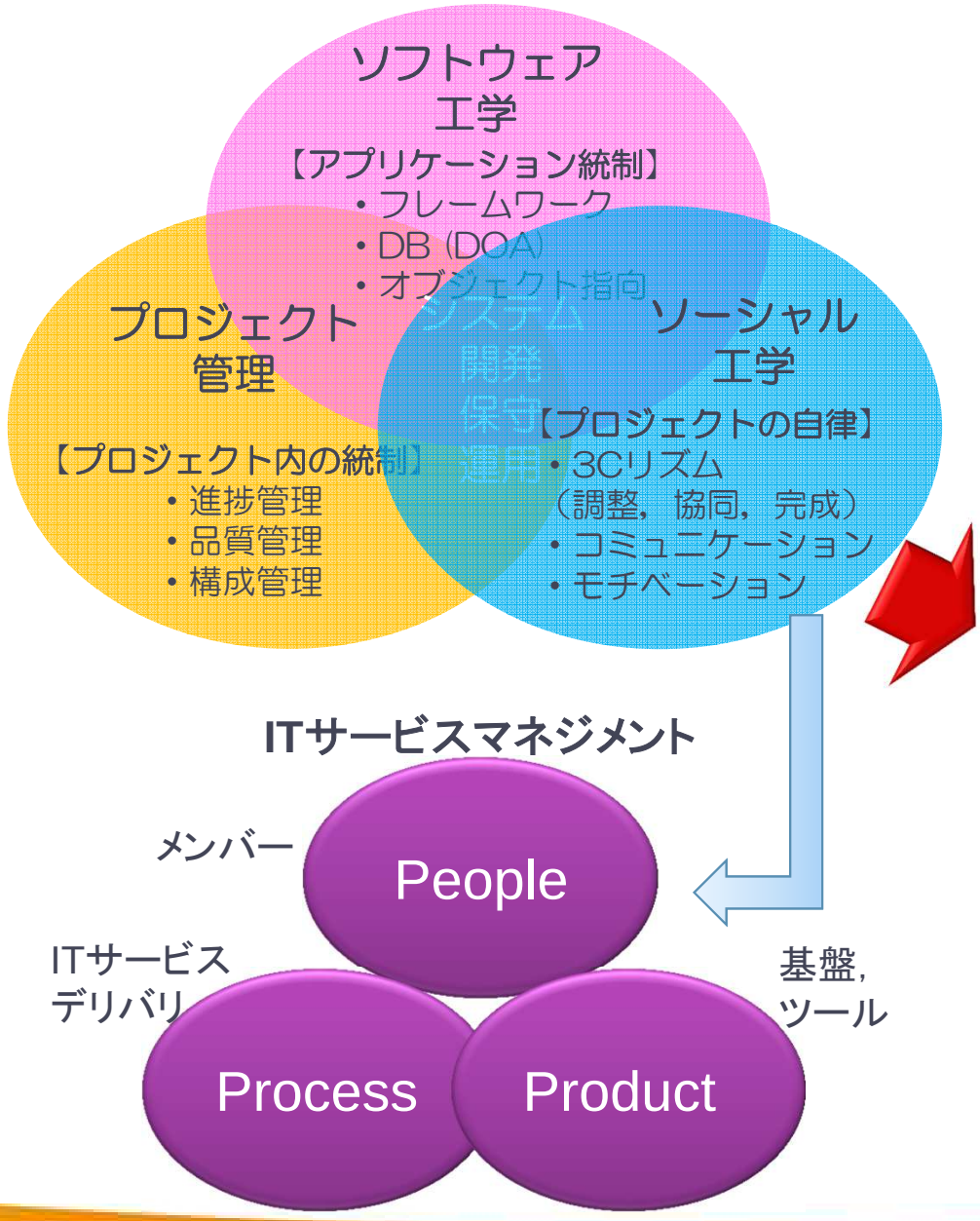
POINT2：期待されるITサービスを届ける仕組み

POINT3：アーキテクチャとしての“可変”と“不変”

アジャイル開発手法導入における留意事項

その他にも・・・

エンタープライズ・アジャイルの導入



Enterprise Software Delivery Method

Software

- *Framework/Library*
- *Data Oriented Architecture*
- *Plain Old Java Object*
- *Development Tools*
- *System Architecture*
- *Software test/deploy*
- *Refactoring*

Project Management

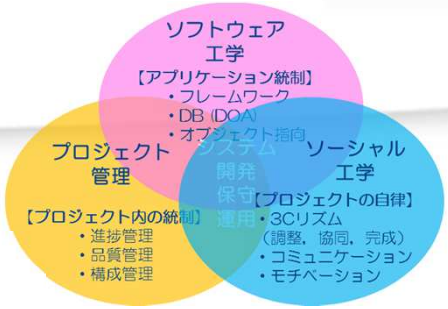
- *Planning*
- *Quality / Configuration / Cost / Risk Management*
- *Agreement promise*
- *Project manage*

- *Development • Operation • Maintenance • Renovation ⇒DevOps Process Cycle*

- *Agile*
- *3C（Coordinate, Collaborate, Conclude)*
- *Just In Time/Lean Startup*
- *Communication Integration*
- *Deming cycle(PDCA)*
- *Process Manage*

Social

エンタープライズ・アジャイルの導入



エンタープライズ・アジャイル

システムにより提供するサービス（開発・保守・運用）

ソフトウェア工学

- アーキテクチャ
- モデリング
- 開発手法
- フレームワーク
- DB（DOA）
- オブジェクト指向　等

プロジェクト管理

プロジェクト計画/運営（演繹法,帰納法）

- 品質管理
- 規模管理
- リスク管理
- 会議体管理
- ﾄﾚｰﾆﾝｸﾞ計画　等

ソーシャル工学

- 3Cリズム
- コミュニケーション
- モチベーション
- チームビルディング　等

- 進捗管理
- 課題管理
- 変更管理
- 体制管理
- ｺﾐｭﾆｹｰｼｮﾝ管理

システム規模

開発スピード

・ソフトウェア工学は、プロジェクト計画を実現するために、技術を提供する

・プロジェクト管理は、ソフトウェア工学で提供する技術を基に、実現できる計画を示し、管理する

・投入されるメンバーの数は、システム規模や開発スピードに比例する。すなわちシステム規模や開発スピードに応じて、ソーシャル工学の重要性が増す
※従来のプロジェクトで失敗した要因に、人に関わる活動が考えられる

⇒確実にプロジェクト計画を遂行するためにソーシャル工学の要素を取り入れている

新規開発時のプロジェクト管理と保守・運用時のプロセス管理の両者を網羅する

＜プロジェクト管理＞
PMBOKに準拠して管理を実施する
＜プロセス管理＞
DADのプラクティスを採用してプロジェクト管理手法をテーラリング

DAD：Disciplined Agile Delivery

プロジェクト計画は、目的に対するメンバー（人）の振る舞いは定義するが、具体的な進め方は定義していない

⇒プロジェクト管理の要素を用いながら、コミュニケーションやモチベーションといった人に関する要素を体系立てて支援することでプロジェクト計画を補完する

おわりに

アジャイル開発手法により構築したITサービスは，ユーザまで届けられて，初めて価値を生むものになります。

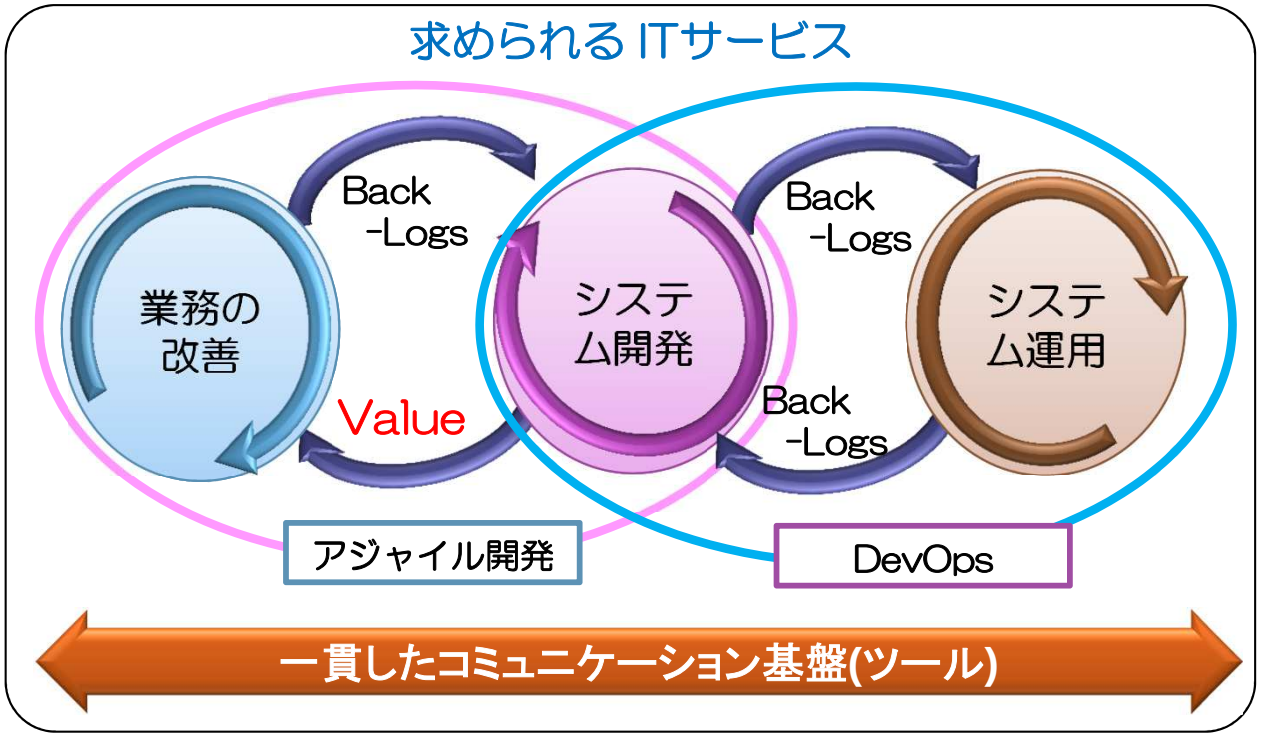
よって，ITサービスのデリバリーまでを確実に遂行できるプロセスが必要になります。

今までの開発スタイルにおいても，品質を確保したうえで，確実なデリバリーが実現できています。それと対立したものではないと考える必要があります。

従来の開発スタイルにアジャイルのプラクティスを取り込むという方法を採ります。

デリバリー後においても，ビジネス環境の変化に追随する必要があり，さらにITサービスは安定して提供し続けなければなりません。

変化と安定，Just In Timeのすべて実現することが必要です。



おわりに

今回，このような機会をいただき，ありがとうございます。

エンタープライズ・アジャイルの領域において，開発プロジェクトへの適用事例や成功事例などは少なく，その取り組みが始まったばかりと言えます。

弊社の取り組みも，まだまだ発展途上ではありますが，お集まりの皆様におきまして，少しでもご参考になれば幸いです。

これからアジャイルに取り組んでみようと少しでもお考えいただければ幸いです。

ご清聴ありがとうございました。

chuden CTI