エンタープライズアジャイル勉強会

エンタープライズアジャイルの可能性と実現への提言

エンタープライズアジャイル勉強会

藤井　拓

アウトライン

- 日本におけるエンタープライズアジャイルの3つの可能性
- チームレベルのアジャイル開発
- チームレベルを越えてアジャイル開発を活用するための観点
  - 動機
  - 戦略
  - 戦術
  - 普及と育成
  - 日本固有の課題(?)と落とし穴
- 事例の概要のご紹介
  - 2015年7月から2016年5月の期間のご紹介事例
- まとめ

日本におけるエンタープライズアジャイルの3つの可能性

エンタープライズアジャイル勉強会とは

- 2014年5月：アドソル日進さん、コベルコシステムさん、オージスの3社社長が集まり、アジャイルに関する勉強会を開催したのが始まり

- 2014年6-8月：実行委員の方々の就任を要請し、2014年10月にエンタープライズアジャイル活用勉強会準備委員会として活動開始

- 2015年6月：エンタープライズアジャイル勉強会として正式に発足し、以降活動をしている

エンタープライズアジャイル勉強会のWeb: https://easg.smartcore.jp/

エンタープライズアジャイル勉強会の正式発足前の2014年度の活動

ご講演時期	ご講演タイトル	ご講演者
10月	アジャイル開発の現状と未来～エンタープライズアジャイルの可能性	永和システムマネジメント/チェンジビジョン 平鍋 健児 氏
11月	EA(エンタープライズアジャイル)にはEA(エンタープライズアーキテクチャー)が必要	アイ・ティ・イノベーション 中山 嘉之　氏
12月	エンタープライズ・アジャイル開発が果たすべき役割	シナジー研究所 依田智夫　氏
1月	部門アジャイル ～ 複数チームのコラボレーションをもっとよくするために	楽天株式会社 川口恭伸　氏
2月	エンタープライズシステムの継続的リフォームにおけるアジャイル開発	メソドロジック 山岸耕二　氏
3月	サービスデザイン思考のエンタープライズ・アジャイルにおける位置づけ	オージス総研 竹政昭利　氏

これらのご講演資料は勉強会のwebに掲載されています

エンタープライズアジャイル

- 大企業(企業のサイズ)、
- 「チーム」から「組織」へ、
- Agile Transformation（組織変革）
- ITを超えてアジャイルを採用、
- 管理職の役割は？
- 人事査定は？
- オフショアやアトソーシングは？

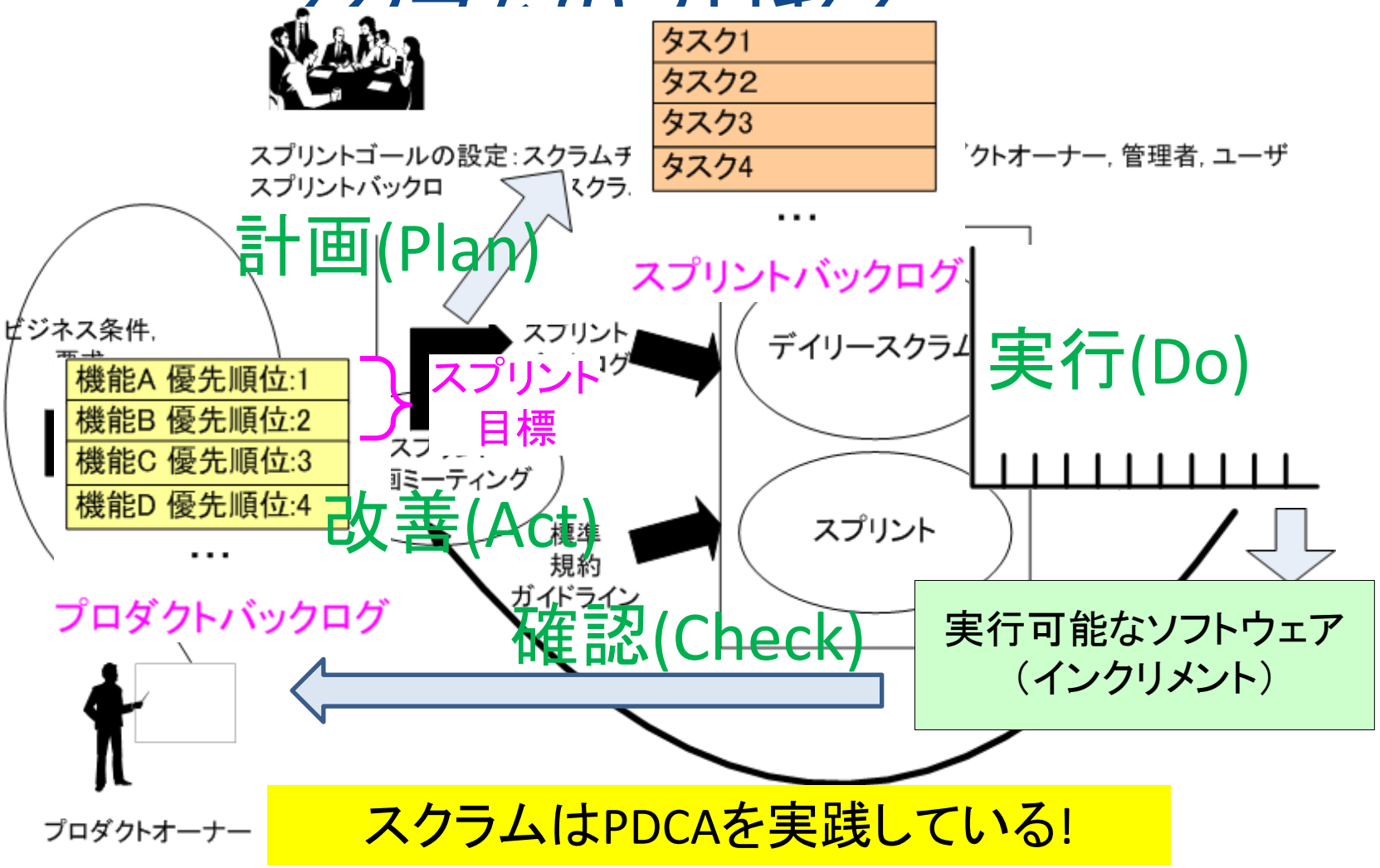
平鍋さんのスライドを引用させて頂きました

エンタープライズアジャイルの解釈

- 第1の立場
  - 業務の変化に対応し、基幹系やバックオフィス系のシステム(「エンタープライズシステム」)を効果的に開発するためにアジャイル開発を適用する
- 第2の立場
  - 競争力に資する製品やサービス、戦略的システムを開発するためにアジャイル開発や反復開発を大規模に適用する
- 第3の立場
  - 変化により良く対応できる、活力のある組織を実現するために企業や事業部をアジャイル化する

チームレベルのアジャイル開発

チームレベルのアジャイル
スクラムのプロセス



スクラムはPDCAを実践している!

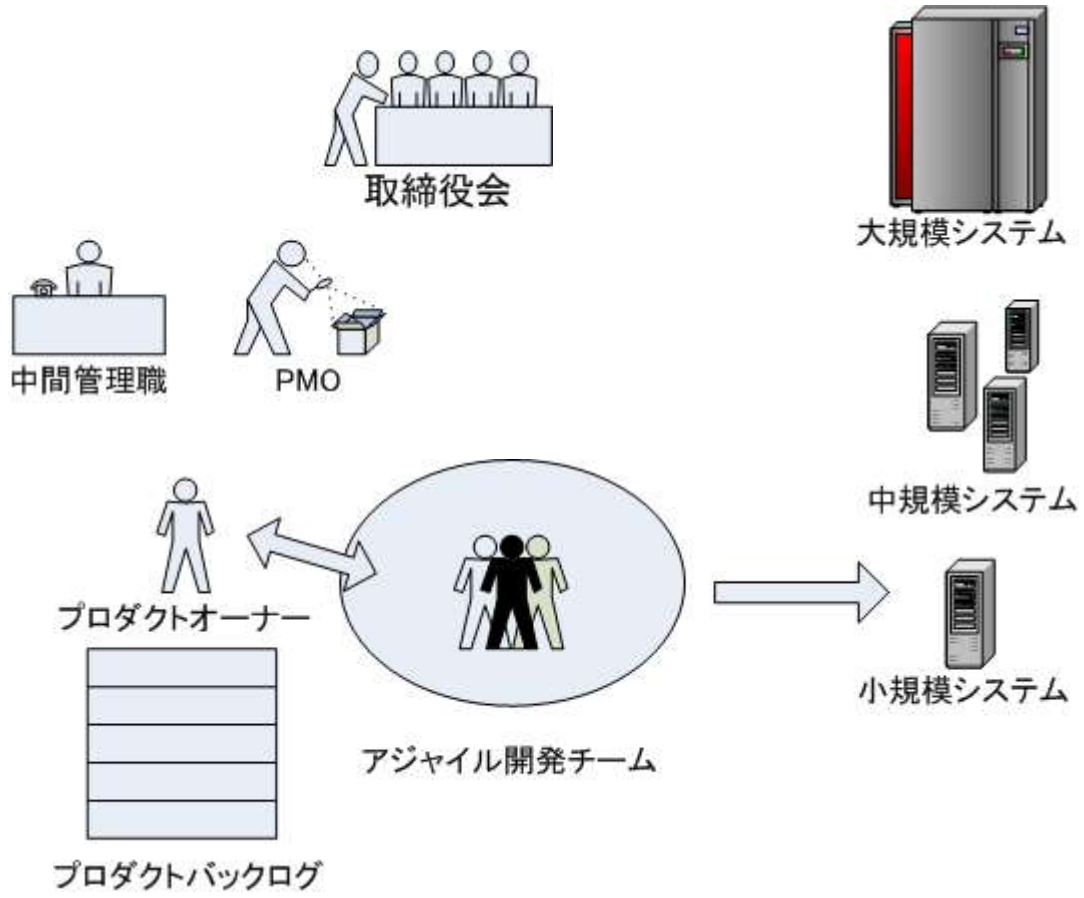
Schwaber, Ken. et al., *Agile Software Development with Scrum*, Prentice Hall, 2002の図をベースに作成

チームレベル
アジャイル開発の特徴

- 反復的な開発
  - 一定の期間毎に動くソフトウェアを作る形でソフトウェア開発を進める
- 顧客との協調
  - 開発期間を通じて顧客と連携し、顧客ニーズの変化やフィードバックに可能な限り対応する
- チームワーク重視（人間中心）
  - 開発者の自律性、コミュニケーション、改善を重視
- 技術的裏付け
  - 変化の影響を抑える技術プラクティス（設計/コード品質、自動化）の適用

後述するが、アジャイル開発ではなく、反復開発でよいとの意見もある

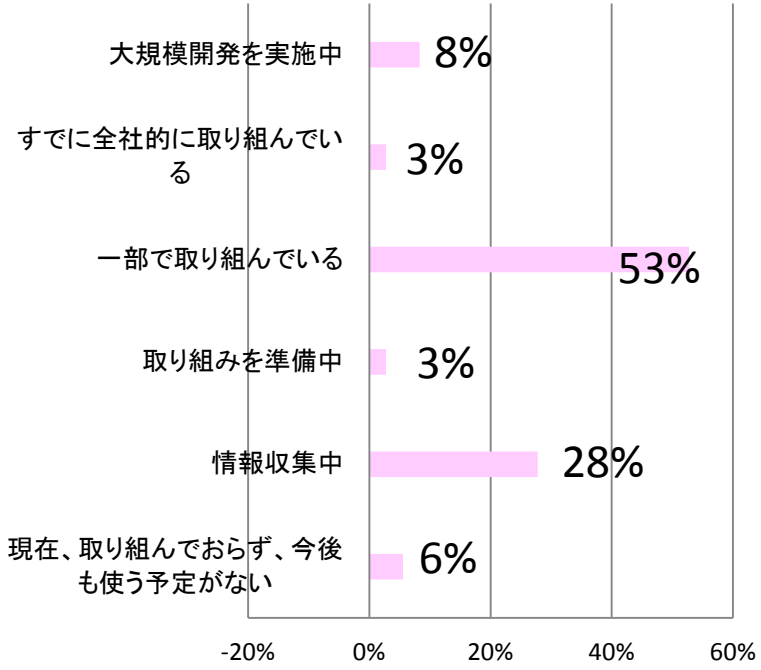
チームを超えたものの必要性



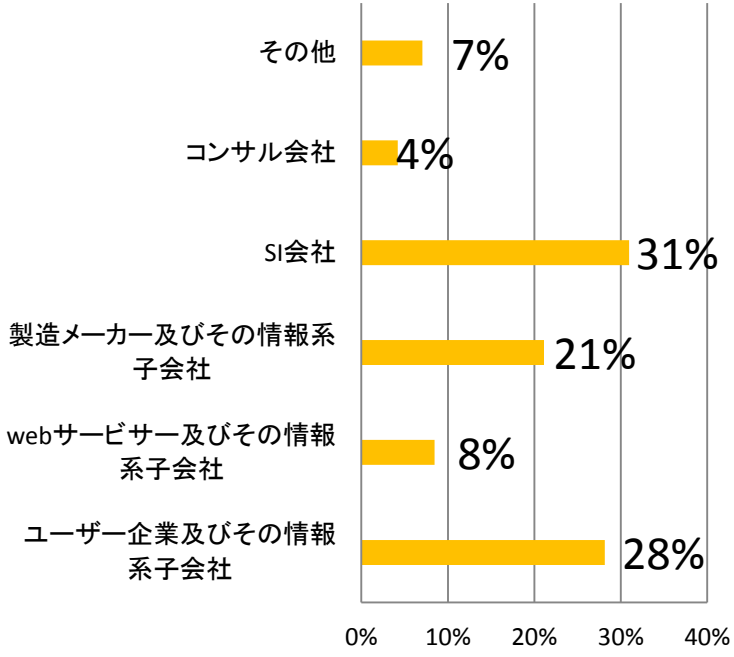
アジャイルは小規模開発にしか向かない？

日本での
アジャイル開発の取り組み状況と業種

取り組み状況



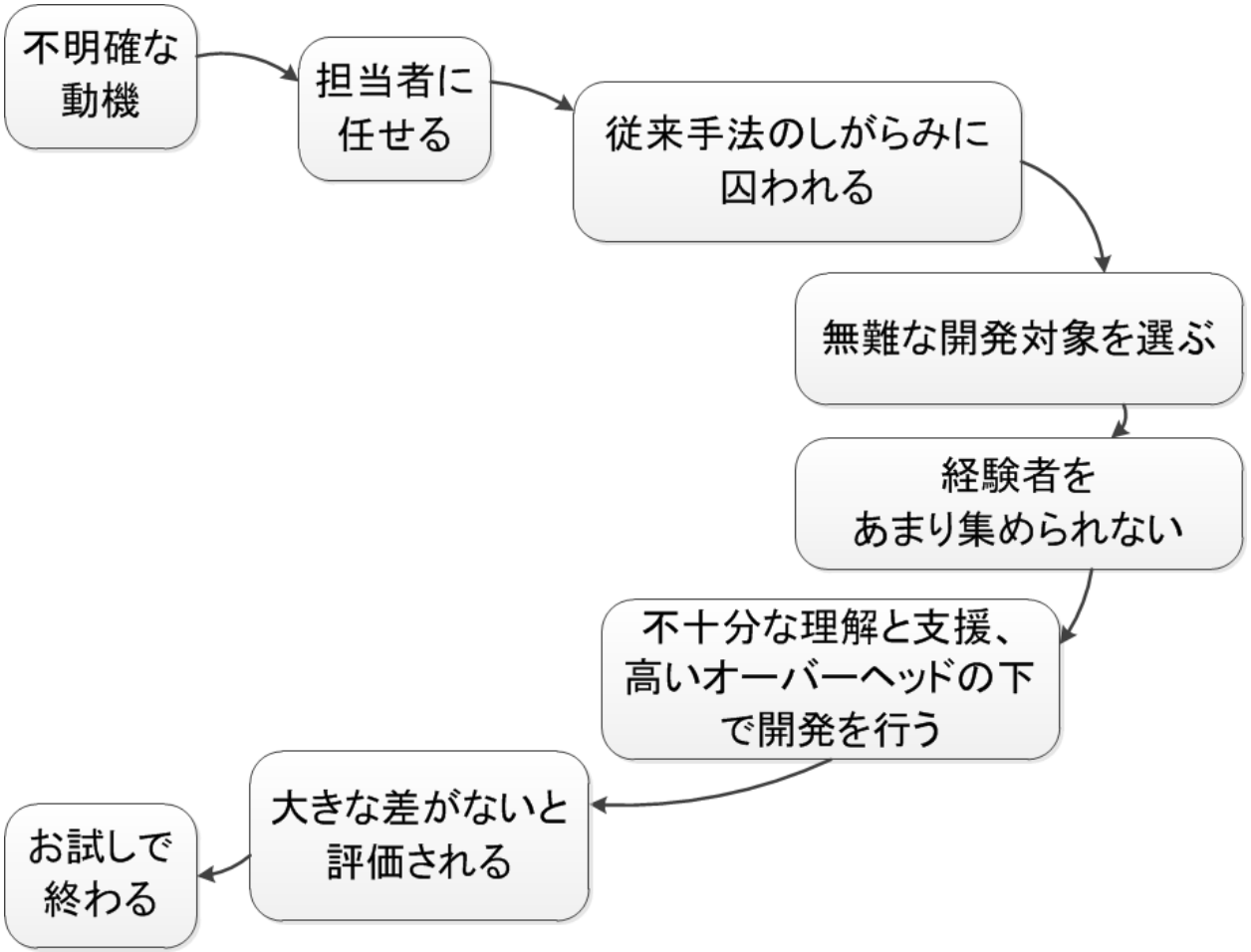
業種



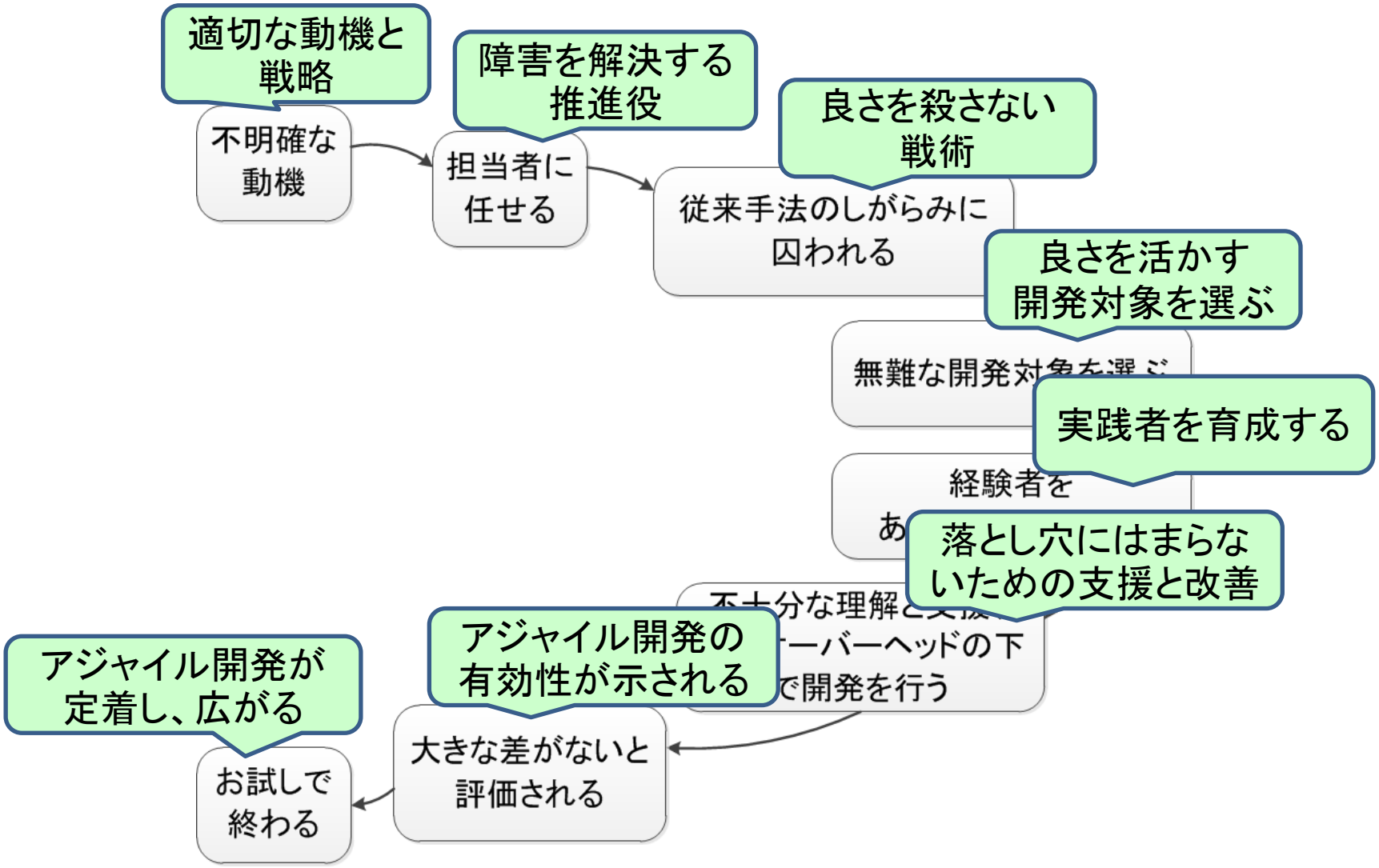
2016年3月に開催したエンタープライズアジャイルの集いで実施したアンケートの集計結果

チームレベルを越えてアジャイル開発を活用するための観点

うちでもアジャイル開発やってみました アンチパターン



うちでもアジャイル開発やってみました
アンチパターンの克服



チームを越えたアジャイルを考える視点

- 動機
  - なぜアジャイルか？
- 戦略
  - 勝つための手段としてアジャイルをどう使うか？
  - ビジネスの変化にシステムが追いつくためには？
- 戦術
  - 戦略を実行するためには？
- 普及と育成
  - 戦術を実行するためのメンバーをどう育成するか？
- 落とし穴
  - よく見かける失敗パターンは？

動機：チームを越えたアジャイル

- ビジネスの観点
  - ビジネス競争の激化に伴い、業務担当者とシステム担当者が一体となって、「有効」と思われる解決策（仮説）を「早く」業務に投入し、その結果により解決策の方向性の軌道修正を行うため

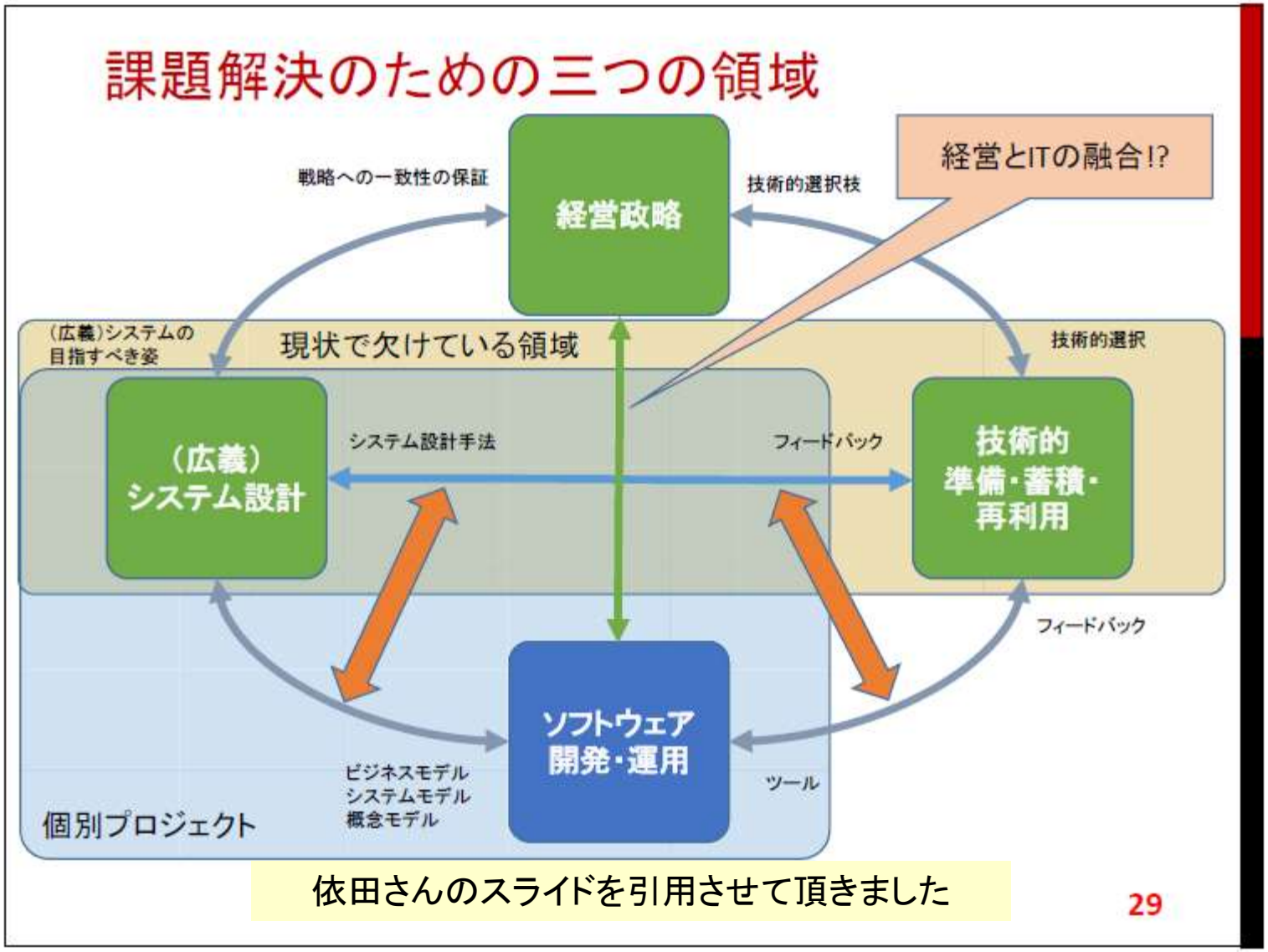
- 経営者の観点
  - 従業員が活き活きと仕事ができる環境を作りたい

「早く」は必ずしも「生産性の高さ」を意味するのではなく、「無駄を減らす」ことでも達成できる

戦略：チームを越えたアジャイル

- 経営戦略
  - 差別化するための戦略を考える
- 適用分野
  - アジャイル開発を適用すべき分野は？
- ITアーキテクチャー
  - アーキテクチャーをどのように考えるべきか？
- ROI
  - アジャイル（反復）の財務的なメリットは？

経営戦略



依田さんのスライドを引用させて頂きました

経営戦略

優れた戦略の五つの条件（競争戦略論）

1. 独自の**価値提案**
自ら選んだ顧客層に特徴ある価値を適切な価格で提供しているか？

2. 特別に調整された**バリューチェーン**
独自の価値提案を実現するのに最も適した一連の活動は、ライバル企業の行う活動と異なるのか？

3. 競合企業とは異なる**トレードオフ**
自社の価値を最も効率的、効果的に実現するために、やらないことをはっきり定めているか？

4. バリューチェーン全体における**適合性**
自社が行う活動は、お互いに価値を高めあっているだろうか？

5. 長期的な**継続性**
組織が得意なことに磨きをかけ、活動の調整、トレードオフ、適合性を促すことができる十分な安定性が、戦略の核にあるだろうか？

具体的なシステム像

選択を迫る方法論

システム工学に活躍の場がある

アジャイルとの親和性

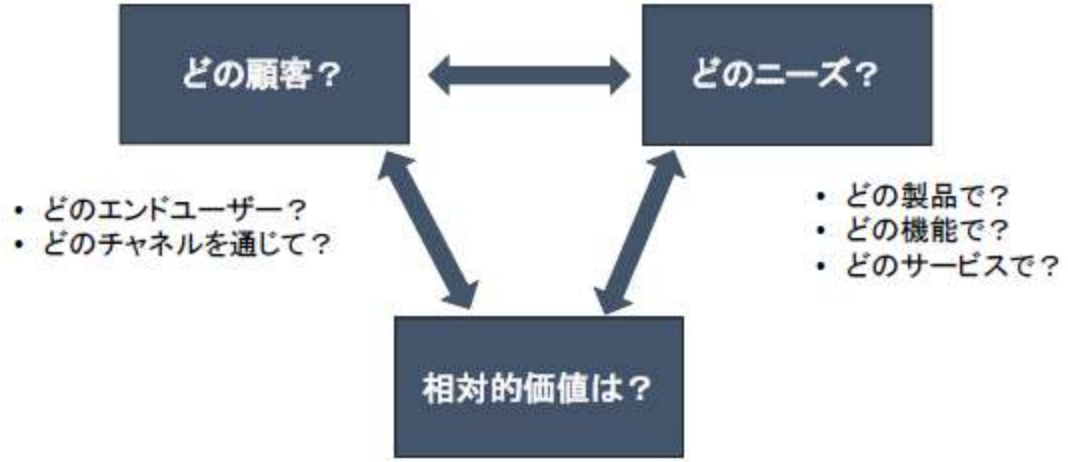
早川書房、ジョアン・マグレッタ、マイケル・ポーターの競争戦略、2012より転載

34

依田さんのスライドを引用させて頂きました

価値提案とは

価値提案は3つの質問に答える



早川書房、ジョアン・マグレッタ、マイケル・ポーターの競争戦略、2012より転載

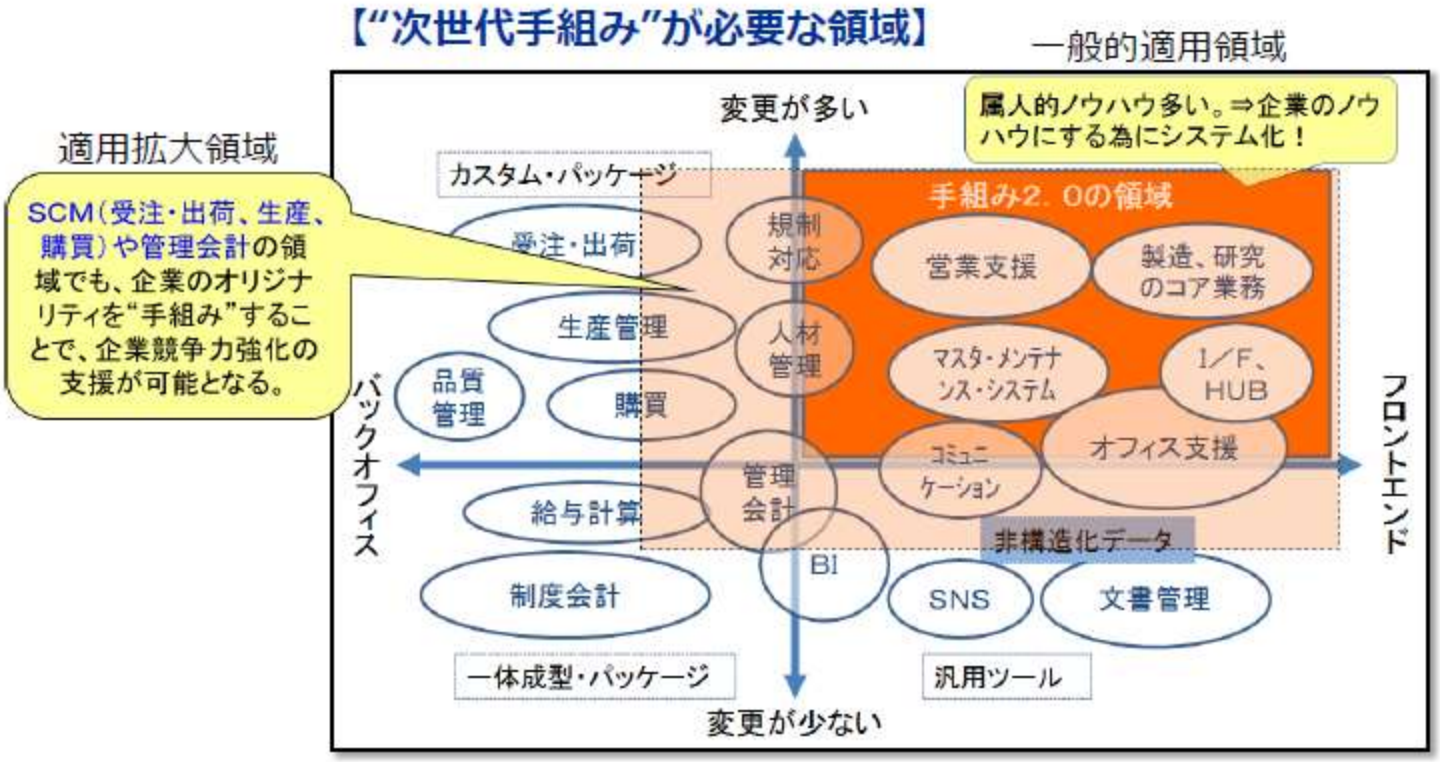
35

依田さんのスライドを引用させて頂きました

ポストＥＲＰとしての次世代手組み

■ １９９０年代後半～今日で大企業のＥＲＰ導入が一段落
- ＥＲＰは財務処理統制、業務処理の標準化などガバナンス、効率化で企業に価値をもたらしたが事業競争力の強化には貢献できていない

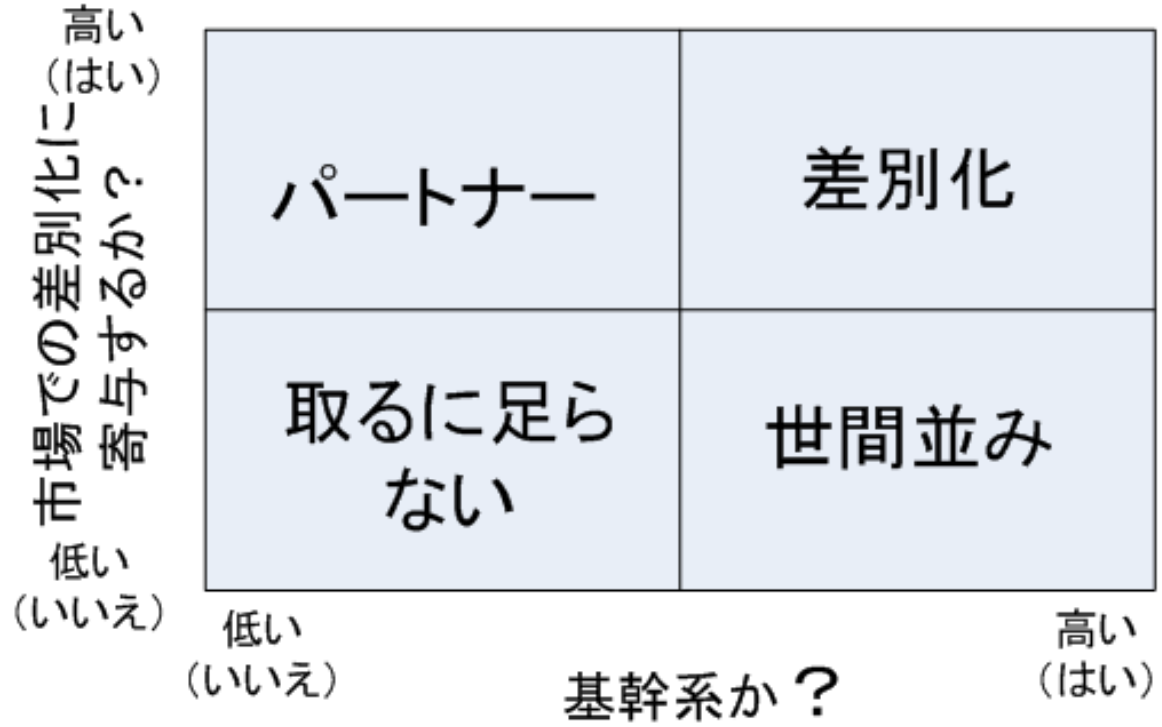
⇒企業の独自性が強い業務はパッケージでは難しく“手組み”が必要



中山さんのスライドを引用させて頂きました

ちょっと違うかもしれませんが、…

- 目的合わせモデル (Purpose Alignment Model)



出典：Pollyanna Pixton, Niel Nickolaisen, Todd Little and Kent McDonald, *Stand Back and Deliver: Accelerating Business Transformation*, Addison-Wesley, 2009

“次世代手組み”の前に過去の“過ち“を振返る

■ １９９０年代：

オープン化で各社はメインフレーム時代のデータモデルを捨てた

⇒物理実装は変われども、論理モデルは普遍であるハズが。。。

原因）日和見主義、集団行動、マスコミの煽り

■ ２０００年代：

ＥＲＰ導入で自社のモデル（データ、プロセス）を捨てた

⇒あたかも企業システムがコモディティであるかの如く錯覚した？

原因）ＣＩＯ不在、ブランド主義、業務音痴

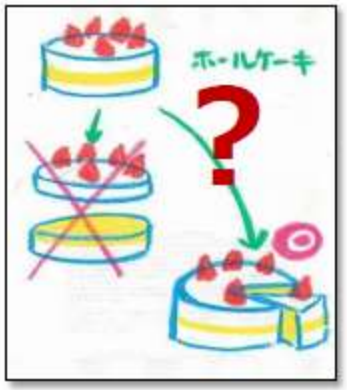
■ 今後の予見：

アジャイル開発で全社的アーキテクチャが不在に！

⇒ＤＢ一元化など全体最適の横串機能は必要なハズが。。。

原因）ベンダー丸投開発、情シス不在、ファッション

★結果、企業システムはスパゲッティとサイロの山を築きカオスと化す⇒“コスト、スピード、情報品質”の面で企業経営に悪影響を及ぼす可能性あり！

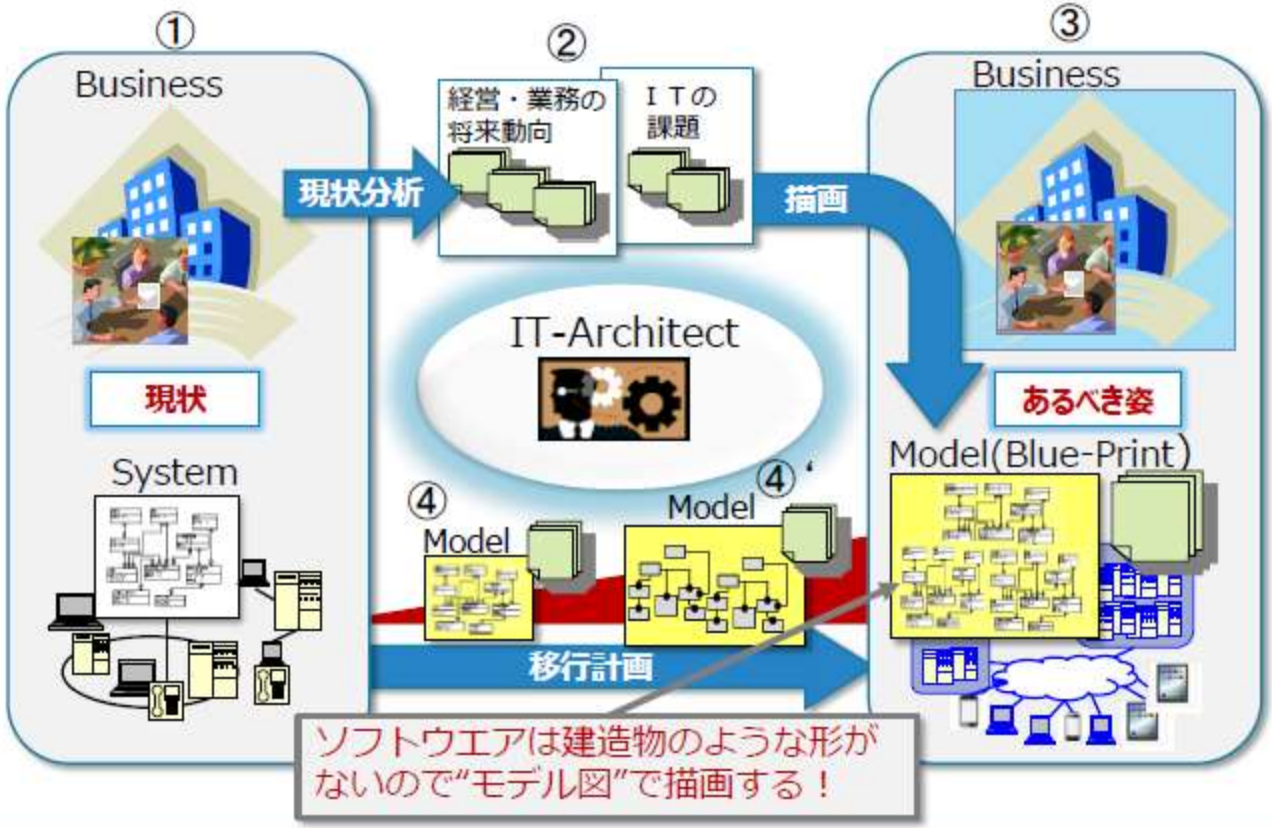


⇒**企業システムの構造（ｴﾝﾀｰﾌﾟﾗｲｽﾞｱｰｷﾃｸﾁｬ）を考える必要あり！**

中山さんのスライドを引用させて頂きました

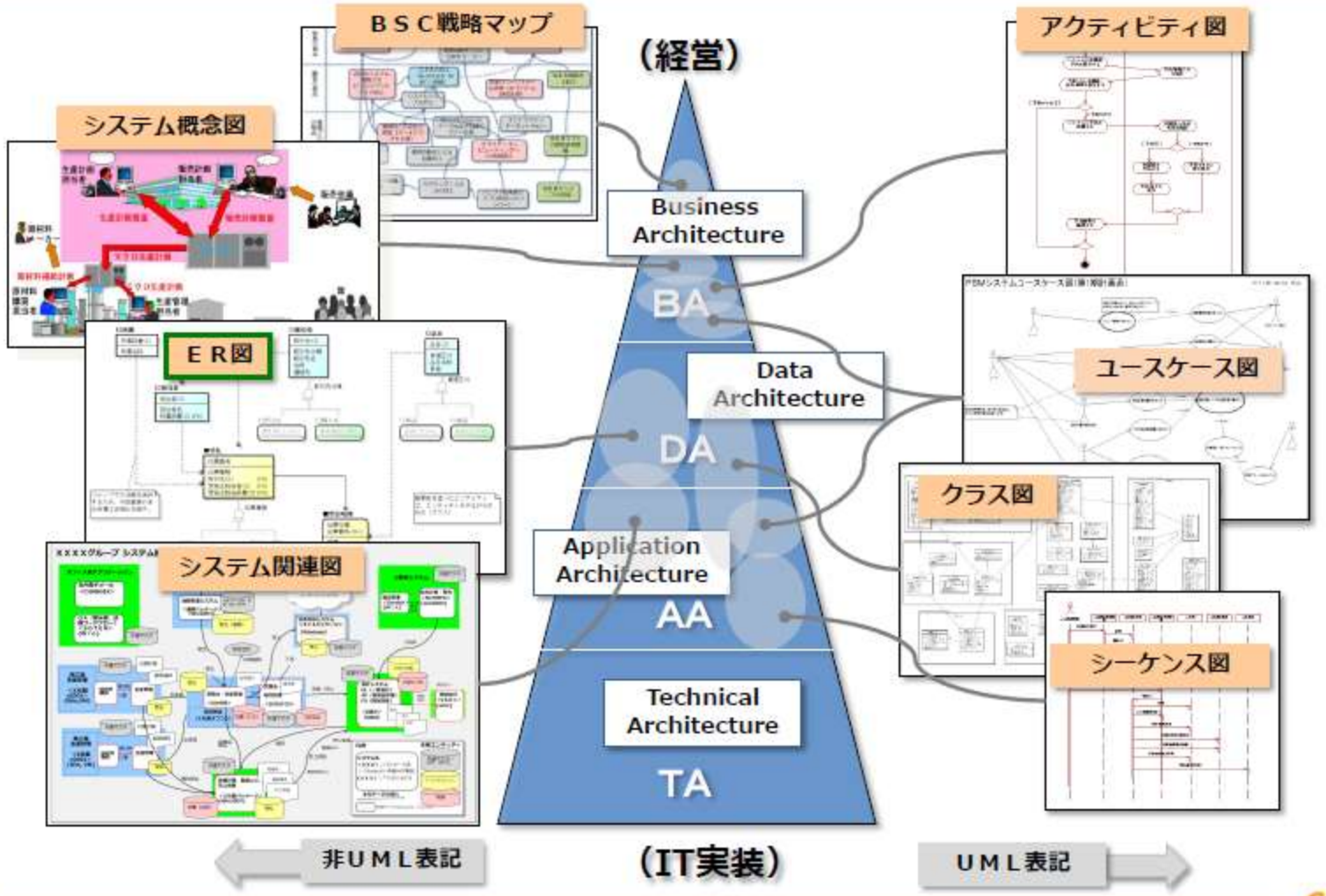
ＩＴアーキテクチャ設計とは？

・企業のケイパビリティ*向上のため、その企業に見合ったシステム構造（アーキテクチャ）の将来像を描き、それに向けた移行計画を立案すること。　※企業が得意とする組織的な能力のこと。スピード、高品質、効率性など。



中山さんのスライドを引用させて頂きました

ＥＡ（ｴﾝﾀｰﾌﾟﾗｲｽﾞｱｰｷﾃｸﾁｬ）を描く各種モデル図



15

中山さんのスライドを引用させて頂きました

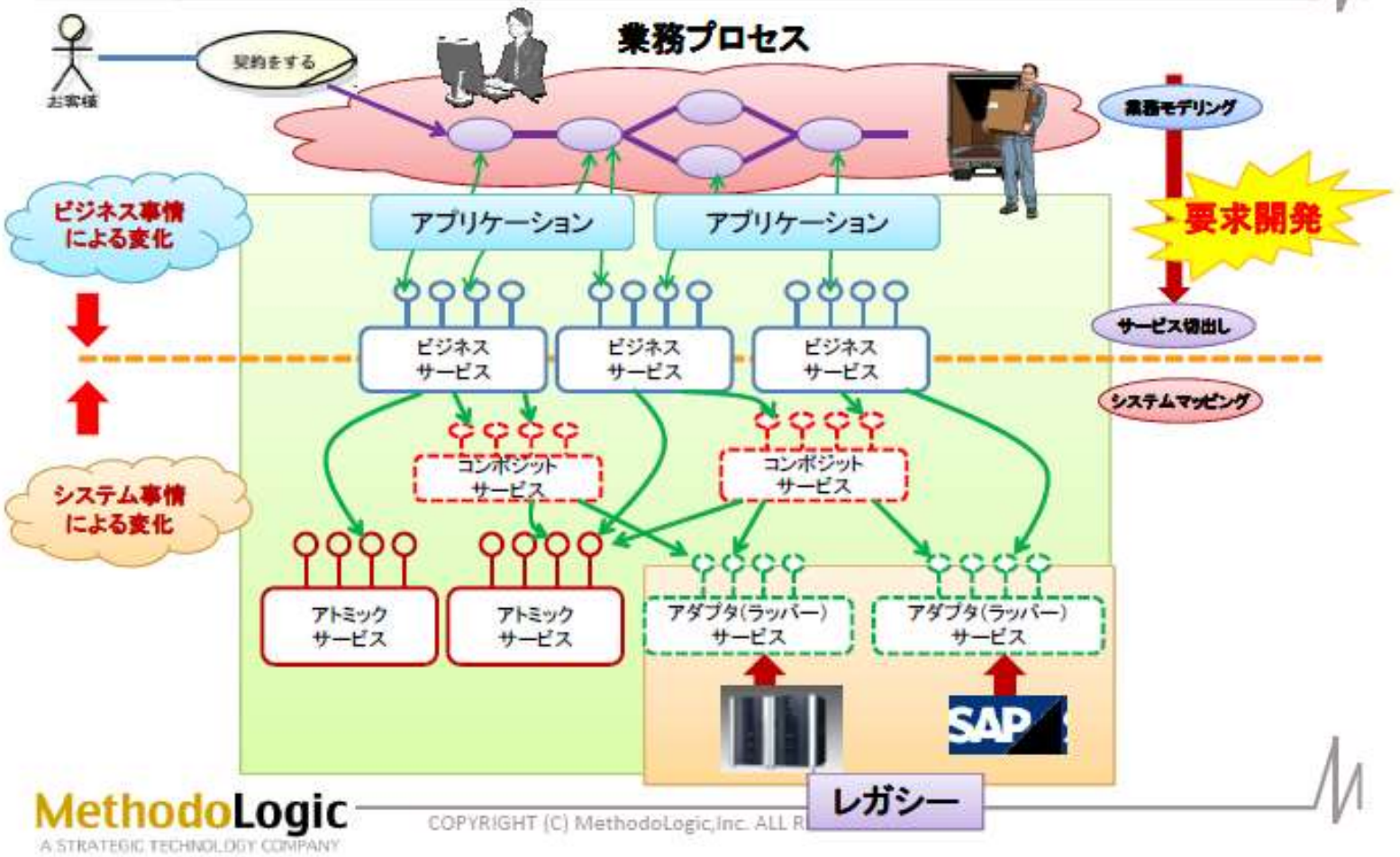
目指すべきTo-Beのシステムアーキテクチャ

- 業務とシステムの整合性確保
  - 業務モデルとシステムモデルのトレーサビリティが確保され、業務変化に合わせてシステムを追従させる仕組み
  - 業務的な単位のシステムサービスを組み合わせてシステムを構成する仕組み
- ビジネススピードへの適応する保守性の確保
  - 個々のシステム構成要素（コンポーネント）が適正粒度に刻まれていること。ユニット単位で保守。
  - 独立性が高く、変更時の影響範囲が限定されている
  - 役割が明確で変更箇所の特定が容易
  - コンポーネント間が疎結合かつ標準的インターフェースで接続されており、コンポーネントの抜き差しが自由
- 頻繁な変更、長期の利用を想定したアプリケーション統合基盤（枠組み）の確立
  - コンポーネントの入れ替え、メッセージ切り替えなどのオペレーションをサポート
  - 共通部分をユーティリティとして管理
  - ハードウエア、ネットワーク、OSなどインフラレイヤーを仮想化。下々の影響から脱却。

MethodoLogic
A STRATEGIC TECHNOLOGY COMPANY

山岸さんのスライドを引用させて頂きました

段階的かつ継続リフォームが可能な エンタープライズシステムアーキテクチャ



山岸さんのスライドを引用させて頂きました

継続的リフォームの推進に求められる取り組み

疎結合

- **エンタープライズとしての基盤の構築**
  - 全社の主要システムを連携し、サービス指向で疎結合に構成されたエンタープライズシステム基盤上へと移行させる
  - 業務の変更や技術の変更に対して安定なレイヤーを維持し、それぞれの変更を吸収し継続的な進化を可能にする

アジャイル

- **自由度の高い開発組織への移行**
  - 開発会社への一括発注では、リスクを載せた高い見積りとなり、細かな要求への対応も困難。事前の仕様フィックスが強いられタイムリーに適切なサービスを提供できない。設計情報やノウハウが開発会社に蓄積され、囲い込み状態で身動きがとれなくなる。システムが企業の中枢を担う状況で、丸投げのアウトソースはありえない。
  - 社内で安定したリソースを確保し、パートナー等を含めたイテレーション開発の体制を作る。アジャイルに要求に応えるモデルで、継続的に業務改善を行う。設計情報やノウハウが社内組織に蓄積される

要求開発

- **業務をITと結びつける手法の導入**
  - 業務を可視化し、上位目的に合致したシステム仕様をロジカルに導き出す手法により、必要となるサービスを適正な粒度で切り出す。これによって基盤上のサービスと業務の対応がとれ、業務の変更やM&A、アウトソーシング化などによるバリューチェーンの変更にも柔軟に対応できるようになる。

MethodoLogic
A STRATEGIC TECHNOLOGY COMPANY

山岸さんのスライドを引用させて頂きました

アーキテクチャの敗北

アーキテクチャからアジャイルへ

- 「象牙の塔のアーキテクト」という批判
  - » プロジェクトの柔軟性を確保するためには事後的な調整を重視することが重要だった
- もちろんアーキテクチャは重要
  - » アジャイルでも「必要なだけ決める」ことは大事
  - » マネジメントには、必ず技術的な土台が必要
  - » 優れたアジャイラーは優れたアーキテクト

鈴木（雄介）さんのスライドを引用させて頂きました

アーキテクチャの逆襲

柔軟なアーキテクチャの登場

- アーキテクチャが十分に柔軟であれば、マネジメント上の柔軟性に頼らないでも良くなる
- 近年でトレンドになりつつあるのがマイクロサービスアーキテクチャ（MSA）

鈴木（雄介）さんのスライドを引用させて頂きました

MSAとは

Microservices Architecture

- サービスによるコンポーネント化
- ビジネスケイパビリティに基づく組織化
- プロジェクトではなくプロダクト
- スマートなエンドポイントと単純なパイプ処理
- 分散ガバナンス
- 分散データマネジメント
- インフラの自動化
- フェイルを前提とした設計
- 進化的な設計

http://martinfowler.com/articles/microservices.html
“Microservices”

鈴木（雄介）さんのスライドを引用させて頂きました

アジャイル/反復開発のROI

- ROIに関する質問例
  - 1,000千ドルを投じて2年間でシステムの一部を開発するか？1,500千ドルを投じて3年間でシステム全部を開発するか？
    - より少ない機能を短期で作るか？より多くの機能を長期で作るか？
- アジャイル/反復開発のROI
  - 市場化できる最低限のフィーチャー(MMF)をベースに考える点が従来と異なる

MMFは中間リリースと考えても良い

ROIを考えるための要素

- 以下の要素で考える
  - 将来のお金の現在価値 (Present Value)
    - $PV = \$x / (1 + i/100)^n$
    - i : 利率、n : 年数
  - 割引きキャッシュフロー (Discounted Cash Flow)
    - DCF = PVで補正した期間毎のキャッシュ持ち高
  - 正味現在価値 (Net Present Value)
    - $NPV = \Sigma DCF$
  - 内部収益率 (Internal Rate of Return)
    - プロジェクトのNPVが0になる利率

従来手法のROI分析の例

説明			期間 1	2	3			合計
収入								
	売り上げ					1,000	3,800	4,800
	削減					200	600	800
総収入			0	0	0	1,200	4,400	5,600
支出								
	資本							
		ハードウェア	700	20	20	200	20	960
		ソフトウェア	300					300
	資本合計		1,000	20	20	200	20	1,260
	実行							
		開発	240	360	550	360	112	1,622
		データセンター	30	30	30	30	30	150
		サポート	140	120	120	150	150	680
		マーケティング	0	0	100	200	300	600
	実行合計		410	510	800	740	592	3,052
総支出			1,410	530	820	940	612	4,312
キャッシュ			−1,410	−530	−820	260	3,788	1,288
投資			−1,410	−530	−820			−2,760
ROI								47%

4年目から売り上げが発生！

出典：Mark Denne, Jane Cleland-Huang, Software by Numbers: Low-Risk, High-Return Development , Prentice Hall, 2003

従来手法のNPV分析の例

5年目で累積のNPVが0になる利率を求めるこれが内部収益率(IRR)になる

期間		1	2	3	4	5	
キャッシュ		−1,410	−530	−820	260	3,788	
自分で資金供給できる状況					X		
							NPV
DCF @	5.0%	−1,343	−481	−708	214	2,968	650
	累積のNPV	−1,343	−1,824	−2,532	−2,318	650	
	収支均衡状況					X	
	10.0%	−1,282	−438	−616	178	2,352	194
	累積のNPV	−1,282	−1,720	−2,336	−2,158	194	
	収支均衡状況					X	
	12.8%	−1,250	−417	−571	161	2,074	−3
	累積のNPV	−1,250	−1,667	−2,238	−2,077	−3	

従来手法の内部収益率(IRR)は12.8%!

出典：Mark Denne, Jane Cleland-Huang, Software by Numbers: Low-Risk, High-Return Development , Prentice Hall, 2003

MMFを用いた場合のROI分析の例

2年目から売り上げが発生！

説明			1			4	5	合計
収入								
	売り上げ			700	1,400	2,100	2,800	7,000
	削減					200	600	800
総収入			0	700	1,400	2,300	3,400	7,800
支出								
	資本							
		ハードウェア	700	20	20	200	20	960
		ソフトウェア	300					300
	資本合計		1,000	20	20	200	20	1,260
	実行							
		開発	240	360	550	360	112	1,622
		データセンター	30	30	30	30	30	150
		サポート	140	120	120	150	150	680
		マーケティング	200	200	200	200	200	1,000
	実行合計		610	710	900	740	492	3,452
総支出			1,610	730	920	940	512	4,712
キャッシュ			−1,610	−30	480	1,360	2,888	3,088
投資			−1,610	−30				−1,640
ROI								188%

出典：Mark Denne, Jane Cleland-Huang, Software by Numbers: Low-Risk, High-Return Development , Prentice Hall, 2003

MMFを用いた場合のNPV分析の例

5年目で累積のNPVが0になる利率を求めるこれが内部収益率(IRR)になる

期間		1	2	3	4	5	
キャッシュ		−1,610	−30	480	1,360	2,888	
自分で資金供給できる状況				X			
							NPV
DCF @	5.0%	−1,533	−27	415	1,119	2,263	2,236
	累積のNPV	−1,533	−1,561	−1,146	−27	2,236	
	収支均衡状況					X	
	10.0%	−1,464	−25	361	929	1,793	1,594
	累積のNPV	−1,464	−1,488	−1,128	−199	1,594	
	収支均衡状況					X	
	36.3%	−1,181	−16	190	394	614	0
	累積のNPV	−1,181	−1,197	−1,008	−614	0	

MMFを用いた場合の内部収益率(IRR)は36.3%!

出典：Mark Denne, Jane Cleland-Huang, Software by Numbers: Low-Risk, High-Return Development , Prentice Hall, 2003

Software By Numbersの目次

- 第２章：新たなROI
- 第３章：市場可能なフィーチャーの識別と評価
- 第４章：インクリメンタルなアーキテクチャー
- 第５章：IFMの順序付け戦略
- 第6章：MMFの分類と並行開発
- 第7章：具体的ではないもの管理
- 第8章：IFMと統一プロセス
- 第9章：IFMとアジャイル開発
- 第10章：情報に基づく意思決定
- 第11章：事例：IFMの実践

IMF：Incremental Funding Methodology
内部収益率(IRR)を適宜見直しながら投資すべきプロダクトを決める
「動的なポートフォリオ管理」という考え方もある

アジャイル/反復開発のROIのポイント

- 大きなものを一気に作る方法と、最低限使えそうなもの(MMF)をリリースし、それをインクリメンタルに発展させる方法との違いを財務的に理解するために有効

- このROIのモデルは当初の仮説にしかすぎず、このROIモデルの精度を上げるよりはMMFを実際に作って仮説を検証することの方が大事

戦略：事例との関係

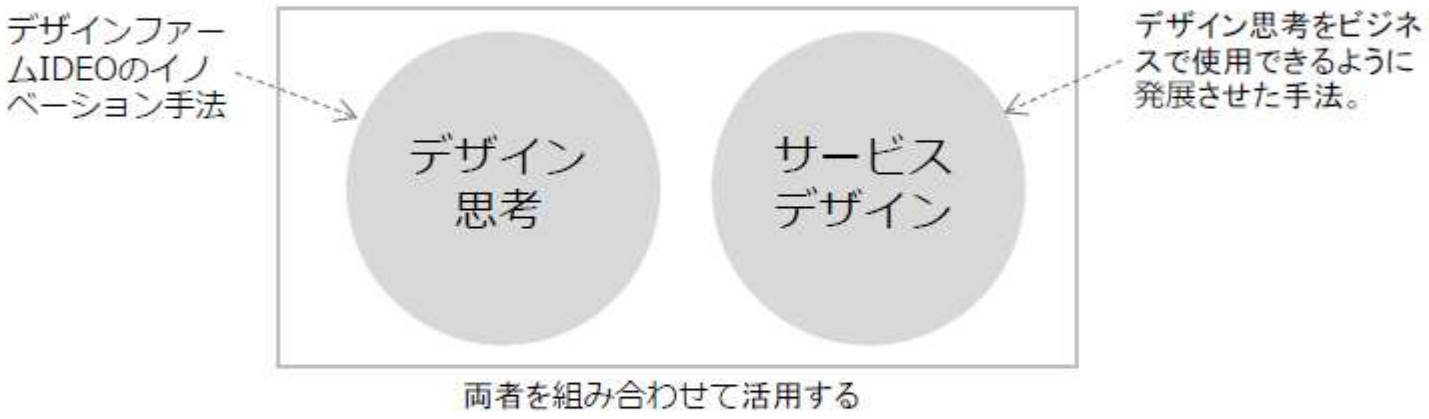
- 2015年7月：東京海上日動システムズ様
  - 保険契約のWeb化
- 2015年8月、2016年3月：KDDI様
  - クラウドビジネス
- 2015年12月：ゼンアーキテクト（カプラン様）様
  - 人財サービス業のamazonを目指す
- 2016年4月：ウルシステムズ（製造業）様
  - SCMシステム
- 2016年5月：ニッセイ情報テクノロジー様
  - ノウハウの商業化

戦術：チームを越えたアジャイル

- 業務、サービス、プロダクトの構想
  - 競争力のあるものをどのように発想するか？
- 要求開発
  - 発想を要求にどのように落とし込むか？
- 開発手法
  - 要求を実現するための開発手法は？
- 仮説の検証
  - 仮説の妥当性をどのように確認するか？

サービスデザイン思考とは？

サービスデザイン思考とは、「顧客の視点から問題点を見つけ出し、顧客にとって好ましく、価値があるアイディアを発想し検証し、提供者の視点から、全体のサービスをデザインし、ビジネスとして提供すること。」



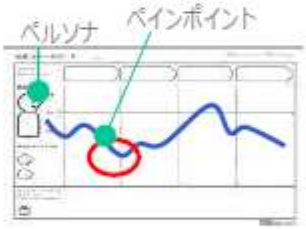
竹政さんのスライドを引用させて頂きました

サービスデザイン思考

単なるインタビューではなく、顧客の自覚していないことを明らかにする。

DISCOVER

共感することで、顧客自身が自覚していない、インサイト（洞察）を**デプスインタビュー**や**（行動観察）**から、明らかにする。
Sympathy(感情共鳴、一緒に）でなくEmpathize（心の中に入り込み、自分のものとして感じる。）



顧客のインサイト（洞察）を捉えているか素早く確かめる。

DEVELOP

Ideateで考えたアイデアの**プロトタイプ**を作成することで、顧客に実際のイメージを持ってもらいます。
プロトタイプで作成したものを**アクティングアウト（寸劇）**を実施、顧客に体験してもらい、顧客のニーズに合致しているかを確かめます。

DELIVER

アクティングアウト（寸劇）で確かめたアイディアを仕事やビジネスに適応できるかどうかの検討を**ビジネスモデルキャンバスやピクト図**を使って検証します。

単に同じメンバーでブレインストーミングをしても、新しいアイディアは生まれない。

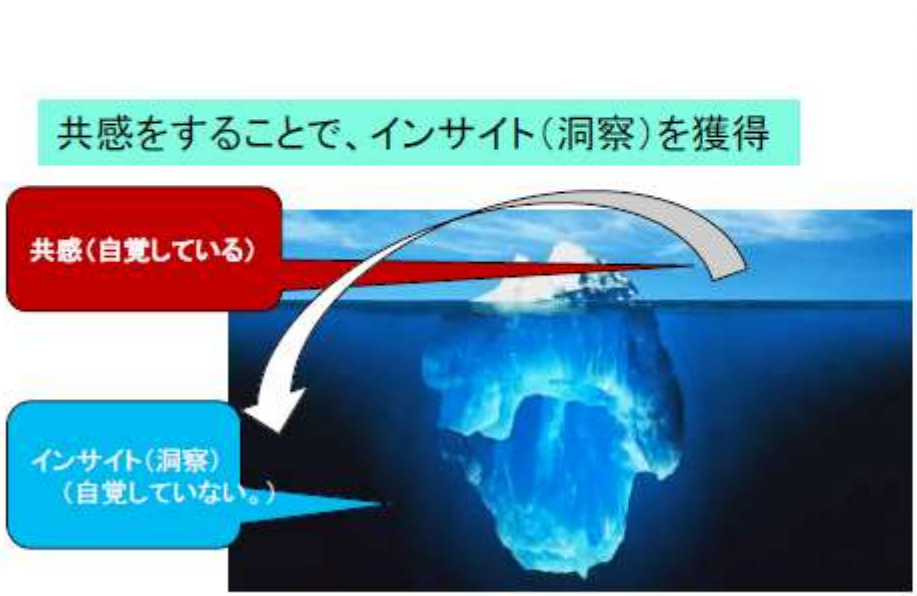
DEFINE

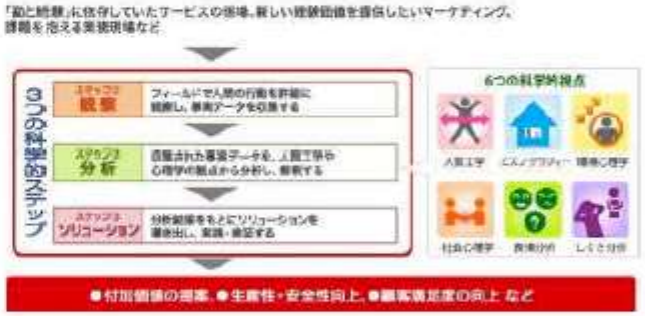
ブレインストーミングで課題解決のためのアイデアを出す。多様性が重要である。多様性が確保できない場合は、バイアスを意識することで、これまでの傾向とは異なるアイディアを**強制発想**する。

竹政さんのスライドを引用させて頂きました

探求(DISCOVER)

- 共感することで、顧客自身が自覚していない、インサイト（洞察）をデプスインタビューや行動観察などから、明らかにします。
- Sympathy(感情共鳴、一緒に）でなくEmpathize（心の中に入り込み、自分のものとして感じる。）





出典：http://www.kansatsu.jp/observation/observation.html

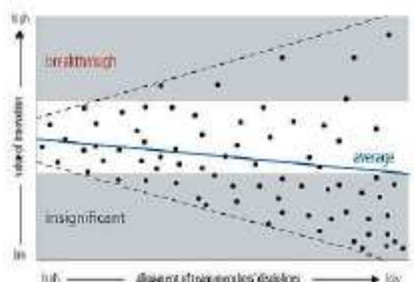
竹政さんのスライドを引用させて頂きました

設計(DEFINE)

- ブレインストーミングなどで課題解決のためのアイデアを出します。多様性が重要です。多様性が確保できない場合は、バイアスを意識し、破壊することで、これまでの傾向とは異なるアイディアを強制発想をします。

Going for Breakthrough

My research suggests that when a creative team is made up of people from very similar disciplines, the average value of its innovations will be high, but it will be unlikely to achieve a breakthrough. On the other hand, a group of people from very diverse disciplines is more likely to achieve breakthroughs—but will also produce many more low-value innovations.



多様性の確保

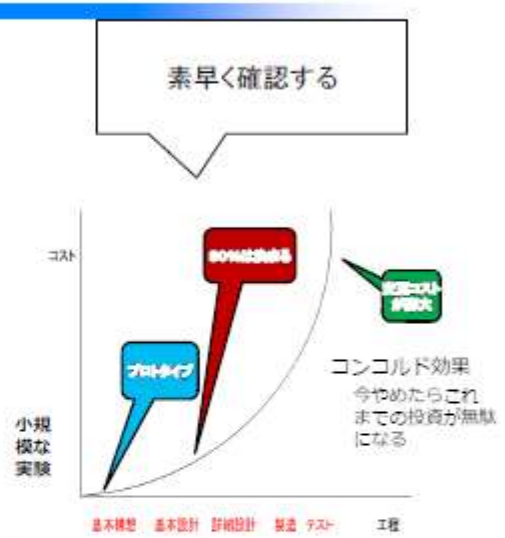
バイアスの意識と破壊

出典: http://hbr.org/2004/09/perfecting-cross-pollination/ar/1
出典: http://enterprisezine.jp/bizgene/detail/5005

竹政さんのスライドを引用させて頂きました

再構成(DEVELOP)

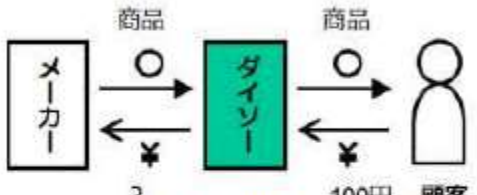
- DEFINEで考えたアイデアのプロトタイプを作成することで、顧客に実際のイメージを持ってもらいます。
- プロトタイプを使用して実際の利用場面をアクティングアウト（寸劇）で表現します。



竹政さんのスライドを引用させて頂きました

実施(DELIVERY)

- 提供者側の視点で、検討してきたアイディアがビジネスとして成立するかを考えます。
- 商品化して運用します。



ピクト図解

ビジネスモデルキャンバス

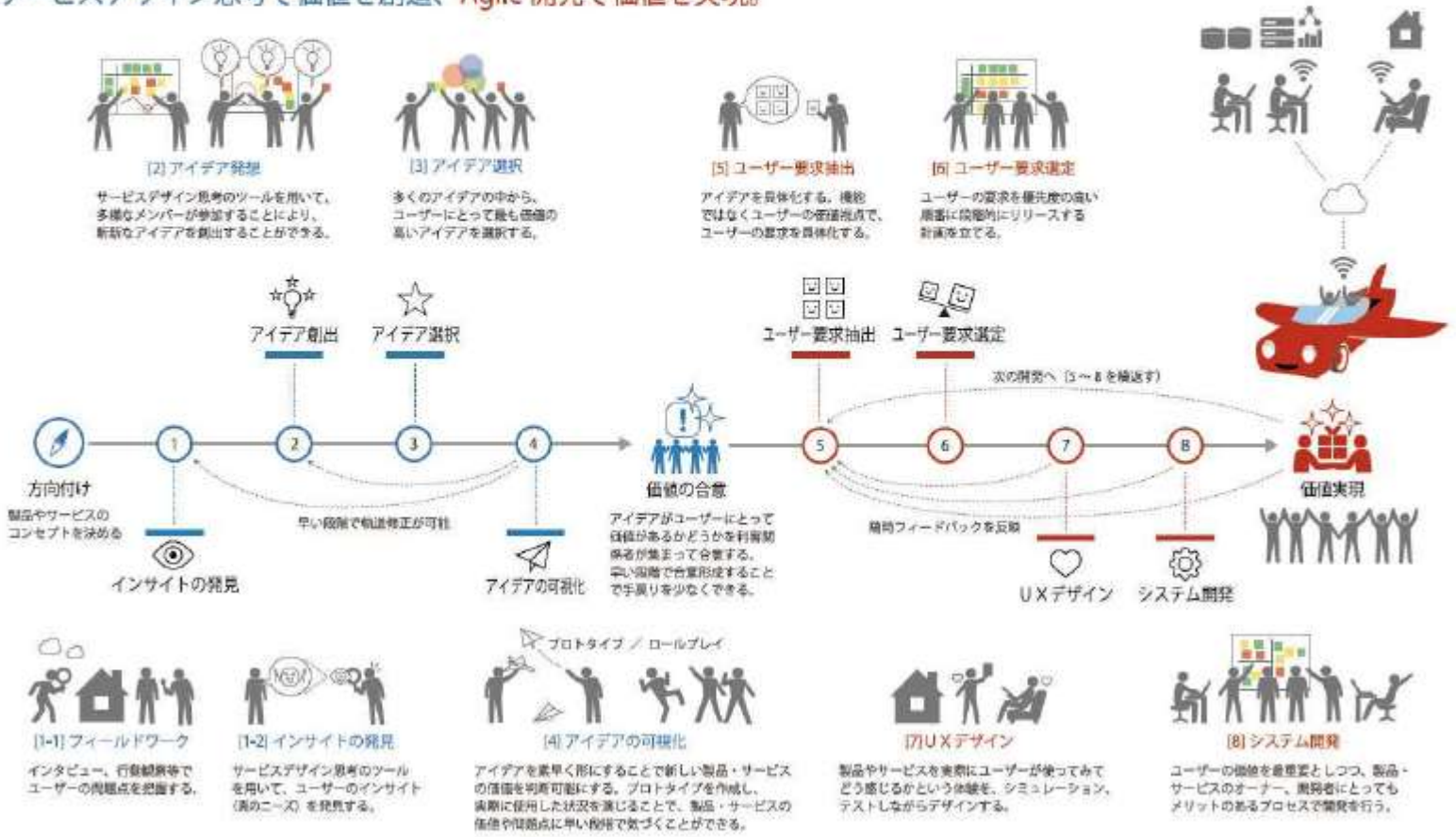

サービスブループリント

OGIS-RI

竹政さんのスライドを引用させて頂きました

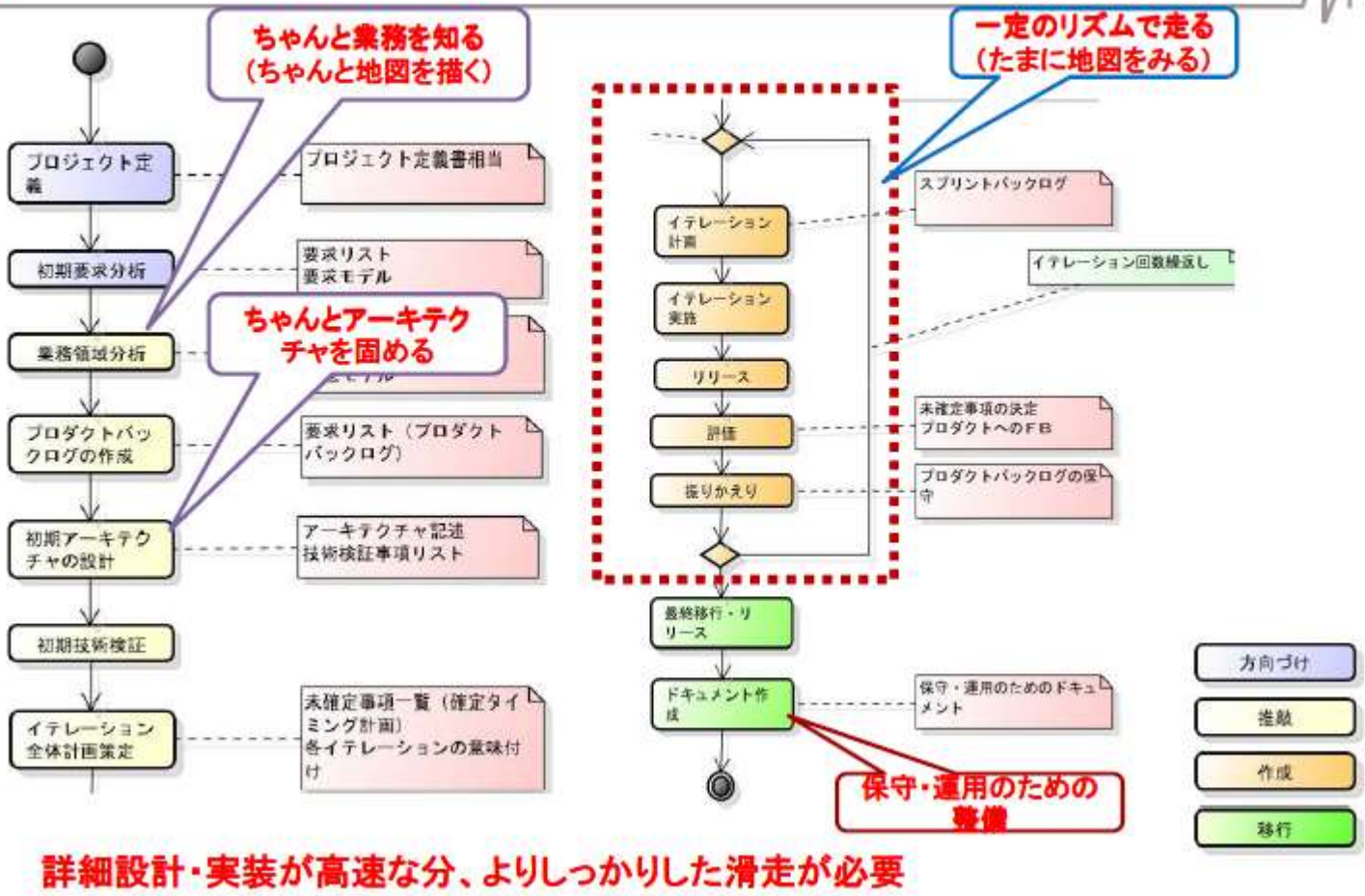
サービスデザイン思考とアジャイル

- 真のユーザーニーズを探り、価値を共創し、価値をタイムリーに提供します
  サービスデザイン思考で価値を創造、Agile 開発で価値を実現。



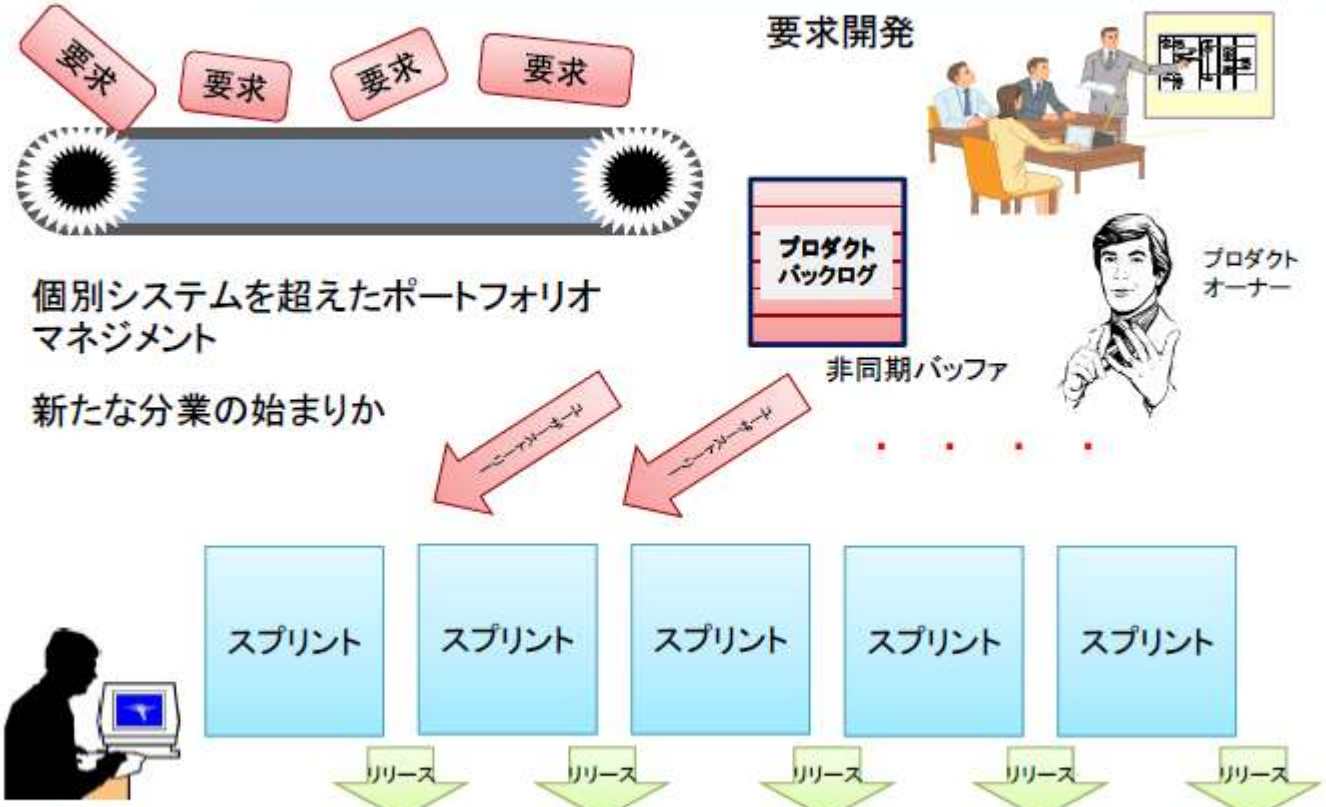
竹政さんのスライドを引用させて頂きました

現実的なエンタープライズアジャイル



山岸さんのスライドを引用させて頂きました

要求開発とアジャイル開発の究極コラボ



MethodoLogic
A STRATEGIC TECHNOLOGY COMPANY

28

山岸さんのスライドを引用させて頂きました

エンタープライズアジャイルでのお薦めモデリング

- スプリント以前
  - 要求モデル
  - 業務をとらえる3種のモデル
    - サービスモデル、概念モデル、業務フロー
  - アーキテクチャモデリング
    - 主要なパターン（設計クラス図とシーケンス図）とサンプルコーディング
  - ユースケース一覧
    - プロダクトバックログへの展開
- スプリント中
  - 詳細設計のモデルは省略する
    - 設計者と実装者が同じ
    - 詳細レベルはコードの方が表現できる
  - コミュニケーションのためのメモとしてモデルを多用する
    - 使い捨て前提
- リリース後、運用準備
  - ポリシーによる（紙の形式か情報として残ればいいか）
  - 主要クラスと主要動作のシーケンス図、特殊なアルゴリズム
  - ユーザーストーリーの集約
  - ビジネスルールの整理だけは怠りなく

MethodoLogic
A STRATEGIC TECHNOLOGY COMPANY

山岸さんのスライドを引用させて頂きました

2大 Enterprise Agile Framework

SAFe=Scaled Agile Framework

- Dean Leffingwell

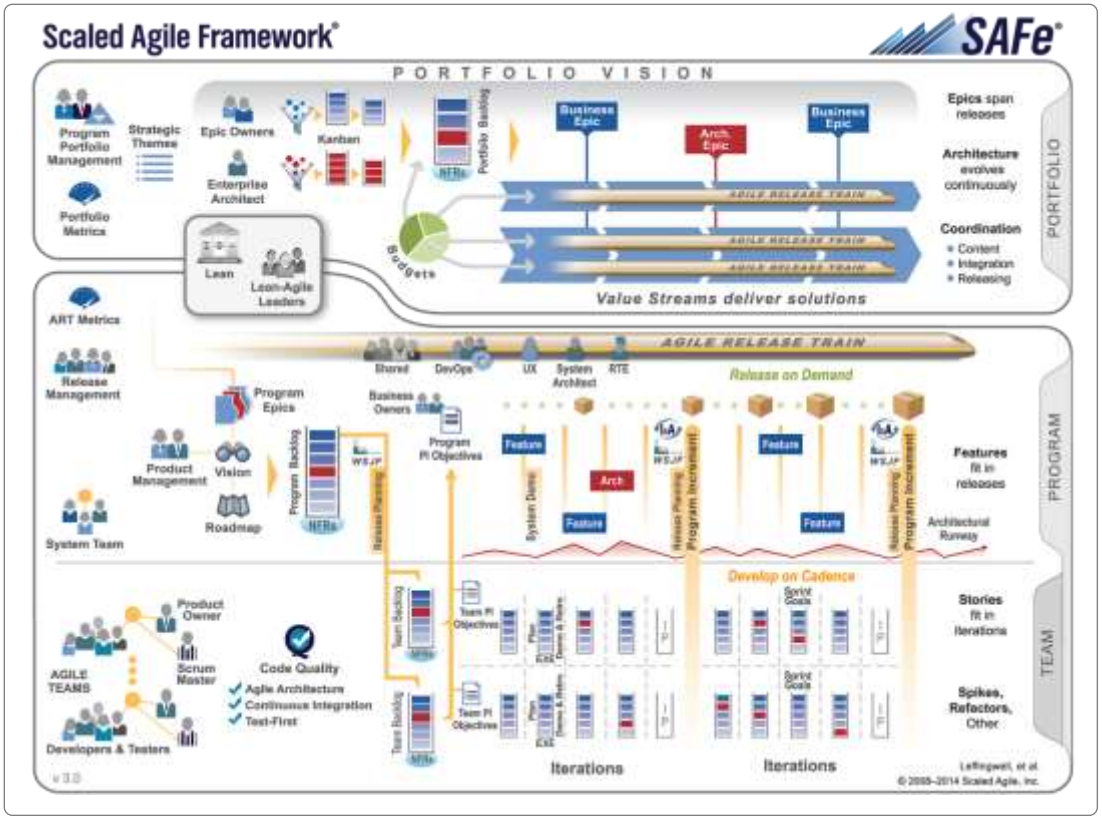
DAD = Disciplined Agile Delivery

- Scott W. Ambler

平鍋さんのスライドを引用させて頂きました

SAFe (Scaled Agile Framework)とは

Scaled Agile Frameworkは、企業規模でリーンとアジャイルのプラクティスを適用するための実証され、公開されたフレークワーク



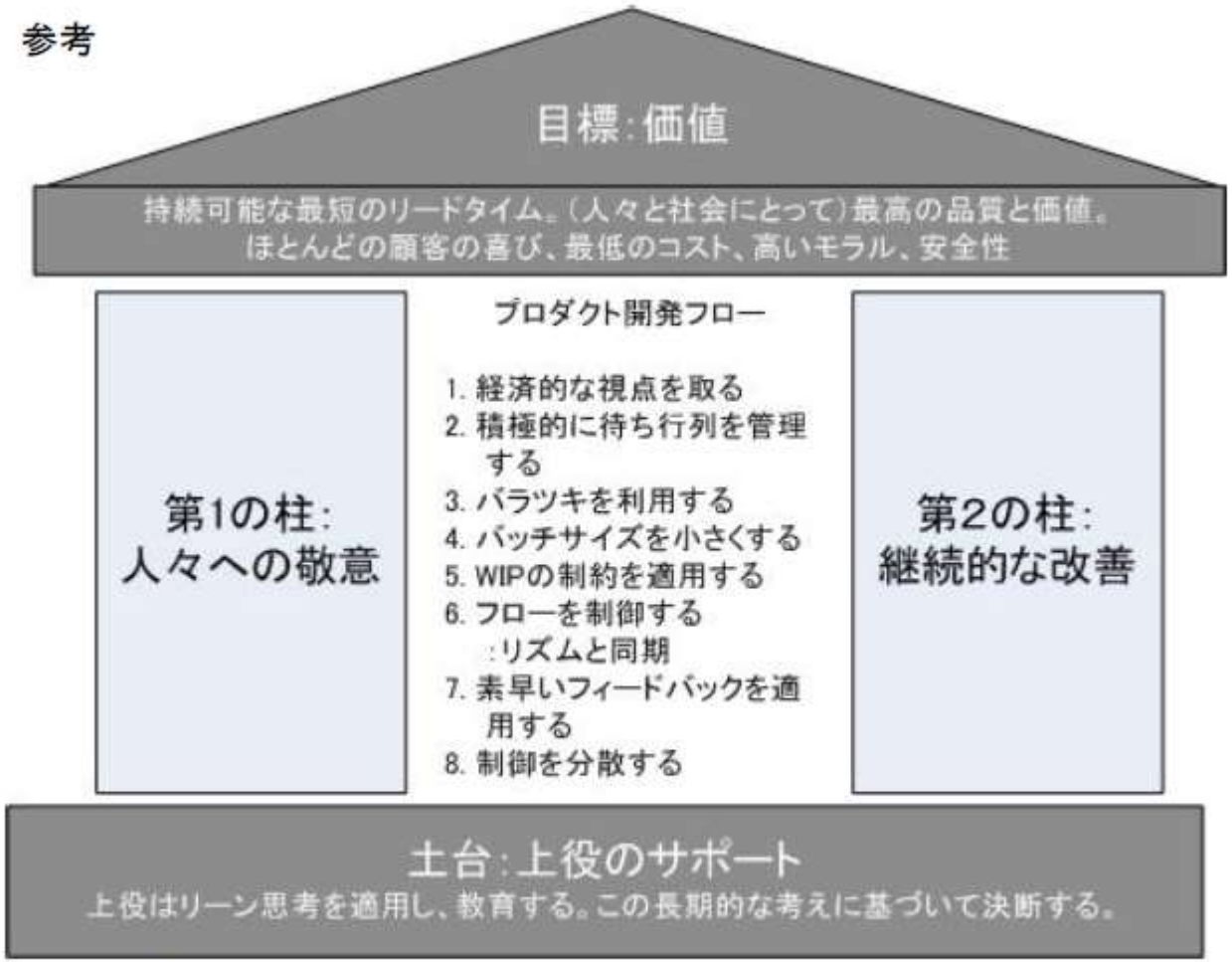
- 方向性、連携、納品を揃える
- 書籍で詳しく説明し、webでも説明を提供している
- 多数の実践者やチームへのスケールアップに成功している

中心的な価値:

1. ベクトル合わせ
2. コード品質
3. プログラム（＝大規模プロジェクト）の実行
4. 透明性

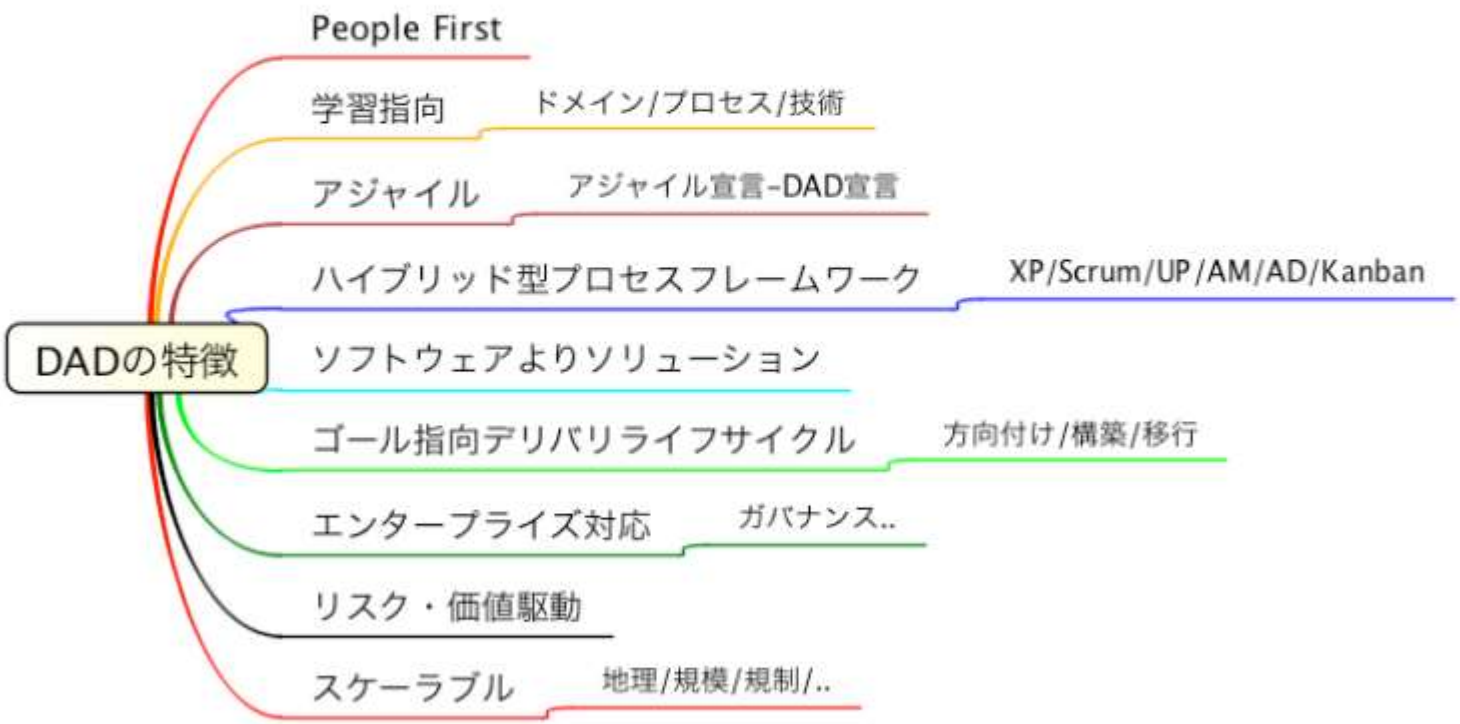
SAFe英語サイト： http://ScaledAgileFramework.com
SAFe日本語サイト： http://ScaledAgileFramework.com/jp

参考



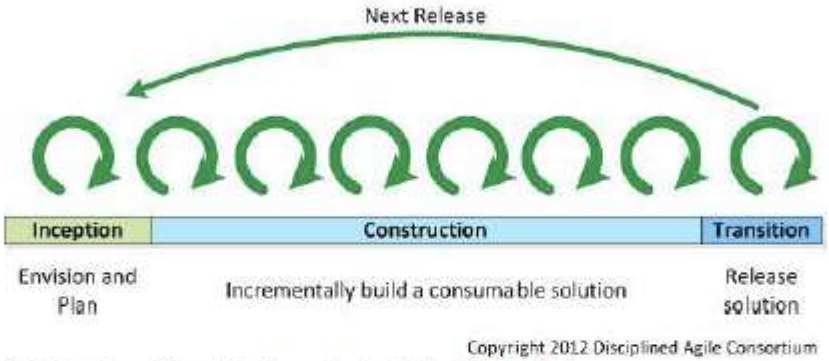
出典：SAFe入門 https://www.ogis-ri.co.jp/otc/hiroba/technical/IntroSAFe/IntroSAFePart2May2014.html

DADの特徴



平鍋さんのスライドを引用させて頂きました

DADにはフェーズがある

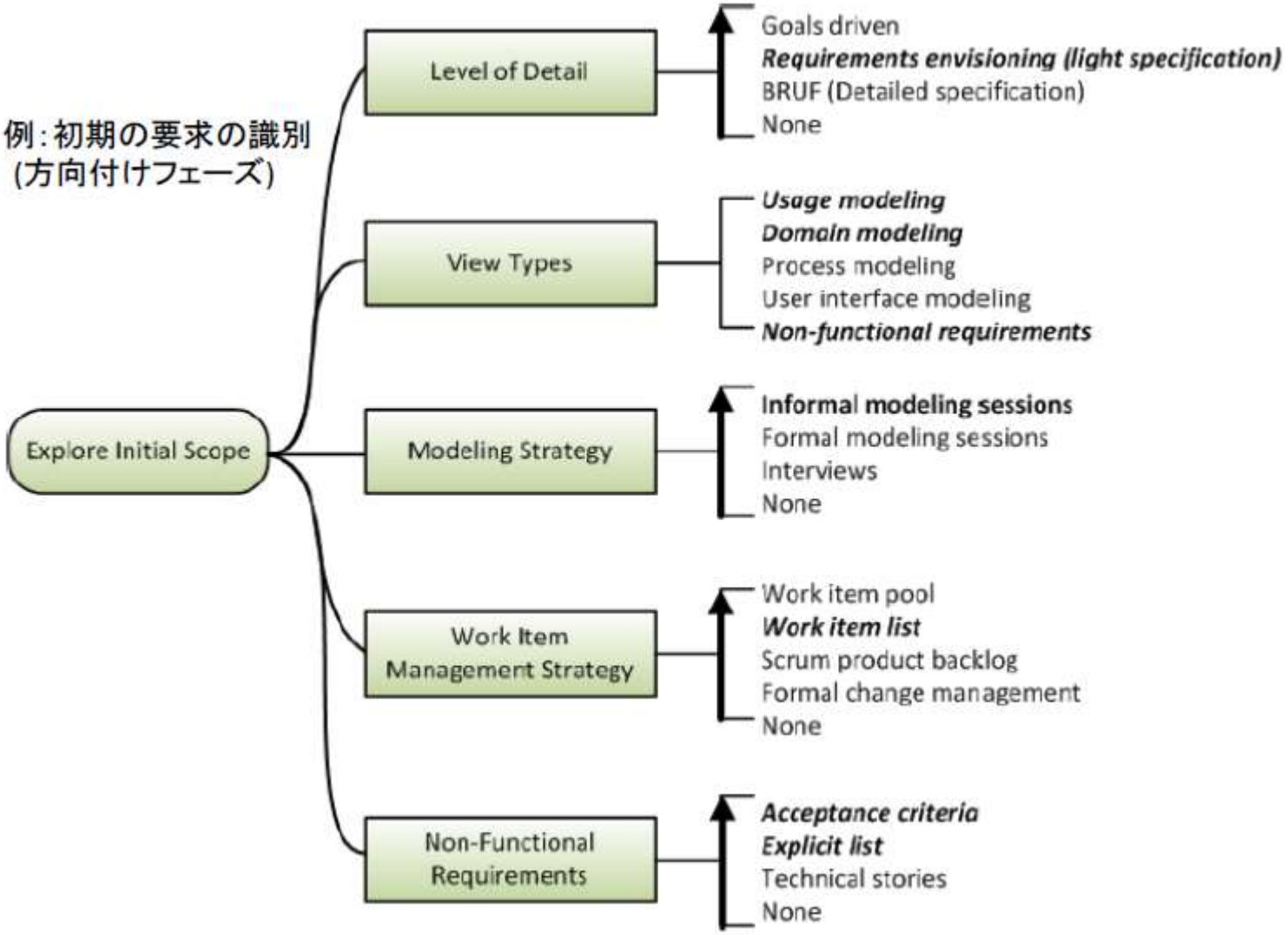


大きなライフサイクル視点。
４つの典型(ただし、テーラリング重要)

1.Scrum+RUP
2.Advanced Lean
3.Lean CD
4.Lean Startup

平鍋さんのスライドを引用させて頂きました

例：初期の要求の識別
(方向付けフェーズ)



平鍋さんのスライドを引用させて頂きました

SAFe と DAD 共通点

- アジャイル宣言を大切に
- これまでの手法のハイブリッド
- 「リーン」コンセプトが浸透
- リーダーシップの重要性
- RUPの要素によってモデレートに
- アーキテクチャの視点

平鍋さんのスライドを引用させて頂きました

スケーリング戦略

フレームワーク	SAFe, DAD, LESS Spotifyモデル
フロー	バリューストリームマップ Kanban
イベント	Scrum of Scrums MetaScrum

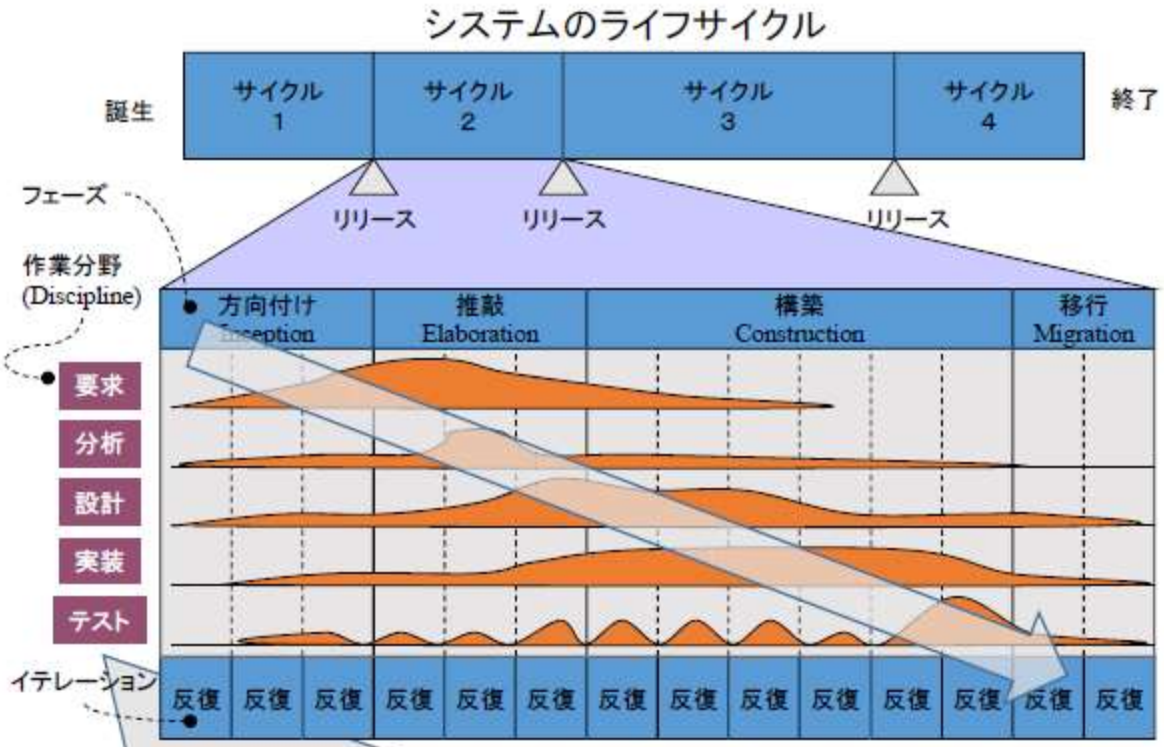
楽天 Rakuten

厳密な分類ではありませんが…

22

川口さんのスライドを引用させて頂きました

最大の特徴： サイクルとフェーズ

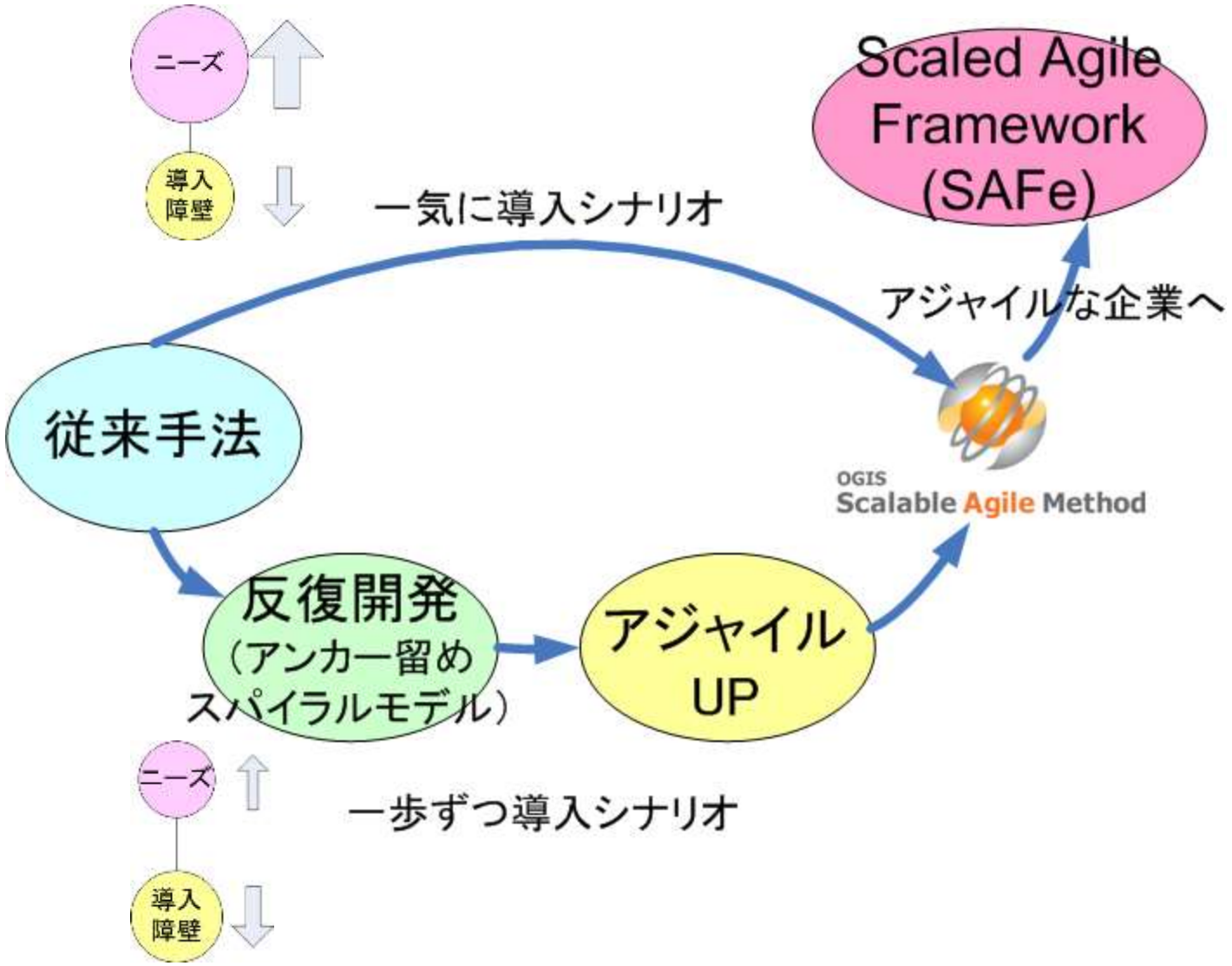


柔軟ではあるが、何かの弾みに、ウォーターフォール（計画主導型）に戻ってしまう危険性をはらむ（対角線に注意）。
計画主導となったとたんに反復の意味がなくなる。（依田）

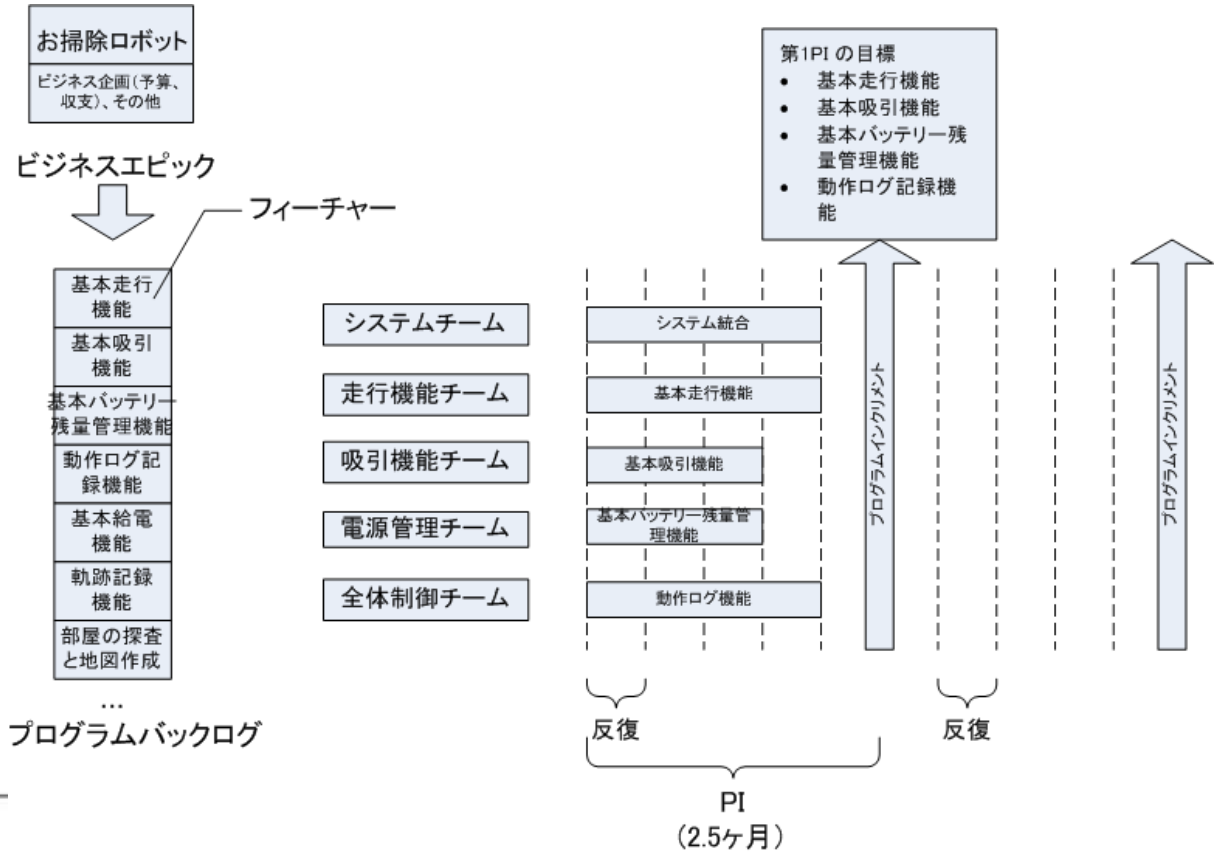
15

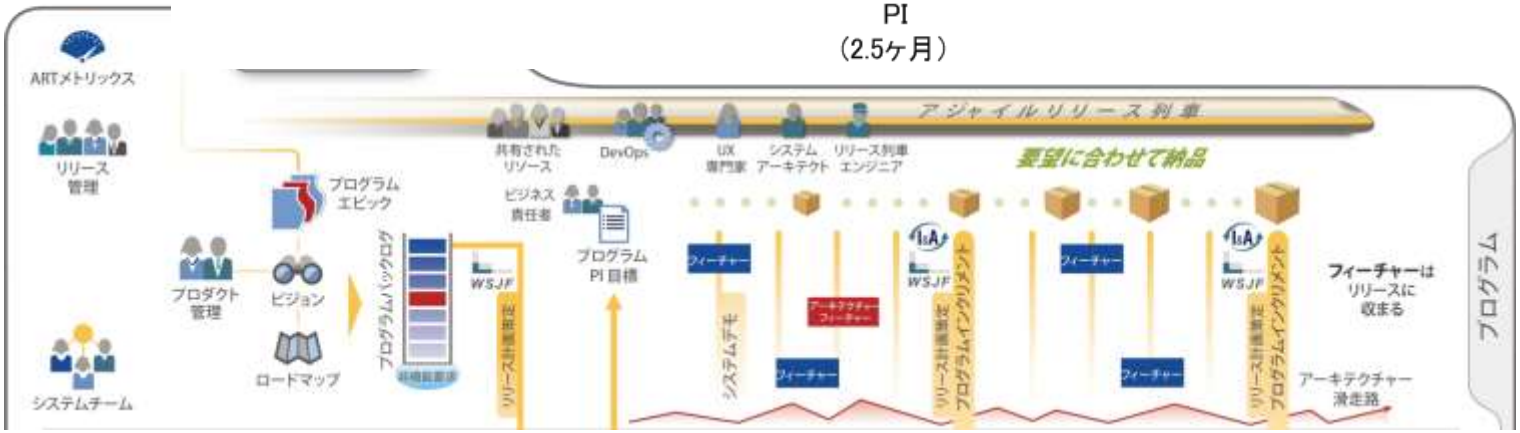
依田さんのスライドを引用させて頂きました

アジャイル開発の導入イメージ



妥当性の確認（フィードバック）





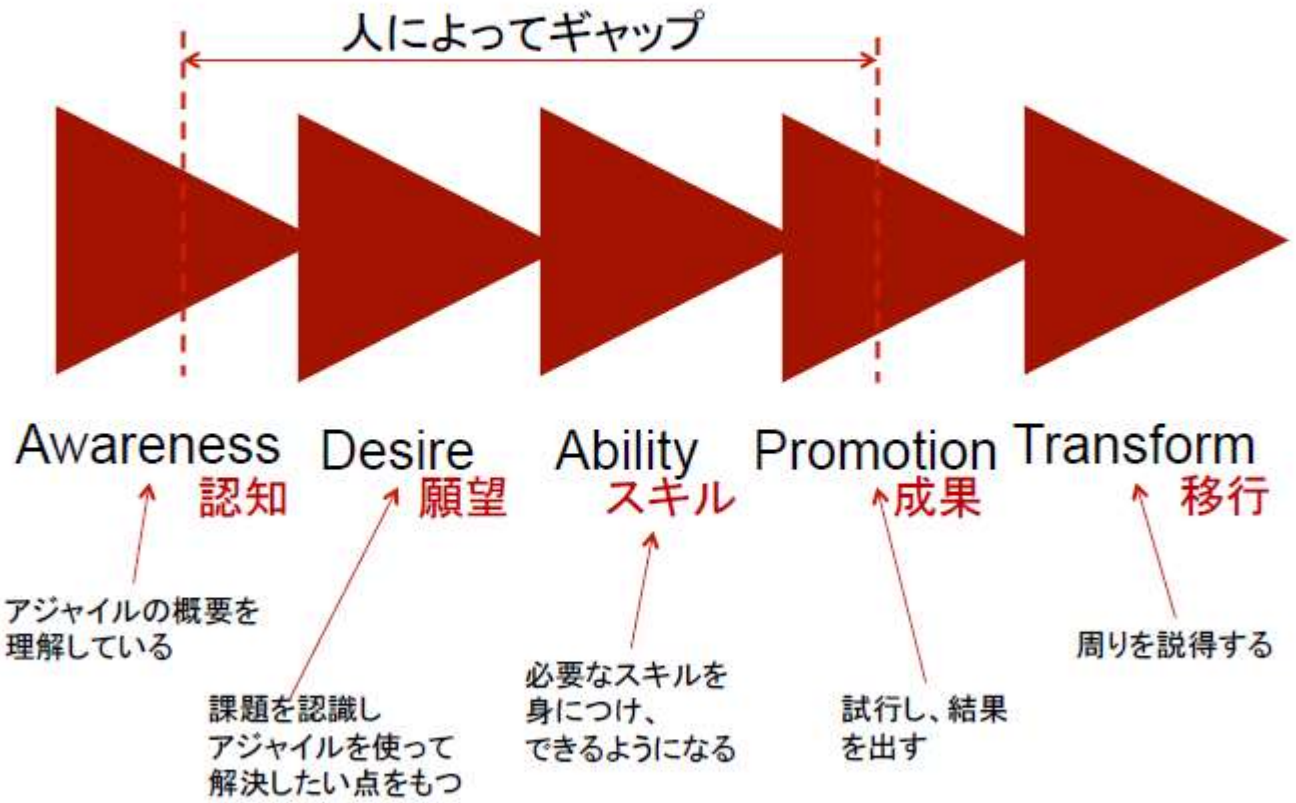
戦術：事例との関係

- 2015年7月：東京海上日動システムズ様
  - スプリントと反復による大規模アジャイルの実践
- 2015年11月：新日鉄住金ソリューションズ様
  - デザイン思考＋アジャイル開発の実践
- 2015年12月：ゼンアーキテクト（カプラン様）様
  - 既存のガバナンスの考え方の転換
- 2016年3月：リクルートライフスタイル様
  - 大規模アジャイルの立ち上げ、運営、改善

普及と育成

- 普及
  - 組織内にどう広げるか？
- 育成
  - 技術者のスキルレベルをどう上げるか？

アジャイル導入

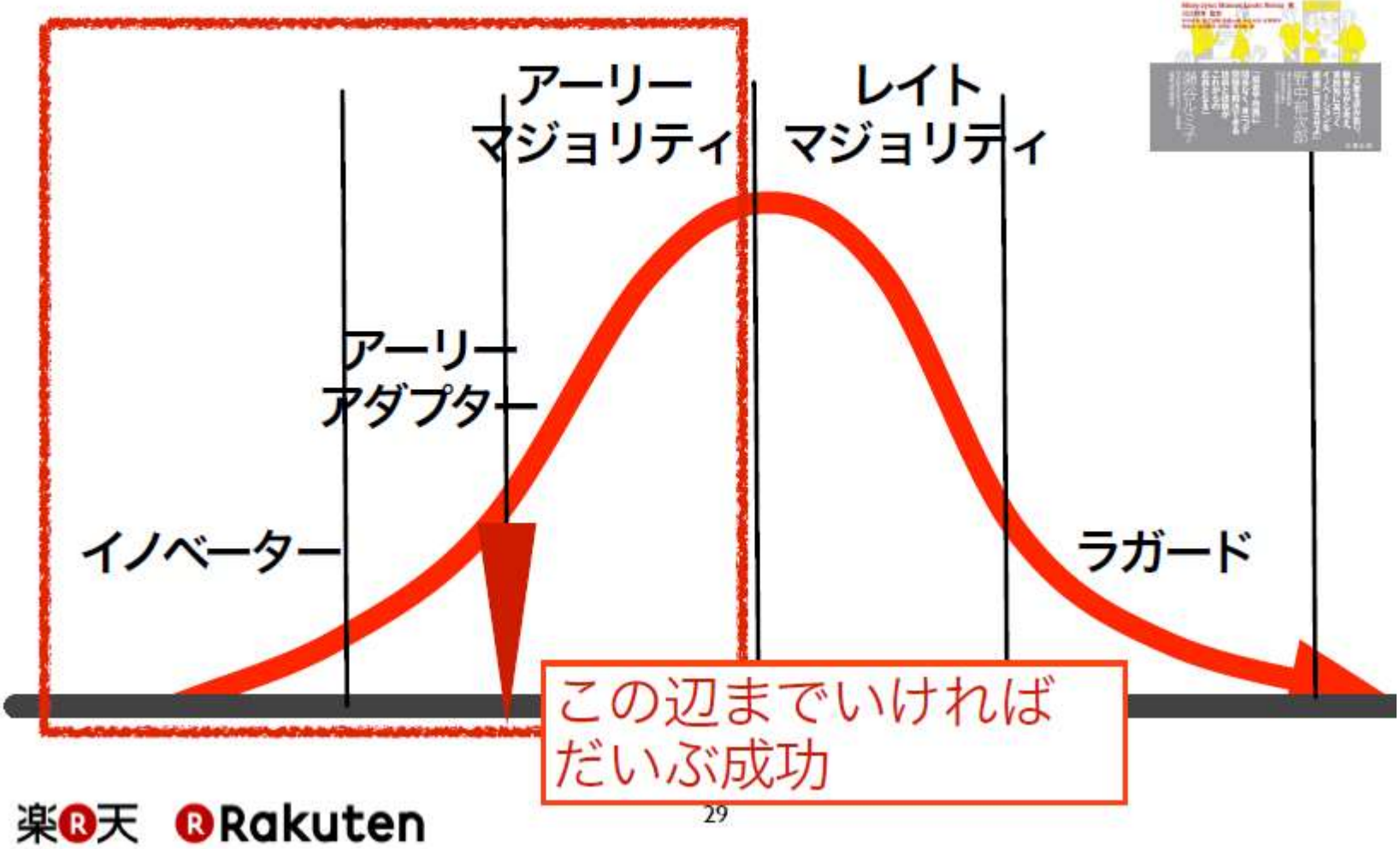


楽天 Rakuten

14

川口さんのスライドを引用させて頂きました

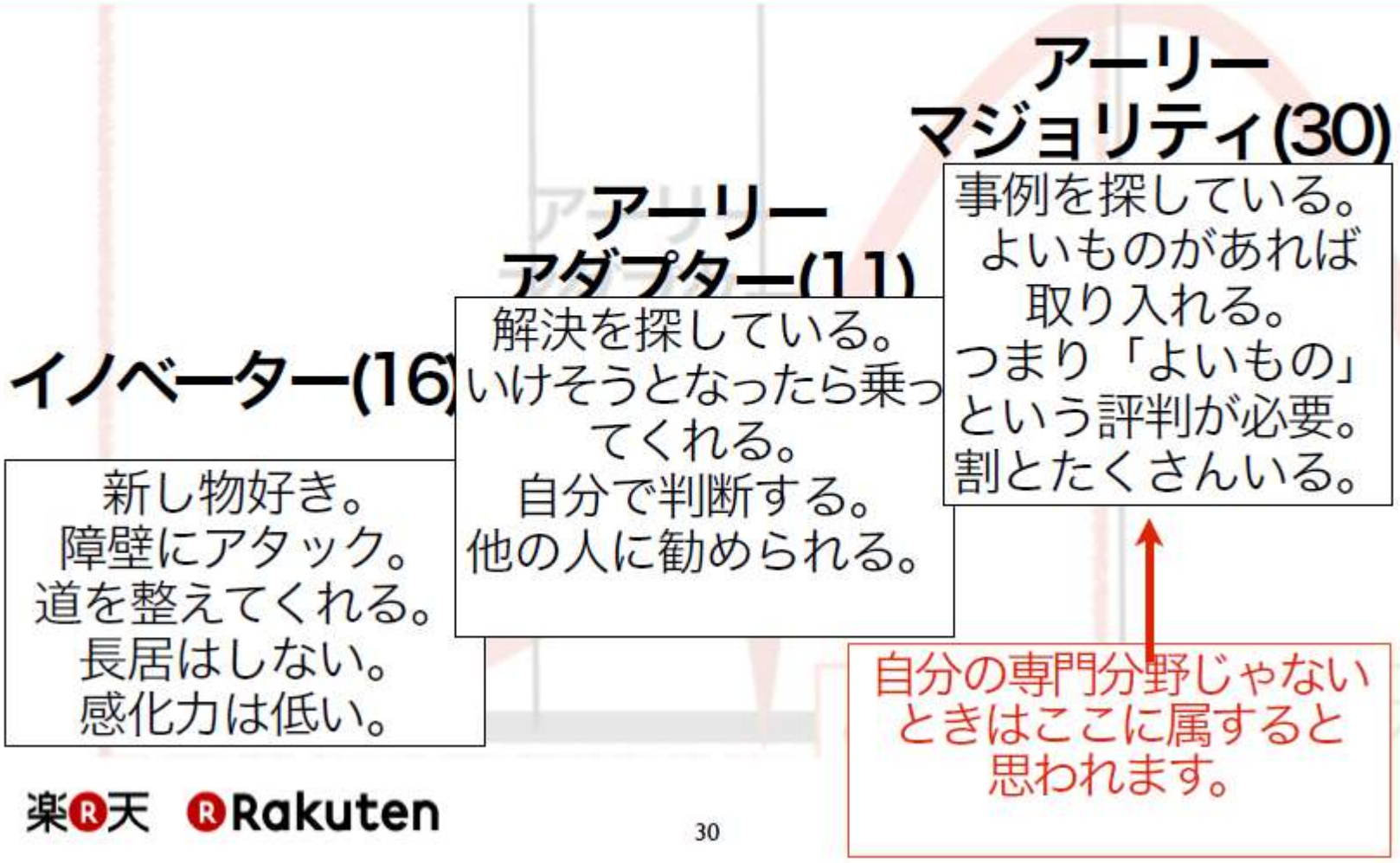
イノベーション普及曲線



楽天 Rakuten

川口さんのスライドを引用させて頂きました

普及序盤の３タイプ



川口さんのスライドを引用させて頂きました

読者の立ち位置

エバンジェリスト(1)
Evangelist

新しいアイデアの
価値を認め、
普及したいという
情熱にあふれた人。
正式な任命を待たず、
ボランティアで活動
を開始する。

正式な推進担当者(29)
Dedicated Champion

普及活動を業務の一部にすることを認められた人。
(やらされる人ではなく)
プロのエバンジェリスト。
そのかわり、説明義務や
効果の立証責任を負う。

~~懐疑的な人や理解していない人たち~~
ともつきあわなければならない。

8～12章

3～7章

楽天 Rakuten

31

川口さんのスライドを引用させて頂きました

Eアジャイル開発での社内アーキテクトのスキル

- 自社のビジネス戦略と特徴を理解できる
- 経営学の基礎を一通り理解している
  - 財務会計、ＳＣＭ、マーケティング、統計、ＨＲなどのセオリー
- データモデルを始めとする各種モデリングが出来る
  - 表記法はメジャーなもので良い（応用は後からで）
- 抽象化能力のトレーニングが出来ている
  - プログラミングが最たる物だがモデリングでも代替可

- 中規模以上のシステム開発における上流経験がある
  - 下流はプログラム言語のメカニズムは知っていた方が良い
- 最新のＩＴで何が解決できるかを知っている
  - イノベーション　＝　ビジネス-ＮＥＥＤＳ　＋　ＩＴ-ＳＥＥＥＤＳ

中山さんのスライドを引用させて頂きました

プロダクトオーナーの育成

- 「アジャイルソフトウェア要求」にいくつかの事例が紹介されている
  - ITまたは業務に通じた人に、他方の経験を積ませる
  - プロダクトオーナーの経験者を中途採用する
  - ITに通じた人と、業務に通じた人が相補的に役割を担う（後者が優先度を決める）

個人的には、業務に通じた人をITに通じた人（分析者等）がバックアップするのが現実解ではないかと思う。

	戦士	武闘家	僧侶	魔法使い	盗賊	旅芸人
特性	ソフトウェアはコードを書かなければ動かない！開発言語に精通し、素早く的確な実装を行う。テストを書きながら、スピーディかつ高品質なコードを生み出し、要件を満たしていくのが戦士の得意技だ！	もっと早く、もっと巧妙に。アプリケーションの本質をつかみ、各種の設計手法を適切に使い、ROIの高いソフトウェア開発を実現するのがソフトウェアのプロ、武闘家の十八番だ！	コードは健全に管理できているか？メンバーの知識は共有され、一人休んでも仕事が止まらない状態になっているか？開発チームの健全性を維持するのが僧侶の役目だ！	あらゆる操作シーケンスを手作業で試すことなど不可能だ。魔法使いは不可能を可能にし、最低限の工数でほぼ全体をカバーするテストを設計し、省力化しながら確実に実行する！	すべての発明を自分ですることはできないし非効率だ。コミュニティや書籍、達人たちとの相互作用を用いて効率的に知識を吸収し、試していくのが盗賊のスキルセットだ。スピーディに成長していく！	要件定義、利用者、マネージャ、コミュニティ、チームメンバー…、あらゆる人々と効率的で健全で前向きなコミュニケーションをとるスキルは見過ごせない。プロセスをゲームに！目標をタスクに！
LV1	使用言語の初級研修を受けるか、独学によって本番向けのコードを生み出せるようになっている。開発用サーバにアクセスできどこにコミットすればリリースできるかを知っている。	クラスやオブジェクトを、明確に意識してコードを書くことができる。	自分の書いたコードを、ほかの人にコードレビューをしてもらうことができる。	要件に従い、確認項目を洗い出すことができる。正常系、少なくとも一つのハッピーパスを書き出してテストすることができる。	社内の自分の専門分野のコミュニティや、詳しい人を知っている。難しい問題が出たときにはその人に質問してみることができる。	成果を出したら発表する。なにかいいことを見つけたら発表する。チームや部門内に迅速に共有するための資料を作ることができる。
LV7	xUNITなどでユニットテストを書くことができる。各メソッドの正常系、異常系を列挙してテストを実装する。どのように実行すればよいかを理解している。	単一責任の原則を意識し、粗結合密結合を考えて設計できる。	コードレビューをファシリテートし、前向きな示唆を与えることができる。コードレビューは門番ではなく、前向きな改善を生み出すためのアドバイスが重要と理解している。	ハッピーパス以外のパスを網羅的に洗い出して、最低限の組み合わせですべての画面や操作を一通りテストすることができる	自分にあったトレーニングは自分にしか発見できない。定期的に情報を探し、次に目標とするスキルを得られるトレーニングを提案できる。	ConfluenceとJIRAは私に任せろ。ドキュメントとミーティングの設計をして、円滑で効率的なコミュニケーションを作り出すことができる。
LV14	ユニットテストのカバレッジは十分だ。かつ、スローテストはなく、ごく短い時間にテストスイートを実行することができる。	業務要件とマッチした利用価値の高いクラスやオブジェクトの設計を意識している	ソースコード管理はGitだ。コードの共同所有を指導し、チームの誰もが自分の生み出したコードを理解した状態に持っていける。	非機能要件や外部システムとの接続など、要件にかかわらないテスト項目の洗い出しとテストの実施を計画できる。	社外の実践者コミュニティは専門情報の宝庫だ。いくつかの勉強会に参加してみて、自分にあった、役に立つ勉強会やコミュニティを見つけている！	開発用のサーバ環境はスピードを支配する。必要な開発ツールがなければ生産性は上がらない。適切なタイミングを見切って先手を打って環境を整備できる
LV21	ユニットテストが向いていないテストもある。各テストの重要性は変化する。必要なテストを常に実行するセットにおさめ、常に実行しないテストスイートも用意する。	業務のヒアリングから言葉の抽出し、モデリングを行い、クラス設計や動作シーケンスなど必要な設計と表現/伝達/文書化を行うことができる	JenkinsなどのCIを使って、日々のコードの健全性は常にチェックできるようにしている。毎日、チェックするのだ。	テストデータを用意できる。ハッピーパス、全量データ、負荷テスト用、特殊ケースなど必要なテストデータをスクリプトなどを駆使して作り出せる	必要なコミュニティがなければ自分で作ればいい。自ら専門的なコミュニティを立ち上げ、社内または社外の人が集う環境を整備した！勉強会の開き方を把握した。	インタビューはソフトウェアの成否を決める重要な情報を得る手段だ。利用者やステークホルダーへの質問を設計し、インタビューを行い、まとめることができる。インタビュー結果の集合から要件を抽出することができる
LV28	テストファーストだ。先にテストを書いてからメソッドを実装する。ペアプログラミングでナビゲーターとドライバーが適切にコミュニケーションしながら進む。	DDDやデザインパターンを利用して、即座に有効な設計を適用できる引き出しを持っている。	リリース作業に手作業を絡めると危険が多い。テスト作業を自動化し、リカバリープランを用意しておく。本番機と開発機の差異も徹底的になくし、スムーズで再現可能なリリースを行えるように整備できる。	探索的テストの名人だ。他の人がいくらやっても発見できない穴を、いつも見つけてしまう。	技術を提供する相手との協調/協力関係は作れているか？意識しているか？あなたの活動を上司に報告して、理解を得ているか？	コーチングはチームやエンジニアに円滑に働いてもらうための必須のスキルだ。よい相談相手になって、気持ちよく仕事を進められるように手助けするぞ。

川口さんのスライドを引用させて頂きました

普及と育成：事例との関係

- 2015年8月：KDDI様
  - ペアプロなどを通じた内製化
- 2015年12月：ゼンアーキテクト（カプラン様）様
  - ラッセルによる技術移転
- 2016年5月：ニッセイ情報テクノロジー様
  - 学びながらのプロジェクトの実践

日本固有(?)の課題と落とし穴

- スケール
  - 産業構造
- 落とし穴
  - 一般的な失敗パターン
  - 安直な大規模化
  - 体制・組織
  - 契約・商慣習

Agileのスケール方向

契約を挟み、多段の下請け構造の中で、どうしたらゴールを共有することができるだろう？

48

平鍋さんのスライドを引用させて頂きました

■ アジャイル開発？失敗のパターン

- 準備不足
  - 準備が必要ないというイメージが悪影響
  - たとえば、本来あるべきAS-ISの情報収集がおろそかになる
  - 「AS-ISの情報収集なんてウォータフォールみたいだ」
  - 「アジャイルで回すうちにいい案が出てくる」
- プロトタイプの同床異夢
  - 要件を確認するためのプロトタイプなのか
  - 技術検証のためのプロトタイプなのか
  - どちらも同時に狙うのか
  - ユーザー、ベンダー、担当者、それぞれが都合のよい解釈をして、前提が不明瞭になる
  - たとえば、「この先、アーキテクチャー設計の時間はとるのか、とらないのか」がわからない
- 実はみんな大ロット体質だった・・・
  - ユーザー、ベンダーともに、短い時間間隔で反復的に、調査し、検討し、決定し、決定に対して責任を持つ体質にない
  - （みんなトヨタの工場で働けるのかな？）

依田さんのスライドを引用させて頂きました

■ 大規模ではさらにリスクが高まりそう

- 始めやすさが裏目に出る
- しかるべき手順を踏まないことから発生するリスクが問題となりそう。たとえば、
  - 経営の承認、経営戦略との一致性が確認されないまま進行する
  - アーキテクチャー設計、方式設計の時間と予算が確保されないまま進行する
  - プロジェクト運営組織が貧弱なまま進行する
  - 大組織に必要なコミュニケーション手段が確保されないまま進行する
- 従来のウォータフォール型開発で大手ベンダーにお任せになっていたところが、すべて心配のネタになるということ
- この手順に関わる問題は、どう解決すれば良いのか。

依田さんのスライドを引用させて頂きました

アジャイル開発導入の課題（体制・組織）

- オンサイト顧客不在
  - 顧客ニーズを代表するメンバーのプロジェクト参画が困難
    - ユーザ企業が理解を示さない。業務理解者は他業務で多忙。
    - ベンダーへの丸投げ体質。積極的関与、リスク共有を望まない
  - ユーザー側／ベンダー側とにまたがった一体化チームは困難
    - 利害が合わない

- 大規模開発・分散開発への対応
  - アジャイル開発は10人前後のチームを単位としている（少数精鋭）
    - 少人数でのコミュニケーションを前提としている
  - 複数のアジャイル開発チームをマネジメントする方法が確立されていない。（試みは始まっている）
    - Scaled Agile Framework (SAFe)
    - Disciplined Agile Delivery(DAD)
    - スクラムオブスクラム： 各チームリーダー(スクラムマスター)が定期的に集まって、プロジェクト全体の進捗や課題、方針を棚卸して決定する
  - 一箇所に集結してのチーム開発が基本。分散開発に関してそのノウハウがない
    - オフショアなどで機会が増えている
    - スカイプや画面共有などコミュニケーションを円滑に進めるためのツールは進化している

MethodoLogic
A STRATEGIC TECHNOLOGY COMPANY

山岸さんのスライドを引用させて頂きました

アジャイル開発導入の課題（契約・商習慣）

- 最終的な実現機能を事前にコミットできない
  - こまめなイテレーションの中で優先順位に従って、盛り込まれる機能が決定される。事前にわからない。
  - 決まった予算とスコープがないと契約ができない。社内稟議を通せない
- 責任の所在が明確にならない
  - 従来の一括請負のウォータフォールでも入らないものは入らないが、受けた側が責任をとって超過分を引受ける。どこで辻褄を合せるかの問題。
    - リスク分のサバを読んだ高い見積り
    - 仕様変更への厳しい縛り
  - ゼネコン的商流（多段階の下請け）で一体化チームの構成は困難
    - チーム内でゴール、ミッション、リスクを共有する必要がある
  - ユーザ側はリスクをとりたくない
    - 丸投げ体質

MethodoLogic
A STRATEGIC TECHNOLOGY COMPANY

山岸さんのスライドを引用させて頂きました

課題と落とし穴の区別

- 日本固有（？）の課題
  - 企業の方針ですぐに克服できる課題と解決に時間がかかる課題がある
    - 契約：すぐに克服できる可能性あり
    - 育成：中途採用が困難だと、解決に時間がかかる
- 落とし穴
  - 理解しないとはまる可能性があるので、理解した方が良い

課題と落とし穴：事例との関係

- 2015年12月：新日鉄住金ソリューションズ様
  - 利害関係者との関係、物理的な距離
- 2016年4月：ウルシステムズ（製造業）様
  - より高度なマネジメント能力が必要

事例の多くは、「うちでもアジャイル開発やりました」アンチパターンにも落とし穴にもはまっていない！

- 適切な動機、戦略の下で、目的志向で取り組んだためではないかと考えらえる

事例の概要のご紹介

2015年7月から2016年5月の期間のご紹介事例

東京海上日動システムズ株式会社
押井英喜様、花宣典様

発注者と開発者の絶妙な連携

- 保険契約手続きの電子化
- 戦略：◎
- 戦術：
  - 開発は委託
    - 反復単位の契約
  - POは協力的
  - SMは2名体制
  - テストの自動化

KDDI株式会社
川上誠司様、藤井彰人様

企画・開発・運用・外部ベンダが一体となったアジャイル開発の活用

- KDDI Business ID
- 戦略：◎
- 普及/転換
  - 開発の一部内製化
  - ３段階の改革
    - アジャイル開発の導入
    - オフショアアジャイル開発
    - 企画承認のスピードアップ

藤井　彰人, 通信事業者におけるLean & Agile 適用事例―企業向けITとエンジニアの物心両面での改善のために―, 情報処理学会デジタルプラクティス Vol.7 No.3, 2016, pp. 235

新日鉄住金ソリューションズ株式会社 斉藤康弘様

デザイン思考＋アジャイル開発によるより良いシステムの実現を目指した

- 病院の病床管理
- 戦術
  - ユーザーニーズの理解のためにデザイン思考を活用
- 障害
  - 決定権のある利害関係者との直接話ができない
  - UXアーキテクトと開発チームが物理的に離れていた

ご講演資料：https://easg.smartcore.jp/C23/file_details/QTJkVVpBPT0=

コベルコシステム株式会社
松本篤様

アジャイル開発により保守開発のデリバリー改善を狙った

- 戦術
  - スクラムとムダの排除、意思決定の迅速化、情報の可視化を組み合わせた
    - 業務負荷の平準化が改善できなかった

ご講演資料：https://easg.smartcore.jp/C23/file_details/V2o1VWF3PT0=

株式会社ゼンアーキテクツ
岡勝様

ラッセルによるアジャイル開発の導入支援

- 人材サービス用のシステム
- 戦略的：◎
- 普及/転換
  - 動機とコミットメント
  - 既存の管理統制スキームからの転換
  - プロダクトのビジョンとスコープのコントロール

ご講演スライド：https://easg.smartcore.jp/C23/file_details/VXpkVVpRPT0=

楽天株式会社
荻野恒太郎様

継続的システムテストによる迅速で高品質のリリースの実現

- 戦略：◎
- 解決策
  - 継続的テスト
  - 自動テスト
  - アジャイルテスター

ご講演資料：https://easg.smartcore.jp/C23/file_details/VkRCVWFnPT0=

株式会社リクルートライフスタイル
塚越啓介様、今井恵子様

POと開発チームの成長を通じた大規模アジャイルの実践

- 海外版AirREGI
- 戦略的：◎
- 戦術
  - チーム編成
- 普及/転換
  - 導入/育成/改善
  - 周囲の理解

ご講演資料：https://www.slideshare.net/keisuketsukagoshi/ss-59705472

ウルシステムズ株式会社
河野正幸様

基幹系システムでのアジャイル開発ではマネジメント能力が問われる

- 製造業のSCMシステム
- 戦略的：◎
- 戦術
  - マネジメント能力を備えた人材の確保
    - 展望、タイムリーな意思決定等に長けた

ご講演資料：https://easg.smartcore.jp/C23/file_details/VWpjSlBnPT0=

ニッセイ情報テクノロジー株式会社 中野安美様、嶋瀬裕子様

コーチの支援の下で経験者ゼロの状態から価値ある製品をリリース

- 福利厚生システムの外販化
- 戦略的：〇
- 普及/転換
  - 教科書どおりからスタートし、改善
  - 早期フィードバックによる製品の改善
  - テストツールの活用

ご講演資料：https://easg.smartcore.jp/C23/file_details/QUdVQk5RPT0=

株式会社オージス総研
竹内俊裕様

リリース間の振り返りにより後続リリースでの品質改善

- 非公開
- 戦略的：？
- 戦術
  - 反復開発＋リリース間で改善
    - 不具合の発生原因を「設計と実装のギャップ」と特定し、その対策を次のリリースの開発で講じた

ご講演資料：https://easg.smartcore.jp/C23/file_details/V2o4RE5nPT0=

まとめ

- 「うちでもアジャイル開発をやってみました」アンチパターンにはまると、「プロジェクトは成功してもアジャイルのメリットは不明確」という結論になりかねない

- 「うちでもアジャイル開発をやってみました」アンチパターンに陥らないためには、5つの側面で考えていくことが重要である

- 事例は、適切な動機、戦略の下で、目的志向で取り組むことにより、「うちでもアジャイル開発をやってみました」アンチパターンを克服できることを示している

書籍「エンタープライズアジャイルの可能性と実現への提言」

- 本書は実行委員の方々や勉強会セミナーでの発表者の方々のスライドを多数引用しており、説明が充実した解説書ではなく、「エンタープライズアジャイル」に関する様々な話題を概観するハンドブック的な書籍です。
- ページ数が「参考文献」を含めて72ページと薄い本です。

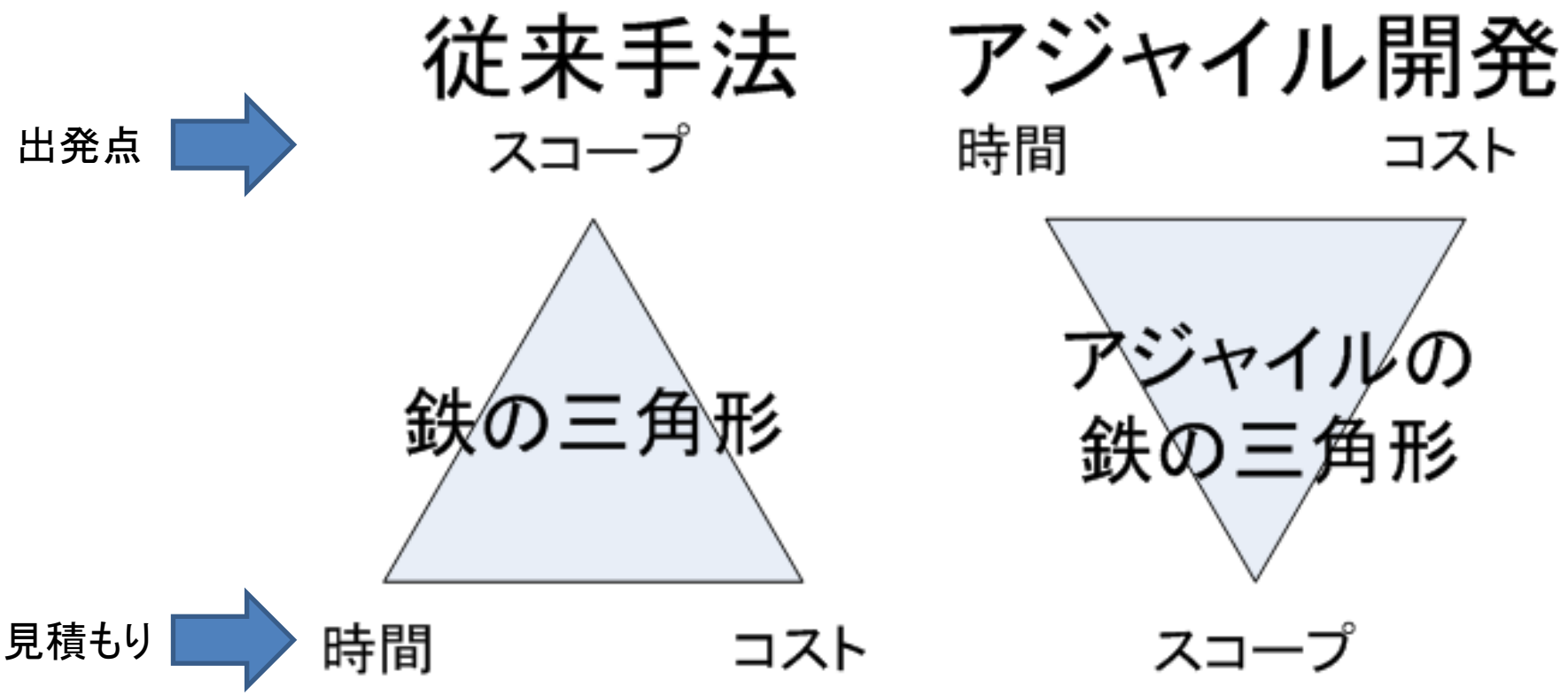
参考資料

- 平鍋 健児,アジャイル開発の現状と未来～エンタープライズアジャイルの可能性
  - https://easg.smartcore.jp/C23/file_details/VXpBSk9RPT0=
- 中山 嘉之, EA（エンタープライズアジャイル)にはEA（エンタープライズアーキテクチャー)が必要
  - https://easg.smartcore.jp/C23/file_details/VlRZSE9BPT0=
- 依田 智夫,エンタープライズ・アジャイル開発が果たすべき役割
  - https://easg.smartcore.jp/C23/file_details/QVdWVFp3PT0=
- 川口 恭伸,部門アジャイル ～ 複数チームのコラボレーションをもっとよくするために
  - https://easg.smartcore.jp/C23/file_details/V2o1V1lRPT0=

参考資料(続き)

- 山岸 耕二,エンタープライズシステムの継続的リフォームにおけるアジャイル開発
  - https://easg.smartcore.jp/C23/file_details/VXpBSk9RPT0=
- 竹政 昭利,サービスデザイン思考のエンタープライズ・アジャイルにおける位置づけ
  - https://easg.smartcore.jp/C23/file_details/VlRZSE9BPT0=
- 鈴木 雄介,エンタープライズアジャイルと全体最適について
  - https://easg.smartcore.jp/C23/file_details/VWpaUll3PT0=

従来手法とアジャイル開発の違いの理解



もちろん同等の品質を達成することが前提となる

2015年7月-10月の活動

開催日	ご講演テーマ	ご講演者
7/15（水）	アジャイル開発の現状と未来 ～エンタープライズアジャイルの可能性	東京海上日動システムズ(株) 押井様、花様
8/26（水）	大企業でアジャイルを実現するために ～　KDDI様のアジャイル導入事例	KDDI（株） 川上様
9/30（水）	エンタープライズアジャイル探求の軌跡（その1）	藤井 拓
10/21（水）	エンタープライズアジャイルと全体最適について ～アーキテクチャ設計とウォーターフォールの必要性について～	グロースエクスパートナーズ（株） 鈴木 雄介様

2015年11-12月の活動

開催日	ご講演テーマ	ご講演者
11/27（金）	•デザインシンキングアプローチによるエンタープライズレベルのAgility向上の取組 ～病床割当業務IT化事例を通した考察～	新日鉄住金ソリューションズ（株） 斉藤 様
	•保守運用フェーズにおけるアジャイル開発の適用について	コベルコシステム（株） 松本 様 （株）神戸製鋼所 大和 様
12/9（水）	•キャプラン様事例に学ぶ エンタープライズアジャイル内製プロジェクトを立ち上げる前に考慮すべき3つのこと	（株）ゼンアーキテクト 岡 様

2016年1-3月の活動

開催日	ご講演テーマ	ご講演者
1/15(金)	• 「納品のない受託開発」から考えるエンタープライズアジャイル ～自己組織化された強い組織を作るためのマネジメント～	(株)ソニックガーデン 倉貫　様
2/17(水)	• 楽天の品質改善を加速する継続的システムテストパターン	楽天(株) 荻野　様
3/18(金)	• Agile Transformation & Leadership in Enterprise ~KDDIでの驚きと学び~	KDDI(株) 藤井 様
	• 大規模なチームを安定させるために必要なこと	リクルートライフスタイル(株) 塚越 様、 今井 様

2016年4-7月の活動

開催日	ご講演テーマ	ご講演者
4/15（金）	•エンタープライズアジャイルで成功するために必要なもの	ウルシステムズ（株） 河野　様
5/20（金）	•金融系IT企業におけるスクラムへの挑戦	ニッセイ情報テクノロジー（株） 中野 様、嶋瀬　様
	•振り返りによる反復開発プロジェクトの改善事例のご紹介	（株） オージス総研 竹内 様
6/22（水）	•UXアプローチがもたらすエンタープライズアジャイルへのインパクト	ソシオメディア（株） 篠原　様
7/20（金）	•コカコーラ vs. ペプシ･･･みたいな話なのか？的視点からみたDA2.0	日本ヒューレット・パッカード（株） 藤井　様
	•機敏な製品リリースを可能にする企業内の連携モデルを提示するSAFe (Scaled Agile Framework) のご紹介＋α	藤井　拓

エンタープライズアジャイルの狙い、解釈、重視する要素（10/20, 29）

	依田さん	鈴木さん	平鍋さん	中山さん
何を改善し（狙いたい）たいと考えているか？	現状の開発依頼者と委託先の関係により発生している損失を削減する	ユーザー様がやりたいことを実現できるように現状を改善したい	日本で技術者がビジネスに貢献し、幸せになる	3)が改善したいこと
立ち位置は1), 2), 3)のいずれか？	価値があるのは3)だが、仕事でしたいのは2)	やっていることは1)の立場	1), 2), 3)のすべてに興味がある	1)と2)。 3)は目的。
重視するアジャイルの要素は何か？	WFにおけるPMBoK、EVMのような科学的な方法、反復開発	WFでもCI、テストの自動化は適用可能 WFとアジャイルのバランスを取る	？	スケールアップはなじまない。 エンタープライズ・モデリングで限りなく疎結合化しアジャイルを適用。

エンタープライズアジャイルの狙い、解釈、重視する要素

	細川さん	山岸さん	竹政さん	川口さん
何を改善し(狙いたい)たいと考えているか？	プロジェクトが成果を出すこと	チームの大規模化による効率の低下	お客様の本当のニーズに対応する	2)、3)の立場によって狙いは異なる？
立ち位置は1), 2), 3)のいずれか？	うまく機能すればどれでもよい	1)「エンタープライズシステム」…	どれもありだが、強いて言えば3)	2)と3)。
重視するアジャイルの要素は何か？	XPのプラクティスをもっと重視すべき	POAgile：人間が効率的に活動できるチームレベルのアジャイル。	開発プロセスに対しては中立だが、試行錯誤を許容する点でアジャイル開発と親和性が高い	2)の立場においてはスクラム、3)の立場においてはリーン

MMFの開発順序の判断(SPNV)

- 各MMFの期間毎のコストと売り上げを予測する

- 各MMFの利率を想定してDCFを求め、DCFの総和からNPVを求める
- さらに各MMFの開発開始をずらして、それらのDCFから開発開始をずらした場合のNPVを求める

- 各MMFの開始期間毎のNPVを並べる（これが順序調整されたNPV (SPNVs))
- 各MMFのSPNVsを比較すると、早く開発すべきものと遅らせても良いものを判断できる

2つのMMFのコストと売り上げ

あるシステムの機能群として2つのMMFが考えられる場合、各々のMMFのコストと売り上げを推定する

MMF	期間毎のコストと売り上げ											
	1	2	3	4	5	6	7	8	9	10	11	12
A	−200	−200	100	120	140	160	200	220	240	300	320	340
B	−250	50	50	80	100	120	140	160	180	200	200	200

出典：Mark Denne, Jane Cleland-Huang, Software by Numbers: Low-Risk, High-Return Development , Prentice Hall, 2003

MMF Aの開始時期ごとのNPVの算出

MMF A（月）												年率10%の現在正味価値（NPV）
1	2	3	4	5	6	7	8	9	10	11	12	
−200	−200	100	120	140	160	200	220	240	300	320	340	1604
0	−200	−200	100	120	140	160	200	220	240	300	320	1285
		−200	−200	100	120	140	160	200	220	240	300	986
			−200	−200	100	120	140	160	200	220	240	708
				−200	−200	100	120	140	160	200	220	486
					−200	−200	100	120	140	160	200	283
						−200	−200	100	120	140	160	101
							−200	−200	100	120	140	−44
								−200	−200	100	120	−170
									−200	−200	100	−277
										−200	−200	−365

出典：Mark Denne, Jane Cleland-Huang, Software by Numbers: Low-Risk, High-Return Development , Prentice Hall, 2003

2つのMMFのSNPVs

NPVが正になるためには、SNPVsが負になる時期以前に開発を開始しなければならない→MMFの開発順序を決める

MMF	期間											
	1	2	3	4	5	6	7	8	9	10	11	12
A	1604	1285	986	708	486	283	101	−44.3	−170	−277	−365	N/A
B	1138	949	761	574	407	260	132	22	−68	−140	−184	−248

出典：Mark Denne, Jane Cleland-Huang, Software by Numbers: Low-Risk, High-Return Development , Prentice Hall, 2003

プロダクト開発フローでは、「遅延のコスト (CoD)」でフィーチャー等の優先順位付けを行うことを推奨している