JUAS研究プロジェクト
「基幹系システムへのアジャイル活用の研究」

2023年４月13日

基幹系アジャイル適用PJ

研究PJの目的

日本ではアジャイルはなかなか定着しない、と言われる。

– 実施事例、成功事例も増えてはいるが、定着というには不十分。

それは、アジャイルが、日本の古くからのモデル（SIerモデル）と対極にあるからではないか。

※SIerモデル：大手ベンダへの一括請負。いわゆる丸投げになることも多い。

また、現場での意思決定が、日本の文化に合わないからではないか。

成功事例の各社が実施しているのは、基幹系とは切り離した独立新規システムが多い。
その部分での実例は増えてきているも、基幹系には手を付けない。（つけられない）

※基幹系システムは、上記SIerモデルの象徴的なもの。
※基幹系システムの開発では、現場への権限移譲も難しいと考えられる。

これが、いくら実例が増えても、日本ではなかなか定着感がない理由ではないか。

→ **基幹系システムへの適用に対する壁をなくし、日本でのアジャイルの定着に貢献したい。**

対象としている読者

日本でもアジャイルを広く普及すべく、以下のような方々に読んでいただきたい。
主に現業として担当しているシステムを持ち、開発に何らか変化を考えているような人に対して参考となるものを目指している。

- ✓ **アジャイルをやりたいが、踏み出せない人**
  - 経験値が足りず、やり方に自信が持てない
  - 机上では理解するも、注意すべき点がわからない
- ✓ **アジャイルの良さに半信半疑の人**
  - やっぱり旧来のやり方に比して、アジャイルに不安がある
  - アジャイルは難しく、失敗するのではという不安がある
- ✓ **アジャイルをやりたいが、会社や関係者の理解が得られない人**
  - 会社としてなかなかGoがもらえない
  - ユーザー部門の合意が得られない

まずはアジャイルを正しく理解しよう　～よくある誤解～

アジャイルの是非を問う前に、まずはアジャイルを正しく理解してほしい。まずは、巷でよく言われる誤解から整理してみる。

- ✓ **アジャイルは安い？**
  → 安くなる時も高くなる時もあります。そもそもコストを目的に実施すべきものではありません。
- ✓ **アジャイルは必ず早い？**
  → 適した案件で正しく実施すれば、早くなることが多いです。
- ✓ **アジャイルは要件をいつまでも決めなくてよい？**
  → 要件の変更を取り込み易いですが、優先度を変更していることを理解する必要があります。
- ✓ **アジャイルは品質が悪い？**
  → アジャイルだから品質が良いとか悪いとかいうものではありません。
- ✓ **アジャイルは特別なスキルが必要？**
  → WFとは異なるスキルセットが求められますが、全員に特別なスキルが必要というものでもありません。

上記のような、基本的な誤解をされていることも多い。
そのような場合は、まずはアジャイルというものを正しく理解することをお勧めする。

→ （参考）IPAによる、アジャイルソフトウェア開発宣言の読み取り方
https://www.ipa.go.jp/jinzai/skill-standard/plus-it-ui/itssplus/ps6vr70000001i7c-att/000065601.pdf

アジャイルは、単なる開発手法ではない。アジャイルのマインドを持つことが大事！

まずはアジャイルを正しく理解しよう　～WFとの比較～

アジャイルをある程度理解できたら、次はWF（＝ウォーターフォール）との違いを整理しておく。WFについても正しく理解し、アジャイルとの違いも正しく見ていきたい。
下の表は、アジャイルを議論する際、WFと比べて懸念されていることの多い論点について、対比させてみたものである。総じて、単純にWFが優れているというものはなく、単に経験値が多いことからくる安心感によるものだけではないか。

	論点	アジャイルの場合	WFの場合	見解
コストの精緻化	アジャイルはWFと比べてコストが見積れないため、予算化が難しい？	概算の見積もりは可能。経験値を積むことで精度も上がるか。	WFの場合もあくまで概算でしかない。経験値とバッファで予算化に利用しているだけ。	どちらも本当は変わらないのではないか。
計画の精緻化	アジャイルはWFと比べて計画が立てられない？	経験値によるベロシティで大枠の計画化は可能。それ以上の精緻化は実施しない。	直近の工程以外も概要を計画化。段階的に詳細化していく。	実は詳細なレベルの計画はどちらもしていない。
作りたいものの実現度合い	WFの場合は最初に要件を定義するため、要件通りに作りやすい？	要件変更を取り入れ、優先度の高いものから作っていくため、当初作る予定のものも作らない場合がある。	最初に細かく定義すればするほど、実現される可能性が高まる。 一方、実際は使われない要件を作ってしまうことも多い。	必要なものから作るという意味では、WFは無駄なものも作られやすい。
スキルセット・体制	アジャイルの場合は、特殊なスキルセットや体制が必要となるのか？	完全な役割分担をするわけでなく、ユーザー部門の直接参画も必須に。	ある程度役割を決めて分担する分、縦割りになりやすい。	両者の違いを理解しておくべき。
品質	WFに比べてアジャイルは品質が悪いのではないか？	考え方が違うだけで品質が悪いというわけではない。要件齟齬リスクは確実にWFに比して良い。自動化により品質を高めることが多い。	全体整合を取って進めることに強みがあり、大規模に向いている。	WFのほうが経験値が多いだけではないか。

なぜ基幹系に浸透しないのか？　～そもそも基幹系とは？～

改めて、基幹系システムとはどういったものを指すのか。

- その会社のビジネスの根幹となる業務を担う重要なシステム
- システムが停止した場合、影響が広範囲にわたり、その企業の活動に大きな支障をきたすもの
  ⇒ **高い可用性、堅牢性、安定性（品質）が求められる**。
- 企業経営に必要な重要なデータを管理している。
  ⇒ **高いセキュリティが求められる**。

基幹系システムは、その会社の根幹業務を担うものであることから、多大な投資を継続して行っており、簡単に変更や代替ができるものではない。**新しいものへの取り組みの難しさ、代替システムへの変更の難しさ、機動力の無さ**、、等々につながっていると考える。

日本の伝統的なSIerモデル、古い開発形態で実施され、ベンダに丸投げされている場合も多い。

なぜ基幹系に浸透しないのか？　～基幹系に適用するメリットは？～

基幹系においても、アジャイルの場合はWFと比べて以下のようなメリットを享受できると考えられる。
そのため、アジャイルを採用したい基幹系開発も少なくないと考えられる。

	アジャイル開発のメリット	基幹系での享受（WF比）
1	仕様バグの早期摘出による品質の向上	・WFでは後半のユーザーテストで顕在化する要件バグが、ユーザーとのコミュニケーション、上流工程からの参画により早い段階で摘出可能
2	ドキュメントの簡素化、質の向上	・コミュニケーションやソースコードで理解することに重点を置くことで、結果的にドキュメントが簡素化 ・コミュニケーションの積み重ねによる質の向上
3	プロダクトの価値の向上（優先順位の最新化、要件変更に迅速に対応）	・開発期間が長い基幹系の場合は、要件の陳腐化が少なくなる（その時点で優先度の高いものを開発可能）
4	開発生産性の向上	・同じメンバーが継続的に開発に携わることで、学習効果により開発生産性は段階的に向上 ・WFに比べてアイドリング状態となる期間が少ない
5	サービスインまでの期間短縮 （優先度付けにより短期間でリリース）	・基幹系は密結合状態となっているシステムが多く、リリースに向けて相応のテストが必要となるケースが多いため、短縮効果を得るにはテスト自動化などの施策を並行して実施する必要がある。 一方で、独立した機能を改修するケースや疎結合化されたシステムに適用する場合は、短縮メリットを享受しやすい。
6	対面でのコミュニケーションによりコミュニケーションコストが低減	・要件定義などの手戻りコスト、リスクの減少 ・WFの場合に発生しがちなチーム間連携でのミス等のリスクを下げられる（ただし、規模が大きい大人数のプロジェクトでは難しい部分も）

なぜ基幹系に浸透しないのか？　～乗り越えないといけない壁は～

基幹系で取り入れる場合の、乗り越えないといけない壁にはどういうものがあるか？

① 体制確保が難しい
- 自社の経験者、有識者が少ないため？
- そもそもどういう体制が必要かわからない？
- アジャイルマインドの醸成が難しい？

② そもそもアジャイル開発が難しい
- 経験者、有識者が少ないため？
- 品質確保が難しいため？

③ どうしても大規模になってしまう
- 基幹系はもともと大規模なものが多く、サービス分割も難しい？

④ 経営陣への説明が難しい。会社の理解が得られない
- 予算取りをするための概算コスト、計画が見通せないため？
- 経営陣がアジャイルを正しく理解していないため？

課題①体制確保が難しい

そもそも、どういう体制が必要なのか。これまでの経験値がないと、なかなかチームビルディングに手も付けられない。そこで、ここでは以下の３つの観点から考察する。

- ✓ **アジャイルマインドの醸成**
  この資料の冒頭にも述べたように、アジャイルマインドの浸透が非常に重要である。アジャイルは魔法ではなく、成功するためには正しくアジャイルマインドを理解、浸透させる必要がある。

- ✓ **プロダクトオーナーの設置**
  アジャイルにおいてはプロダクトオーナー（PO）の位置づけが非常に重要である。各事業会社ごとにIT部門と業務部門の関係やベンダとの関係も異なってくるが、誰がPOを担うかについて考えてみる。

- ✓ **スクラムチームの組成と進化**
  アジャイルの場合は基本的に内製化することが望ましいが、いきなり有識者や経験者を揃えることは難しく、当初はベンダの力を借りることが現実的と思われる。そこで、当初の姿のポイントと、その後の目指すべき姿について考える。

アジャイルマインドの醸成

アジャイルを成功させるためには、開発チームだけでなく、その会社全体がアジャイルを正しく理解して進める必要がある。アジャイルのマインドセットについて、ステークホルダーの理解が不十分だと、失敗リスクが高まることとなる。
基幹系システムはステークホルダーが多くなる傾向があるため、**如何にステークホルダーのアジャイルマインドを浸透させるか**が、ひいては企業全体のマインドの変化となり、アジャイルプロジェクトの成功につながる。

持ってほしいマインド（ToBe）	現状（AsIs）	改善の対策（例）	さらなる施策
プロダクトの価値の最大化を優先する	与えられたタスクのみ実施一度決めたことは必要性を再評価しない。	トップダウンで顧客価値にフォーカスしたKPIを設定する	アジャイル開発を支援するための中立的・客観的な組織の立ち上げ（第三者組織）
チームとして活動する/プロセスを共有する	決められたプロセス、役割を全うする。組織・役割の縦割りも上記を助長。	組織の構造変化が発生してしまう可能性を許容する	
		ビジネス／開発に偏った立場ではなく、フラットなメンバー（第三者）が客観的に判断するような体制をつくる	
チームを継続的に改善し続ける	ルールの縛りが強く、変更のハードルが高い。改善活動の価値の理解に乏しい。	改善活動を定量化し、ビジネス側（ステークホルダー）のメリットを伝える	
ドキュメントより対話を重視	経営層への報告、ビジネスとの課題検討等、ドキュメントを都度必要とする。	対話によるコミュニケーションの時間を確保してもらう	

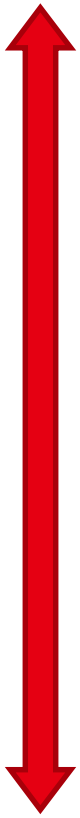
プロダクトオーナーの設置

アジャイル開発においてPOは非常に重要である。POへの権限移譲がポイントとなるため自社での設置が最低限必要と考える。そのうえで、POをどの部門が担うかについていくつかのパターンが考えられる。各社の状況に応じて最適なパターンは変わりうるが、いずれにしてもシステム技術要素を含めた全体の要件について適切な優先順位を決められるかがポイント。

	PO	メリット	デメリット
1	**ビジネス部門**	• POとステークホルダーの関係性が構築できているため、ビジネス要件の決定等が早くできる	• 非機能要件等のシステム技術面で、開発チームがPOをかなりサポートする必要ある。（技術要素を正しく理解しないと、要件の優先順位が正しく決められない） • POを担うビジネス部門の負担が高く、PO人材の育成も難しい
2	**情シス部門**	• 技術的な課題への理解が得られやすい。	• ビジネス部門との距離が遠く、ビジネス部門の協力を得られない ⇒社内で「情シス部門が勝手にやってるだけ」となる可能性 • POとステークホルダーとの関係性が希薄で要件調整が難航 • 真のPOがチーム外に存在することとなる可能性
3	**ビジネス部門と情シス部門との共同**	• 技術要素を必要とする非機能要件含め、全ての要件の優先順位を適切に決めることが可能。	• POの意思決定を統一できるかが課題 （信頼関係の構築）
ー	**ベンダ**		• ステークホルダーをまとめきれない恐れ • 優先順位を付けられない恐れ • 実質的な意思決定権がない恐れ • 正しい要件を理解できない恐れ

スクラムチームの組成（内製　vs　外製）

アジャイルの場合は基本的に内製化することが望ましいが、有識者や経験者を揃えるのが難しいため、ベンダを活用することが考えられる。以下では、ベンダを活用する度合いについて、スクラムマスター（SM）を含めた開発チームの完全内製から完全ベンダ依存までのそれぞれのメリデメを整理する。



		PO	SM	開発チーム	メリット	デメリット
内製型	1	**社員**	**社員**	**全て社員**	• 業務システムの知見が社内に内在するので、手の内化が推進しやすい • 契約や受発注がないのでリードタイムが短い	• アプリの開発要求は一定じゃないためリソースコントロールが難 • いきなり体制確保するリソースが難 • アジャイルスキルのリソースが難
	2	**社員**	**社員**	**大半が社員（一部ベンダ）**	• 業務システムの知見が社内に内在するので、手の内化が推進しやすい • ベンダリソースを流動的に扱うことが可能なので、リソースの柔軟性あり	• アプリの開発要求は一定じゃないためリソースコントロールが難 • いきなり体制確保するリソースが難 • アジャイルスキルのリソースが難
ベンダ依存型	3	**社員**	**ベンダ**	**一部が社員（大半ベンダ）**	• 内部リソースが低減できるので立ち上げやすい（リソースの柔軟性） • アジャイルに習熟してるベンダの利用	• ブラックボックス懸念 • ベンダ都合での人の流出
	4	**社員**	**ベンダ**	**全てベンダ**	• 同上 • さらに自社に経験が少なくても可能	• 同上 • さらにブラックボックス懸念大
	5	**ベンダ**	**ベンダ**	**全てベンダ**	• 自社に全く経験がなくても可能	• アジャイルをする意味あるの？

スクラムチームの進化

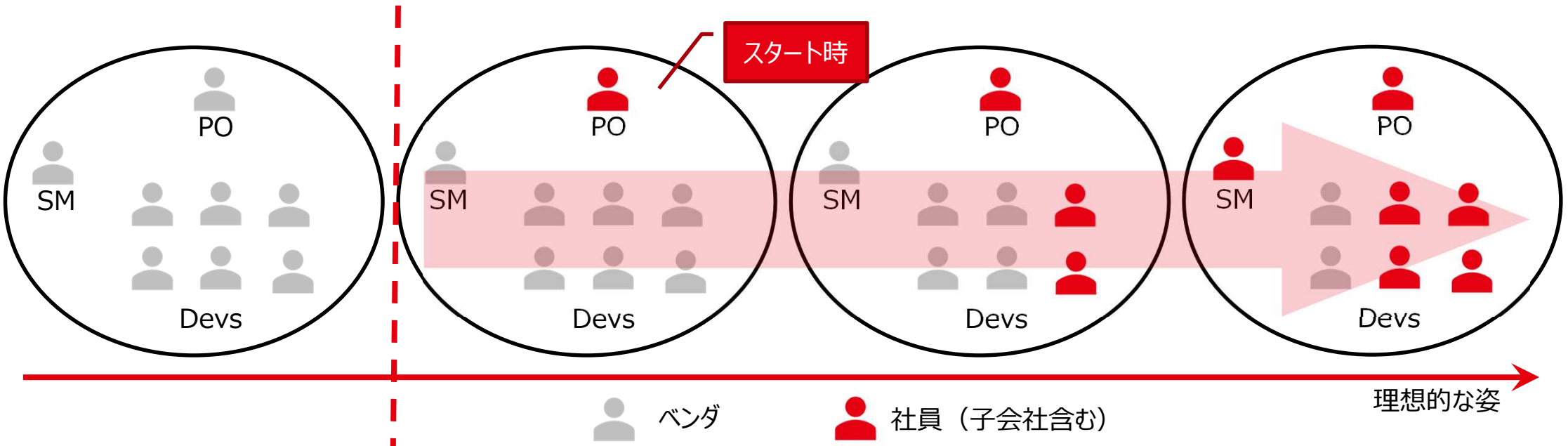
●経験者や有識者の不足から当初はベンダの力を借りることが想定されるが、その場合であっても、**POは自社で担うべき**である。それは、開発の優先順位付けの判断において、自社でスピーディに実施する必要があるためである。（ただし、フェイクPOとならないようにも注意）

●そのうえで、経験値を積んでいき、理想的な姿（内製化）に近づけていくことが望ましい。

●また、開発者の注意点として、**設計～リリースまでを行うスキルや、CI/CDの理解を必要**とする。これは、工程作業ごとに役割分担を可能とするWFとの大きな違いからきている。

●その他の注意点として、ベンダと契約する際には**準委任契約とすることが望ましい**。それは、目的物の成果物責任を請負う契約はアジャイルにそぐわないと考えるからである。

（参考）IPAモデル契約　https://www.ipa.go.jp/digital/model/agile20200331.html



課題②そもそもアジャイル開発が難しい

経験者や有識者が少ないこともあり、アジャイル開発そのものが難しい。基幹系として品質が重要視される中、プロジェクトそのものが難しいと採用に躊躇してしまう。
そこで、ここでは大規模な基幹系のアジャイル適用事例を紹介する。参考にしつつ、自社の場合の適切なものを考えてもらいたい。

- ✓ **開発工程のハイブリッド（X社の例）**
  - この例は、基本的にはアジャイル開発を採用するものの、開発工程として、WFを意識した工程を加えたものである。
  - アジャイルの良さも一部失われるが、従来型の開発しか経験のない人にも入り易いのではないか。

- ✓ **システムのハイブリッド（Y社の例）**
  - この例は、基幹系本体はWFで開発しつつ、新商品開発等の切り出した部分をアジャイルで開発し、PJ内でWFとアジャイルを並列させた事例である。
  - 基幹系部分も高速WFと呼ばれるような形を追求しており、やや高度な事例か。

開発工程のハイブリッド事例（X社の事例）

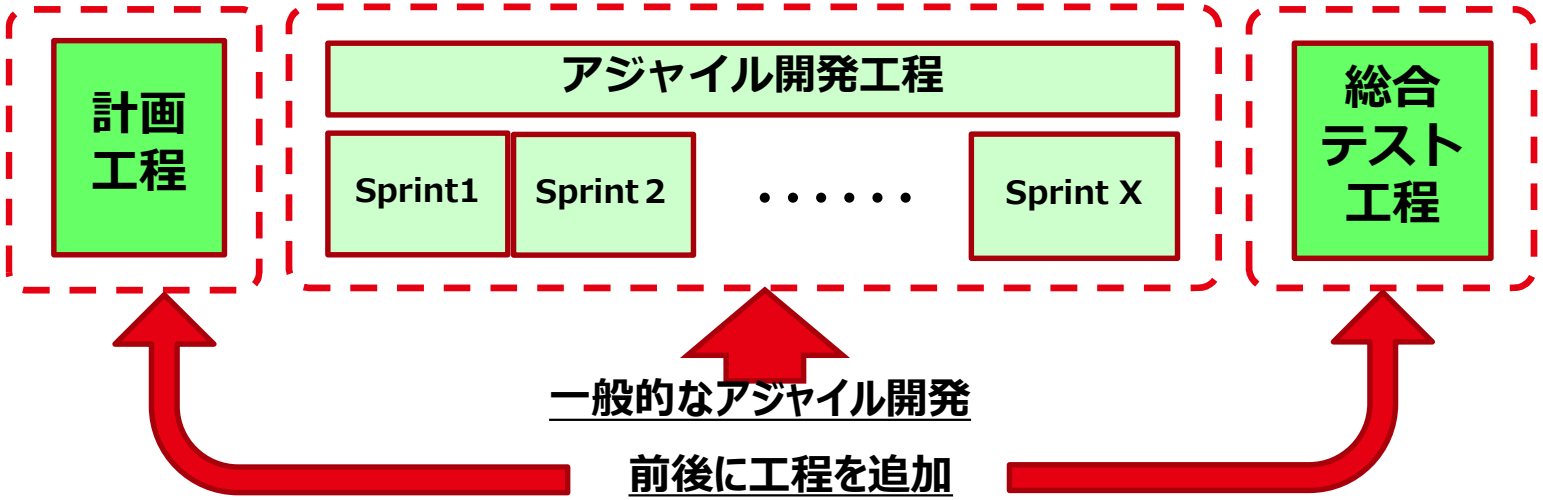
基本的にはアジャイル開発を採用するものの、WFを意識した工程を加えた、開発工程としてのハイブリッド事例を紹介する。

●基幹系システムにアジャイルを適用する場合、特に経験値が少ない場合は一般的なアジャイルほどの自由度で開発することは難しい。そのため、**事前にある程度の要件や計画を検討する工程**を設定。
- 開発工程がぶれすぎないように要件の概要を決めるもので、一般的な**WFの要件定義工程レベルには定めない**。
- 全体のデザイン、プロジェクト計画を策定する。

●基幹系システムへの影響を考えると、次々とリリースを繰り返すようなことも難しいことから、アジャイル開発後に、**全体を通した総合テストを実施する工程**を設定。
- 全体的な整合性確認含め、基幹系に耐えうる品質を確保。

●逆に、できるだけアジャイルの利点を失くさないためにも、アジャイル開発工程は一般的なアジャイルに。

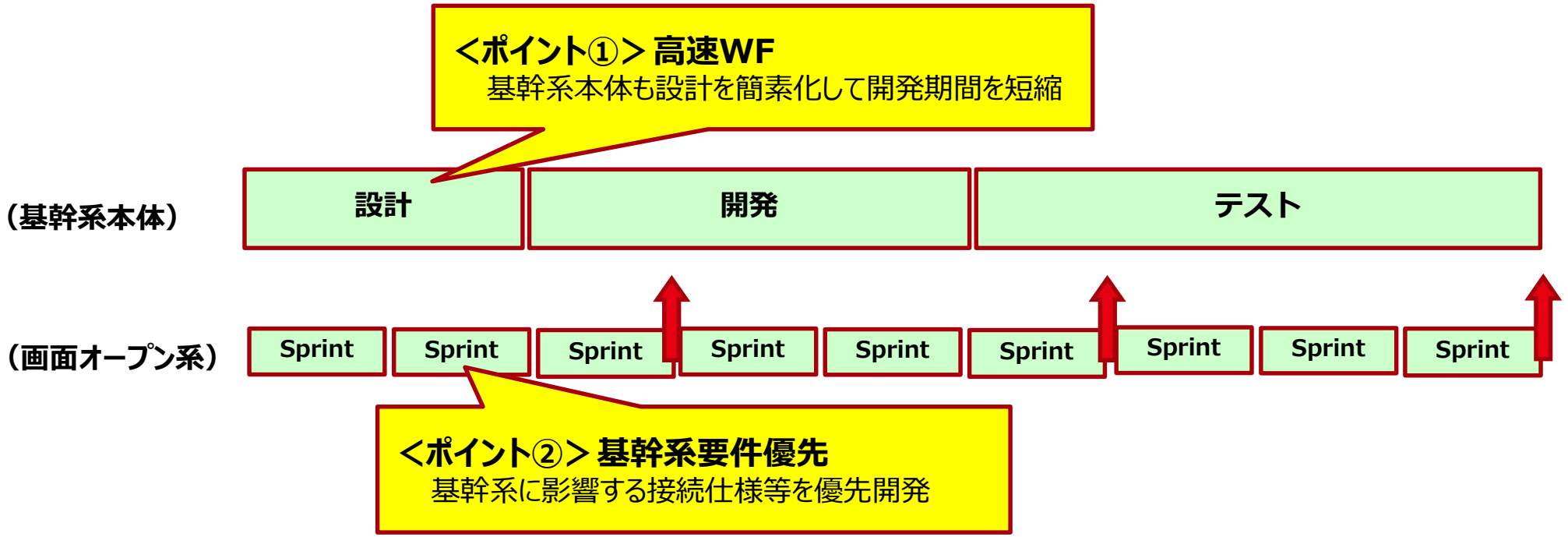
➡ **アジャイル開発工程の前後に工程を加えた形**のマスタースケジュールを設定。



システムのハイブリッド事例（Y社の事例）

大きなPJの中で、システムごとにWFとアジャイルを並列させて開発し、システム単位でハイブリッドに開発した事例を紹介する。

- 基幹系ホスト部分をWFで、新商品の画面を中心としたオープン系部分をアジャイルで実施した、ハイブリッド開発。
- 基幹系についても高速WFで開発し、全体的な期間短縮を図る。
  - アジャイルのように、ユーザー部門とIT部門が一体となったチームで開発。
  - 一部設計書の作成を後回しにする等、スピードを優先した開発。
- アジャイル開発側も、基幹系に影響する接続仕様等を優先的に開発。
  - 基幹系の手戻りを防ぐ。



課題③どうしても大規模になってしまう

基幹系システムの開発は大規模となることが多いが、基幹系に関わらず、アジャイルの場合は一定規模を超えるとかなり難易度が上がると言われている。どうすべきか？

- ✓ **アジャイルを始めるには、まず小さく始めて、少しずつ広げるのが良い**
  - 基幹系システムの全部をいきなりアジャイルで作り直すのはナンセンス。
  - サービス分割することが難しい場合も多いので、まずは小さくサブシステムから。
  - アジャイル開発で作る部分を増やしていき、少しずつサービスを切り出していく。
- ✓ **WFとアジャイルとを切り替えられるようにしておくのがベストか。**
  - 新規構築時はWFで開発し、維持保守フェーズに入ってから徐々にアジャイルという選択肢も。
  - そのためには自動化が必須か。
- ✓ **それでも大規模な場合の例**
  - 大規模アジャイルは難しいと言われているが、いくつかの手法は既に存在している。
  - 本書の読者がいきなり採用することは推奨しないが、一般的な大規模アジャイル手法を紹介。

（参考）一般的な大規模アジャイル手法

	主な大規模アジャイル手法	特徴
1	Scaled Agile Framework（SAFe）	◻ リーンプロダクト開発、アジャイル開発、システム思考の原則に基づいて、企業や事業部レベルでリーンとアジャイルを実現する際のプラクティス集 ◻ 組織全体での取り組みとして、ポートフォリオ管理という概念が追加されている ◻ スクラム、XP（eXtreme Programming）、デザイン思考、DevOpsなどさまざまな概念を取り込んでいる ◻ 世界でも日本でも採用率が高い
2	Scrum@Scale/ Scrum of Scrums	◻ 組織運営のフレームワークとなりプロダクトそのものよりも、組織にフォーカスしている ◻ プロダクトオーナー用のサイクルと、スクラムマスター用のサイクルがある ◻ スクラムマスターサイクルでは、スクラムオブスクラムの形で、個々のスクラムチームでの課題を1段上のレイヤーで収集し、組織的に解決する。最上位のチームをEAT（エグゼクティブアクションチーム）と呼ぶ ◻ プロダクトオーナーサイクルでは、複数チームのプロダクトオーナーが集まってプロダクトオーナーチームを形成し、それを司るCPO（チーフプロダクトオーナー）がいる
3	Large Scale Scrum（LeSS）	◻ プロダクトバックログは全体で1つ、それぞれのスプリントでインクリメントを作るという点も通常のScrumチームのものと違いはない ◻ スプリントバックログは開発チーム単位で存在し、各チームごとに作る ◻ 全チームは同じ日にイベントを行い、スプリント期間も同じ ◻ プロダクトオーナーは1人、スクラムマスターは1人で1～3チームの面倒を見る ◻ 完成の定義は全チームで共通となる

アジャイルをスケーリングする手法についていくつかのフレームワークの採用率
（参考：https://info.digital.ai/rs/981-LQX-968/images/AR-SA-2022-16th-Annual-State-Of-Agile-Report.pdf）

- Scaled Agile Framework (SAFe) : 53%
- Scrum@Scale/Scrum of Scrums : 28%
- Lean Management : 8%
- Agile Portfolio Management (APM) : 7%
- Spotify Model : 7%
- Enterprise Scrum : 6%
- Large Scale Scrum (LeSS) : 6%
- Disciplined Agile (DA) : 3%
- Nexus : 3%
- Recipes for Agile Governance in the Enterprise (RAGE) : 1%
- Some other framework : 8%
- Not at all sure :15%

課題④経営陣への説明が難しい、会社の理解が得られない

実際にアジャイルをしたくても、決裁する経営陣の理解が得られないと、会社として実行する許可が下りない。計画が立てられないからなのか？経営陣がアジャイルを正しく理解していないからなのか？

- ✓ **アジャイルでの概算計画立案にチャレンジする！**
  - ー 予算取りのための概算コストが見通せない、計画が立てられない、という声。果たして本当か？
  - ー 先述の通り、計画が立てれないことは実はアジャイルであることと本来は関係がない。（WFの場合も見通せている気になっているだけ）
  - ー であれば、アジャイルでの概算計画立案レベルの経験を増やすしかない。（WFも実はいい加減）
  - ー 初めて取り組む場合は、本書の他の事例を参考にしつつ、やはり小さく始めるのが妥当ではないか。そのうえで経験値を積み、フィージブルな計画を立てられるように洗練していくべきではないか。

- ✓ **組織のアジャイルマインドを醸成する！**
  - ー かなりハードルは高いが、組織的にアジャイルマインドを醸成し、トップダウン的に始めることができるならベストである。 社長自らが率先してアジャイルを推進する事例もあり、それができれば最も話が早い。
  - ー それが無理であっても、アジャイルを正しく理解してもらうように努める必要がある。本書の冒頭のアジャイルの誤解等を丁寧に説明し、正しく理解してもらうところから始める必要がある。

（おわりに）

アジャイル開発が話題になり始めてから久しいが、日本ではなかなか定着しないように思われる。いくつかの要因が考えられるが、それは、既存のやり方が染みついている人たちがアジャイルをやったことがない、やれない、という面も大きいのではないか。

アジャイルを推奨する人は、アジャイルの良さを強調するのみで、既存のやり方が染みついている人にはなかなかハードルが高く感じられるのではないか。

今回、我々は、既存のやり方が染みついている人がアジャイルに取り掛かれない理由を突き詰めてみたつもりである。アジャイルしか経験のない人、既存の開発しか経験のない人、両方の経験がある人等で構成し、既存の経験しかない人がアジャイルに踏み出せない理由について、真剣に議論してきた。世の中には、アジャイルに初めて取り組むための書籍も多数あるが、それらとは一風変わったものに仕上げたつもりである。

アジャイルをやりたいのになかなか踏み出せない人たちの、少しでも参考になれば幸いである。

JUASラボ　無料開催！ どなたでも参加可能

基幹系システムへのアジャイル活用について

～JUAS基幹系システムアジャイル適用研究プロジェクト2022年度成果と今後の展開を考える

5月24日（水）15:00-17:00

会場：JUAS２F会議室　＋　オンライン（Zoom）
※参加方法は申込時にお選びください

参加費：無料（事前登録）

参加方法：下記URLからお申込みください
https://juasseminar.jp/seminars/view/4723004

参加される方へのお願い：
Jフェスの発表を当日までにご欄ください
（約30分、JUAS Web-channelで公開予定）

プロジェクトメンバーがファシリテートします
どんどん疑問や提案をぶつけてください！

JUAS　基幹系システムアジャイル適用研究プロジェクトは、
「基幹系システムへの適用に対する壁をなくし、
　日本でのアジャイルの定着に貢献したい。」と考え研究を進めてきました。

2022年度成果報告の内容について、より広く、深く理解いただくため、また、より活動の幅を広げ、実践につなげたいと考え、多くの方のご意見をいただきたく、ラボを開催します。

- 研究内容をもっと知りたい方
- 研究内容にご質問やご意見がある方
- 基幹系システムにアジャイル適用をしたい（している）が課題がある方
- 一緒に研究を深堀りしたい方
- プロジェクトメンバーと直接話をしてみたい方

・・など、研究に興味を持っていただいた方、ご協力いただける方のご参加をお待ちしています。