「アジャイル開発への道案内」
～経済競争力を得るための方法論～

2019年11月30日

小原　由紀夫, PMP, ｹｲﾃﾞﾝｽﾏﾈｼﾞﾒﾝﾄ社認定講師
（富士通株式会社）

講師紹介

小原　由紀夫（コハラ　ユキオ）

Kohara.yukio@jp.fujitsu.com

富士通シニア・アジャイリスト

プロフェッショナルサービス本部

第二プロフェッショナルサービス事業部　担当部長

富士通株式会社

1983年	電機・自動車の工場基幹システム構築の担当・ＰＭ
2000年	プロジェクトマネジャー育成
2005年	ケイデンスマネジメント社(2009PMI Best Provider)講師
2015年	(株)富士通アドバンストエンジニアリング　人材開発室長
2017年	「アジャイル開発への道案内」(日本PM協会編)共著
2018年	富士通におけるアジャイル関連ビジネスの立上げ･推進 アジャイルコンサルティング、人材育成トレーニング、アジャイルコーチ
2019年	「ＩＴサービスのためのアジャイル」(日本ＰＭ協会編)共著

目次

1．はじめに

　アジャイル開発方式は、今や米国におけるソフトウェア開発方式の大勢を占めるようになっている。かつての「ウォーターフォール対アジャイル」論争は影を潜め、議論の中心は如何にしてアジャイル開発の適用範囲を拡大していくか、に移っている。

　一方、日本でのアジャイル開発の普及は遅々として進展していない。日本のソフトウェア業界は何を恐れているのであろうか？発表者達は、日本のアジャイル開発普及を加速すべく、次の書籍をまとめた。

「アジャイル開発への道案内」　片岡雅憲・小原由紀夫・光藤昭男
共著　日本プロジェクトマネージメント協会（PMAJ）編
2017　近代科学社

　この講演では、当書籍の紹介を通じて、アジャイル開発全般を概説するとともに、日本でのアジャイル開発に伴う「リスク」あるいは「心配ごと」について考えたい。
（注）PMAJでは、当書籍に基づく**アジャイル開発教育**を計画している。

（調査）アジャイル適用率

皆さんの組織で、アジャイルを適用する割合はどのくらいですか？

A　常に
B　頻繁に
C　時々
D　まれに
E　使用しない

お隣の方とアジャイルの適用する割合に関して話してください。
（２分）

（参考）デジタル変革に関する予言

- 2011年 米国の著名な投資家であるマーク・アンドリーセンは「Why Software Is Eating The World」の手記を発表
- ソフトウェアを中心に駆動する企業が台頭し始めており、今後、あらゆる産業を飲み込んでいくと予言した


THE WALL STREET JOU

Home World U.S. Politics Economy Business Tech Market

ESSAY

Why Software Is Eating The World

By Marc Andreessen

August 20, 2011

This week, Hewlett-Packard (where I am on the board) announced that it is explo
jettisoning its struggling PC business in favor of investing more heavily in softwa
it sees better potential for growth. Meanwhile, Google plans to buy up the cellpho
handset maker Motorola Mobility. Both moves surprised the tech world. But both
are also in line with a trend I've observed, one that makes me optimistic about the
growth of the American and world economies, despite the recent turmoil in the st
market.

In...

RELATED VIDEO

2011.8.20 ウォールストリートジャーナル

世の中で一番大きな書店　Amazon

世界で最高・最新の映画制作会社　Pixar

大規模ダイレクトマーケティングのプラットフォーム　Google

最も急速に成長している通信会社　Skype

最も急速に成長しているリクルーティング会社

LinkedIn

世界で市場を制圧している音楽会社　AppleのiTunes

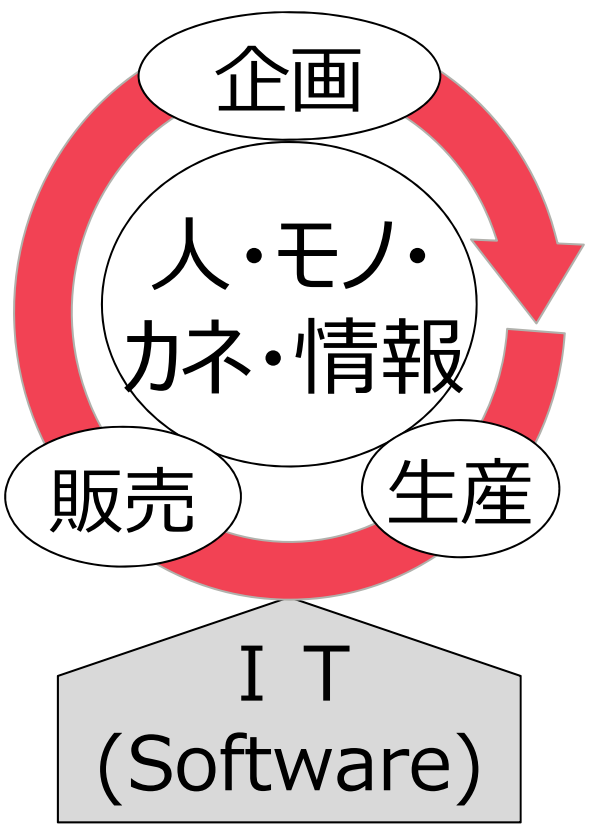
現実世界でも、WalMartやFeDexはロジスティックスと配送機能を増強するためにソフトウェアの力を使っている

ソフトウェアはあらゆる世界に入り込んで、世界を変えようとしている

（参考）ITの企業内の位置付けの変化

＜これまでの経営＞

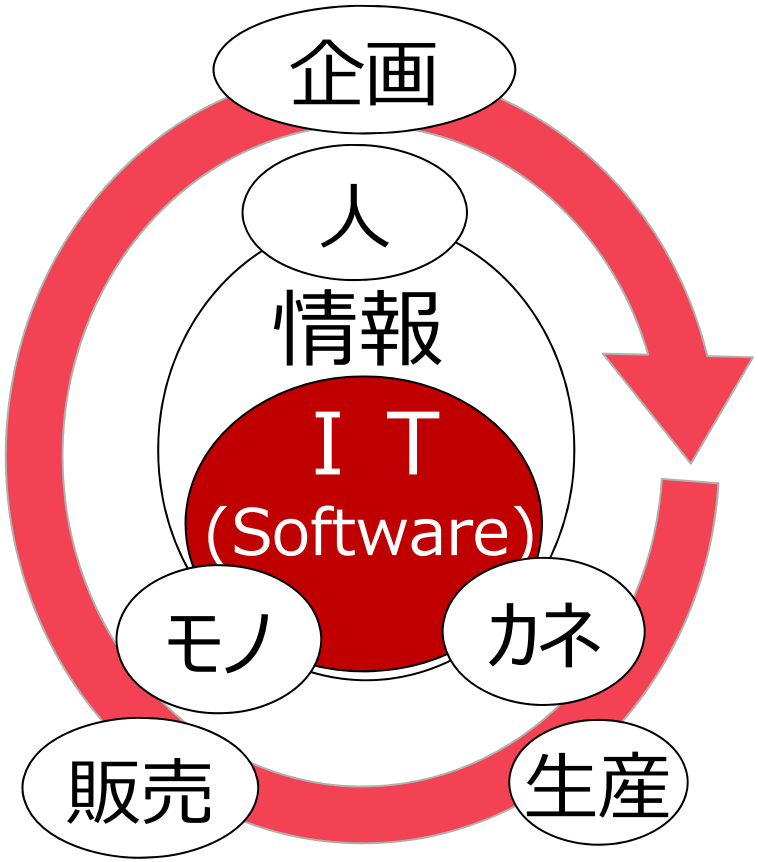
ITは活動効率化の道具



デジタル変革

＜これからの経営＞

ITが価値創造の中心



（参考）DXレポート「2025年の崖」

経済産業省　http://www.meti.go.jp/shingikai/mono_info_service/digital_transformation/20180907_report.html

注）ＤＸ＝デジタルトランスフォーメーション

2025年の崖の脅威：既存システムがブラックボックス化＆データ活用ができない

■構築後２１年以上の基幹システム：２割(現在)⇒６割 (2025年)

（技術的負債(短期的観点で長期的な維持費が高騰)により維持管理費がＩＴ予算の9割に）

➡ 2025年以降、12兆円／年の損失

2025年の崖の好機：ブラックボックス解消＆本格的なＤＸを実行し、“デジタル企業”へ

➡ 2030年、実質ＧＤＰ130兆円の押し上げ

対策案（抜粋）

1．「見える化」指標、中立的な診断スキームの構築　**経営者自ら**が適切にガバナンス

2．「DX推進システムガイドライン」の策定　**経営者、取締役会、株主**等のチェック・リスト

3．DX実現に向けたITシステム構築におけるコスト・リスク低減のための対応策　マイクロサービス等の活用

4．ユーザ企業・ベンダー企業間の新たな関係　**アジャイル開発契約**ガイドライン

5．DX人材の育成・確保　**アジャイル開発の実践**による**事業部門人材のIT人材化**

➡経営層とアジャイルが焦点

（参考）電子政府への取り組み

1. デジタルガバメント実行計画
   2018年7月20日　改定
   （デジタル・ガバメント閣僚会議決定）
   「デジタル・ガバメント推進方針」に組み込まれた
   サービスデザイン思考を具体化した、
   「サービス設計１２箇条」を示している。
   https://cio.go.jp/node/2422

＜サービス 設計 12 箇条＞
第１条 利用者のニーズから出発する
第２条 事実を詳細に把握する
第３条 エンドツーで考える
第４条 全ての関係者に気を配る
第５条 サービスはシンプルにする
第６条 デジタル技術を活用 し、サービスの価値を高める
第７条 利用者の日常体験に溶け込む
第８条 自分で作りすぎない
第９条 オープンにサービスを作る
第１０条 何度も繰り返す
第１１条 一遍にやらず、一貫してやる
第１２条 システムではなくサービスを作る

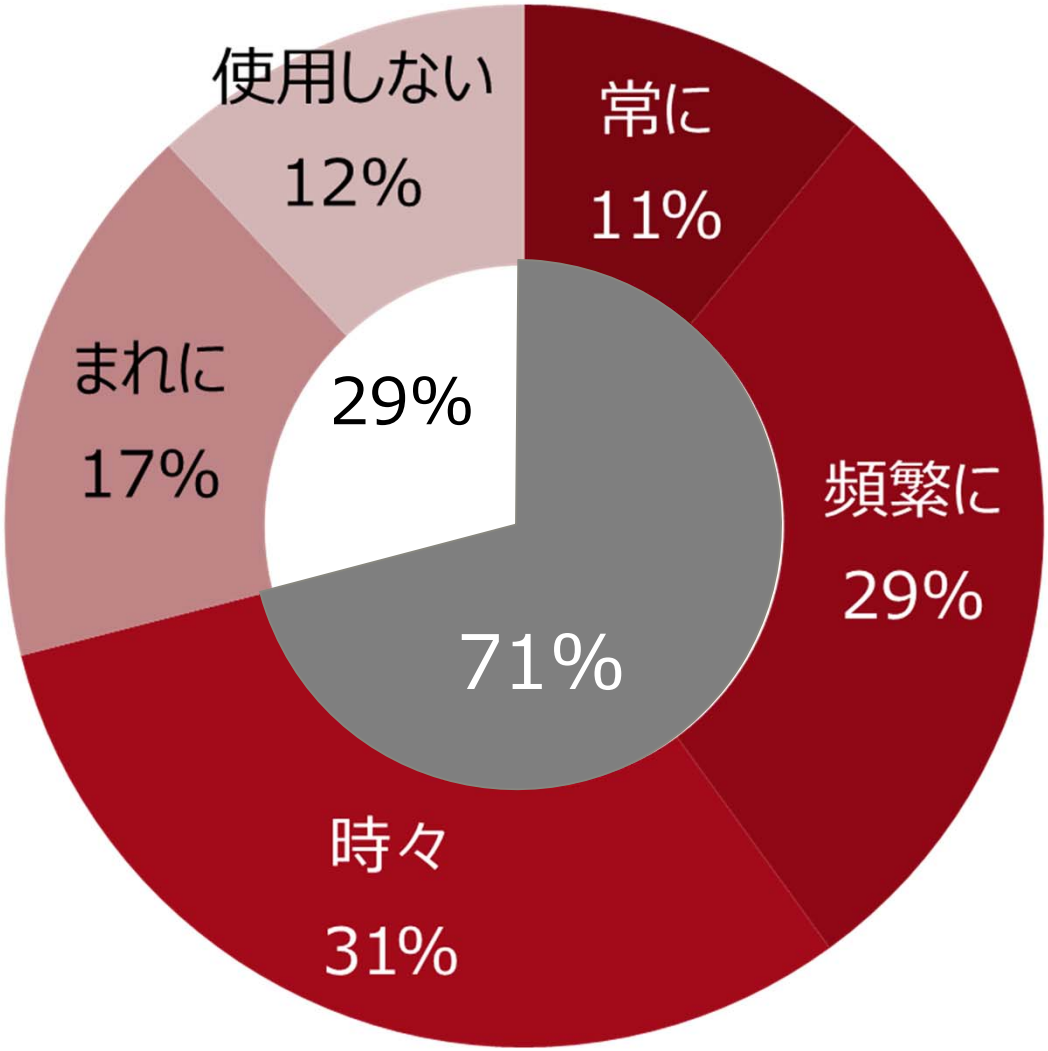
2. 米国連邦政府のＩＴプロジェクトでアジャイルまたはイテレイティブの割合
   2011年　10%　⇒　2017年　80%

出典：PMNetwork by PMI
2018 February
http://www.pmnetwork-digital.com/pmnetwork/february_2018/MobilePagedReplica.action?pm=1&folio=46#pg48

3. アジャイル開発を推奨するためのガイドライン
   米国、英国、オーストラリアが採用。
   シンガポール、ブラジルが追従

（参考）アジャイル開発の適用率

グローバルでは、もはやアジャイル開発が主流



出典：PMI 第9回グローバルPM調査より

2.1　アジャイル開発方法論の誕生

(1)アジャイル開発とは

　1990年代の後半に、従来のプロセス重視の重い管理手法に対するアンチテーゼとして、軽量プロセスを提唱・実践する人が出始めた。

　アジャイルとはこのような軽量級のソフトウェア開発方法論の総称である。そして、それらのソフトウェア開発方法論に共通する考え方や哲学を指す言葉でもある。

　したがって、アジャイルに関する厳密な定義はなく、ソフトウェア開発方法論の大きな枠を指すものといえる。

(2)アジャイルマニフェスト

　アジャイル開発方法論は多様であり、一義的に定義することは困難な状況にあった。そこで、関連する人達が、2001年2月にアメリカ・ユタ州に会し、それぞれの手法の共通部分を見出すべく、議論した。 そして、その結果として合意出来たことを、

　　アジャイルソフトウェア開発宣言(アジャイルマニフェスト)

としてまとめて発表した。

　アジャイルマニフェストを次のページに示す。

2. アジャイル開発の概要

(2)アジャイルマニフェスト(続き)

私たちは、ソフトウェア開発の実践あるいは実践を手助けをする活動を通じて、よりよい開発方法を見つけだそうとしている。この活動を通して、私たちは以下の価値に至った。

プロセスやツールよりも 個人と対話 を、
包括的なドキュメントよりも 動くソフトウェア を、
契約交渉よりも 顧客との協調 を、
計画に従うことよりも 変化への対応 を、

価値とする。すなわち、左記のことがらに価値があることを認めながらも、私たちは右記のことがらにより価値をおく。

Kent Beck	James Grenning	Robert C. Martin
Mike Beedle	Jim Highsmith	Steve Mellor
Arie van Bennekum	Andrew Hunt	Ken Schwaber
Alistair Cockburn	Ron Jeffries	Jeff Sutherland
Ward Cunningham	Jon Kern	Dave Thomas
Martin Fowler	Brian Marick	

（参考）アジャイル宣言の背後にある原則

＜アジャイル開発を一言で＞

■戦略的（欧米＞日本）

戦略的な納期を自ら設定する

■持続的（グローバル共通）

UVCA環境変化に常に対応し続ける

■緊急(トラブル)対応

(欧米＜日本)

業務上のデットラインまでに提供

⇒チーム一丸で、やるべきことをやる。

⇒優先順位を付けて対応する。

⇒提供できないものについて利用者と協議

＜アジャイル宣言の背後にある原則＞

- 原則01:顧客の満足を求め続ける
- 原則02:要求の本質を見抜き、変更を前向きに
- 原則03:成果物を2-3週間でリリースし続ける
- 原則04:全員で共通の目標に向かう
- 原則05:人の意欲は信頼から
- 原則06:顧客も開発チームも直接対話で
- 原則07:進捗も品質も現物で
- 原則08:一定のペースでプロジェクトにリズムを
- 原則09:よい技術、よい設計、よい品質の追求
- 原則10:ムダ=価値を生まない、を探してやめる
- 原則11:よいモノはよいチームから
- 原則12:自分たちのやり方を毎週、調整する

参照：「アジャイルソフトウェア開発宣言の読みとき方2018」、
IPA https://www.ipa.go.jp/files/000065601.pdf

2. アジャイル開発の概要

2.2 代表的なアジャイル開発技法

表2.1 代表的なアジャイル開発技法

技法名称	発案者	概要
XP(eXtreme Programming)	Ward Cunningham, Kent Beck, Ron Jeffries	比較的少人数の開発に適用。5つの価値と12の具体的プラクティスを定義
Scrum（スクラム）	Ken Schwaber, Jeff Sutherland, Mike Beedle	プロジェクト管理技術が中心。開発途中で、顧客は要求を変更可、素早い対応のためのチーム能力最大化が主テーマ
Crystal	Alistair Cockburn	チーム内のコミュニケーション重視、高頻度のリリース、自動テスト、構成管理
ASD(Adaptive Software Development)	Jim Highsmith	開発環境、開発条件は常に変化する、それに適応(Adapt)していく開発
FDD(Feature Driven Development)	Jeff De Luca, Peter Coad Stephen R. Palmer, John M. Felsing	Featureとは顧客価値に基づく機能性、シンガポール銀行での開発経験、Object Modeling技法が基礎
DSDM(Dynamic Systems Development Method)	DSDM Consortium	DSDMは、British Air, AMEX, Oracle等によるコンソーシアム、固定された予算と工期の範囲で、優先付けされた機能を開発

3. アジャイル開発の特徴

3.1 アジャイル開発とウォーターフォール開発の比較

表3.1 ソフトウェアエンジニアの理解と利用のレベル

(注)Cockburnが提案したものに、Boehm等が改定を加えたもの

レベル	特徴
3	先例のない新しい状況に適合するために、(ルールを破ってでも)手法を改定することができる。
2	先例のある新しい状況に適合するために、手法をカスタマイズできる。
1A	トレーニングを受ければ、手法の手順のうち自由裁量の部分を遂行できる(例えば、インクレメントに応じた開発ストーリーの調整、パターンの開発、リファクラリング、パッケージの活用・統合等)。経験を積めばレベル2になれる。
1B	トレーニングを受ければ、手法の手順のうち手続き的な部分を遂行できる(例えば、単純なメソッドのコーディング、簡単なリファクタリング、コーディング標準や構成管理基準への準拠、テストの実行等)。経験を積めば、レベル1Aになれる。
-1	技術的スキルは持ち合わせているかもしれないが、協調したり共通の手法に従ったりできない(あるいはしたがらない)。

3. アジャイル開発の特徴

表3.2　アジャイル開発とウォーターフォール開発を特徴つける5つの主要因

要因	アジャイル	ウォーターフォール
規模	小規模な製品、チーム向け。暗黙知に依存しているため大規模化は困難。	大規模な製品、チーム向け。小さなプロジェクト用に切り詰めるのは困難。
重要度	安全性が重視される製品への適用例はない。設計がシンプルで文書が少ないことが前提。	重要度の高い製品向けに進化した手法である。重要度の高くない製品用に切り詰めるのは困難。
変化の度合い	変化の度合いが高い環境で、設計をシンプルにして継続的にリファクタリングしていく。	安定した環境で事前の大きな設計(Big Design Up Front)を実施する。不安定な環境ではやり直しのコストが高くつく。
人	レベル2,3の熟練者が開発期間を通じて十分な割合で必要。レベル1Bのアジャイルに慣れていない人を使うことによるリスクは高い。	プロジェクトの初期段階ではレベル2,3の熟練者が必須。後期の段階では、レベル1Bの人も使える。
文化	高い自由度を持つことで、快適で権限が与えられていると感じられる文化で繁栄する。（カオスにおける繁栄）。	明確な方針と手続きで自分の役割が定義されていることを感じられる文化で繁栄する。（秩序における繁栄）。

3. アジャイル開発の特徴

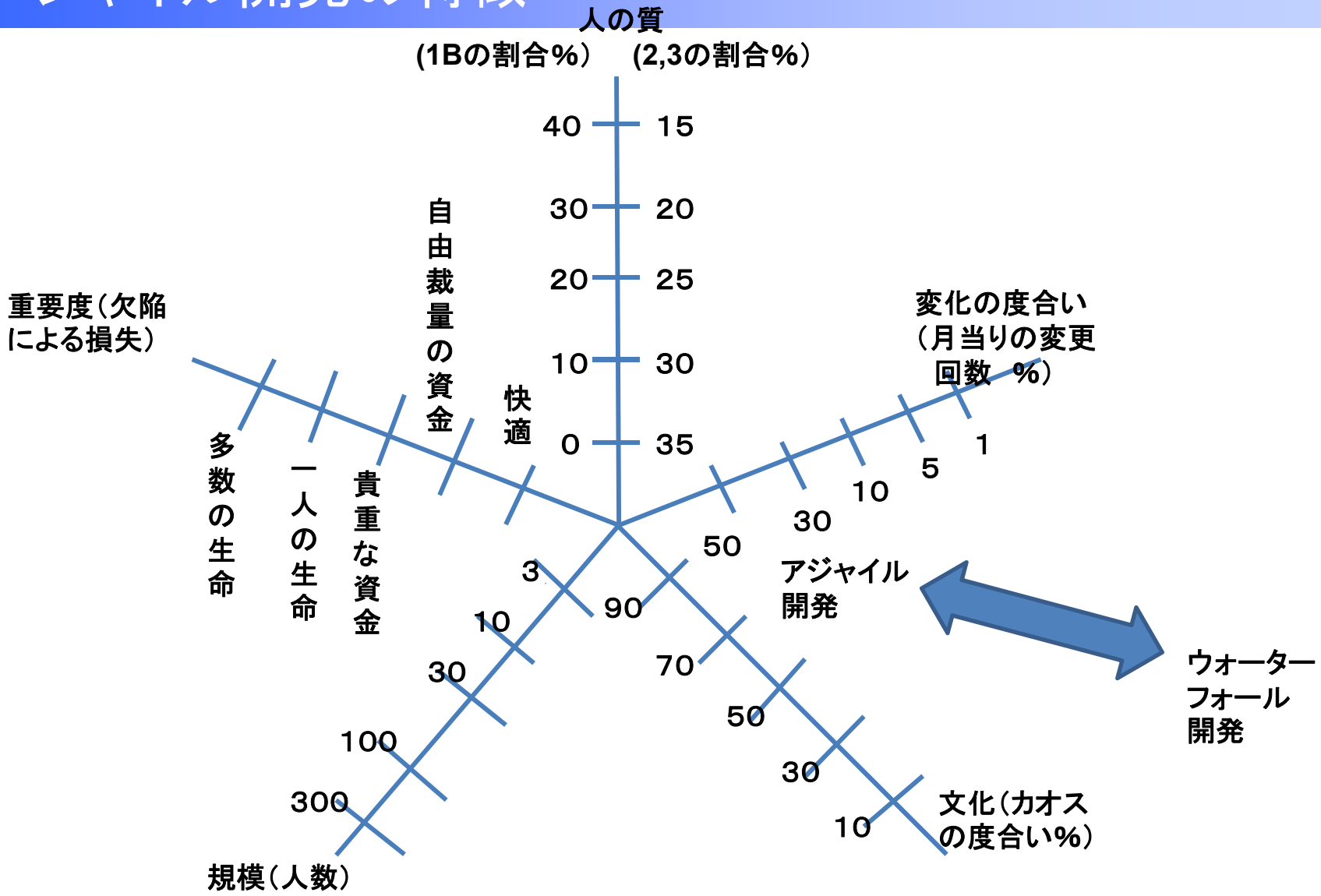


図3.1　アジャイル開発とウォーターフォール開発かの選択要因

3. アジャイル開発の特徴

(1)アジャイル開発の基本理念

アジャイル開発の基本理念は、システム全体を一括して開発するのではなく、小さな単位に分割し、段階的に積み上げていくことにある。

ー時間(開発期間)を分割する

ー機能(開発対象機能)を分割する

ー対象機能を開発期間に割り当てる(ビジネス優先順位順)

ことを行なう。結果として、アジャイル開発は表3.3に示すような8つの特徴を持っている。

(2) 自動化ツールの高度に活用

アジャイル開発における自動化ツールの活用を表3.4に示す。

① 作業効率を向上させ、作業の正確性を改善させることが出来る。

② また、自動化ツールは各種の作業管理指標を自動収集し、図表などの分かり易い形式で集計してくれる。

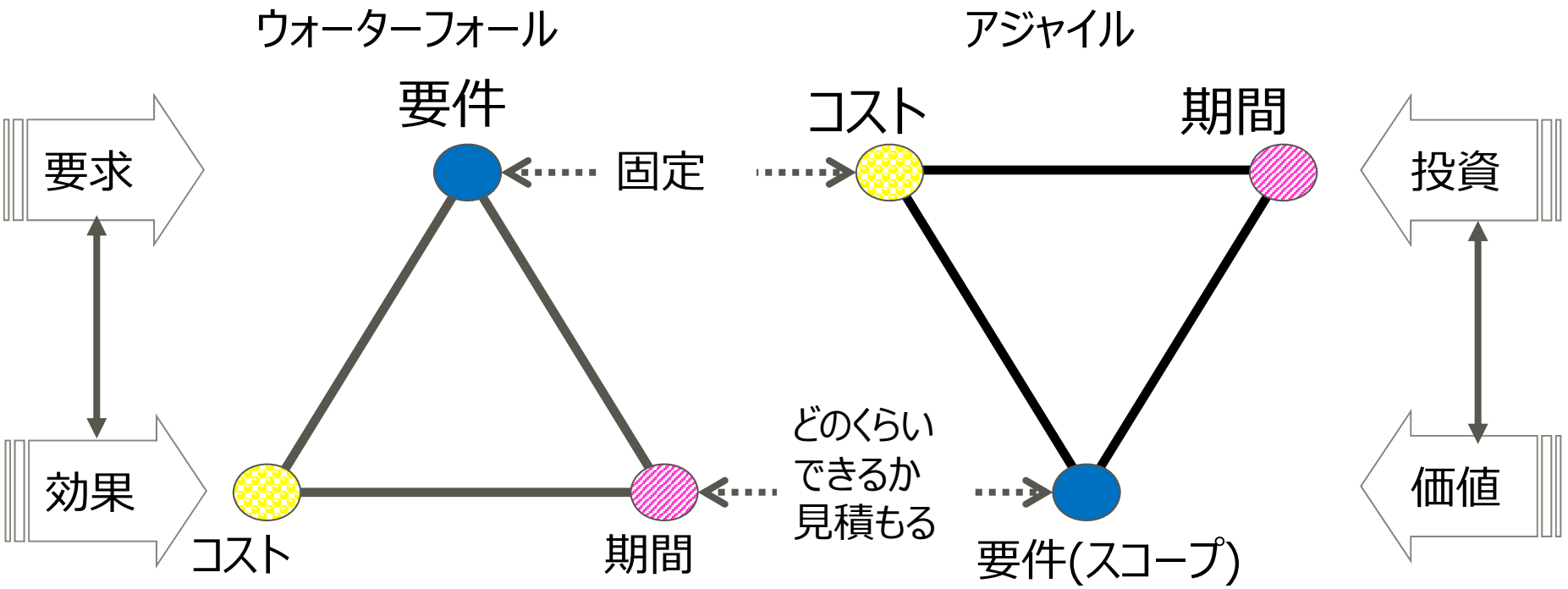
③ 代表的なアジャイル開発技法の一つであるXPにおいては、コード分析・評価ツール、継続的インテグレーションCIツール、単体テスト支援ツール 等の活用をXPの標準プロセスの中に組み込んでいる。

3. アジャイル開発の特徴

表3.3　Agile開発技術の特徴

分類	項番	特徴
ビジネス優先順位	1	重要度の高い機能から優先的に開発する
時間分割	2	動くソフトウェアで顧客と確認 （ドキュメント量を最小限に）
	3	反復型開発期間＝タイムボックス（開発期間を固定化）
	4	反復型開発期間を固定するが開発項目は固定しない
機能分割	5	開発対象機能の分割 （一つの反復型開発期間内に収まるように分割）
	6	すべての作業をチケット管理する
その他	7	顧客、チームメンバー間の直接対話
	8	自動化ツールの高度に活用

（参考）コストと期間を固定し要件を可変にする



見積り
提供
リスク

開発視点
機能視点（全ケース）
脅威(曖昧性)を減少

ビジネス視点
業務視点（限定可）
＋好機(曖昧性)を活かす

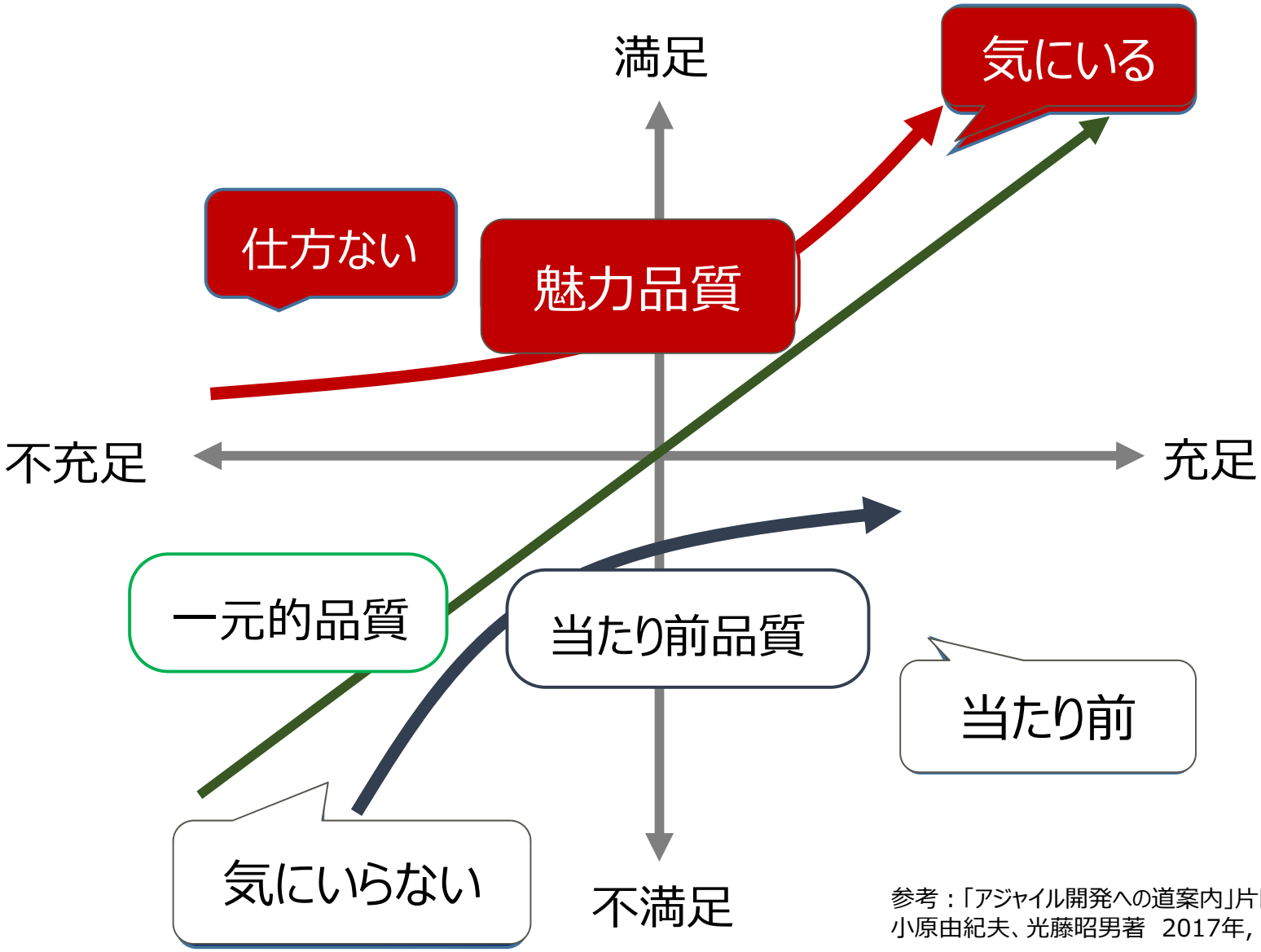
参考：「アジャイルソフトウェア要求」 Dean Leffingwell著　2014年

3. アジャイル開発の特徴

表3.4 アジャイル開発の各工程で活用される自動化ツール

Agile開発工程名	自動化ツール分類	具体的な自動化ツール（例）
コーディング	変更歴管理ツール、 バージョン管理ツール	CVS, Subversion, Git, GitHub
ビルド	ビルドツール、 構成管理ツール	ANT, Maven, Gradle
統合（CI）	CI(Continuous Integration)ツール	Jenkins
単体テスト	単体テスト支援ツール	JUnit
コードレビュー、リファクタリング	コード分析・評価ツール	Checkstyle, PMD, FindBugs, Jtest, PMD-CPD, JDepend, JavaNCSS, Covertura
機能テスト、 GUIテスト	GUIテスト支援	Selenium2
システムテスト（クラウド上のテストを含む）	システム環境設定・管理	puppet, Chef
工程全般	ITS (Issue Tracking System)	Trac, Redmine

（参考）狩野モデル



参考：「アジャイル開発への道案内」片岡雅憲、
小原由紀夫、光藤昭男著　2017年, P53

（演習＃1）アジャイル開発への疑問

アジャイル開発に対する懸念・疑問を挙げて下さい。

１．懸念・疑問のキーワードを以下の余白に記入してください。（２分）
２．お隣の方と記入した懸念・疑問に関して話してください。（２分）

4. アジャイル開発プロセス（XPとスクラム）

4.1 XP

（1） XPとは

XP(eXtreme Programming)とは、

良いと思われるもの・ことを極限(extreme)まで行い、
不要と思われるもの・ことは一切行わない

プログラム開発手法である。

XPは、代表的なソフトウェアアジャイル開発手法の一つとしてとらえられている。

（2） XPの提唱者

次の3人が共同で、XPを提唱した。

Ward Cunningham - the inventor (発案者)
Kent Beck - the articulator (表現者)
Ron Jeffries - the realizer (実現者)

上記のうち、Kent Beckが表面にたつことが多いため、XPは、Kent Beck個人によるものと誤解している人が多い。

4. アジャイル開発プロセス（XPとスクラム）

（3） XPの実践（プラクティス）

XPで実践すべき項目（プラクティス）として、次の12項目がある。

P-1. きめ細かなフィードバック
－ペアプログラミング
－計画ゲーム
－テスト駆動開発
－チーム全体

P-2. 継続的プロセス
－常時結合
－リファクタリング
－短期リリース

P-3. 理解の共有
－コーディング標準
－コードの共有
－シンプル設計
－システムメタファー

P-4. 労働時間
－継続可能な作業ペース

以下では、上記のうち、テスト駆動開発、常時結合、継続可能な作業ペース　について、更に詳細を説明する。

（4） テスト駆動開発（TDD：Test Driven Development）

TDDではテスト仕様を先に記述し、そのテストに合格する様にコーディングをする。（TDDは、テスト法ではなく開発法である、としている）

TDDのキイワード：
Red/Green/Refactor
—最初はテストだけなので不合格（赤）、コードを書いてテストに合格（緑）、そしてそれを磨いてきれいにする（リファクタリング）。

—Test a little/Code a little
/ Refactor a little
(少しづつ着実に進む)
＝＞テストを繰り返し実施する必要があり、テストの自動化が、必須である(xUnitとセットで発展）

（注）TDDを、”Test First Programing”と呼ぶ場合もある

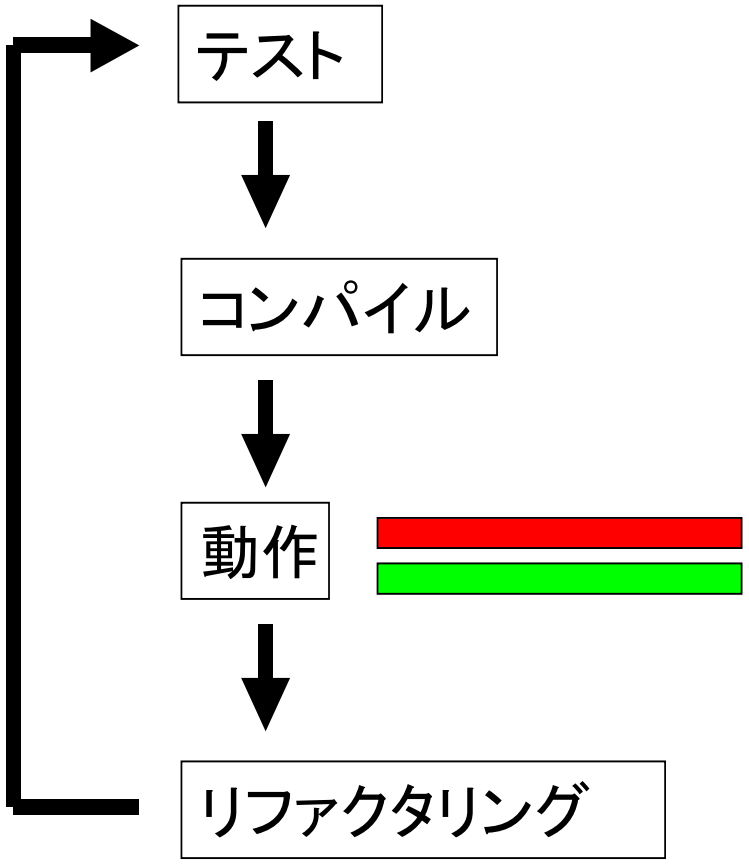


図4.1 TDDの概念図

4．アジャイル開発プロセス（XPとスクラム）

（5） 常時結合(Continuous Integration)

CIサーバは、ソースコード管理リポジトリ、ビルドツール、リグレッションテストツール他と連動する。

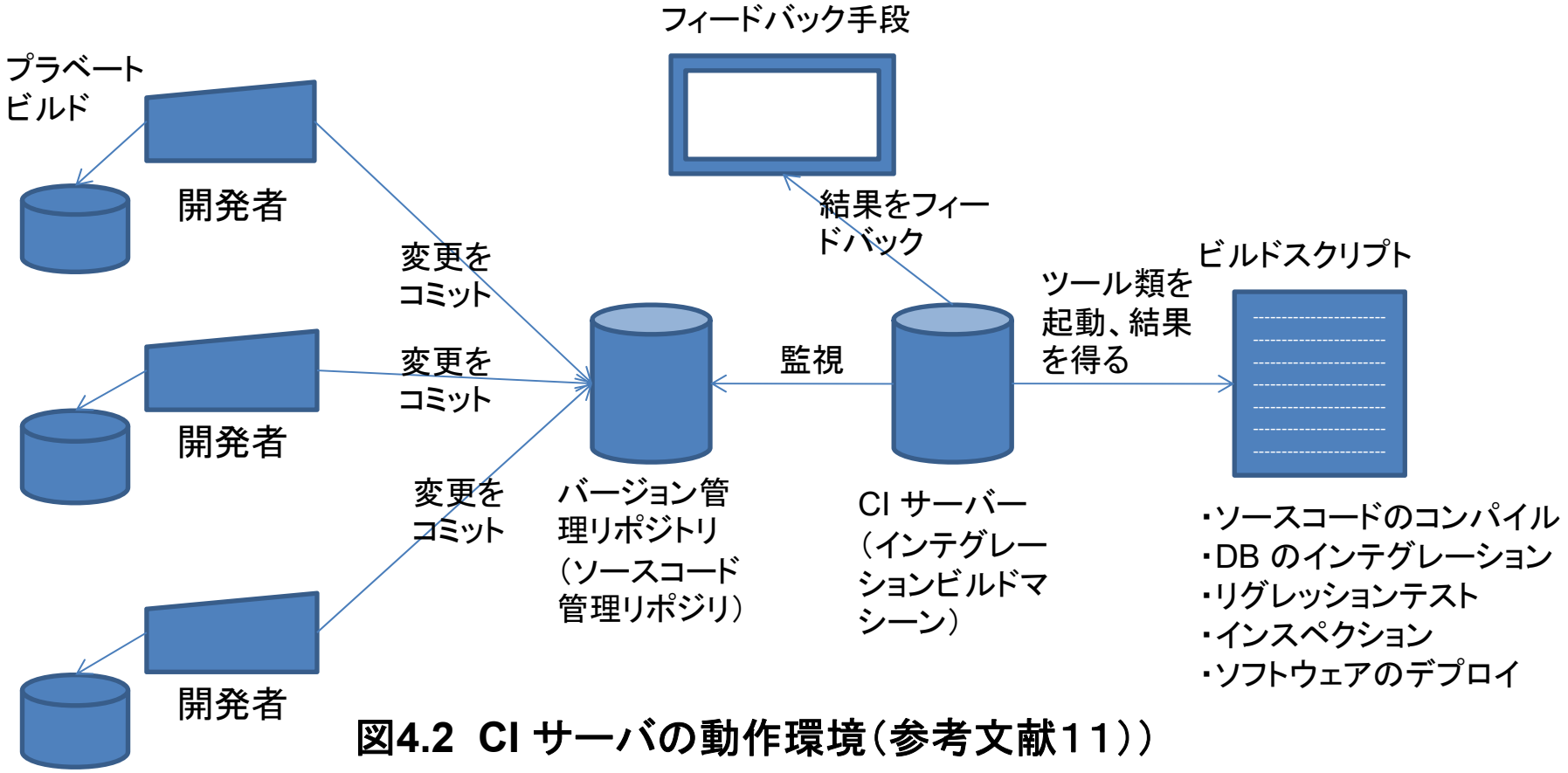


図4.2 CI サーバの動作環境（参考文献11））

（注） 開発者は各々がプライベートビルドを実行、結果が正しいことを確認した上でバージョン管理リポジトリに変更をコミットする。CI サーバは、これを監視していて、ビルド他のプロセスを自動実行する。

4. アジャイル開発プロセス（XPとスクラム）

（6） 継続可能な作業ペース

① 適正な作業時間により、良好な作業ペースを維持することが大切である。疲れていては最善のことは出来ない。結果として、混乱が生じ、生産性を落とす。

② したがって、最適なペースを維持できるように作業計画を立てて、その計画を守っていくことが大切である。このことは、作業を安易にしようとすることではなく、作業を効率良く進めようとの発想に基づいて進めるものである。

③ アジャイル開発では、反復型開発期間ごとに開発内容を調整しながら開発を進める。したがって、作業対象項目と作業時間との関係を調整し易い。

④ 米国における統計では、アジャイル開発チームの残業時間の平均は、ウォーターフォール開発チームそれよりも少ない。

4. アジャイル開発プロセス（XPとスクラム）

4.2 スクラム

（1） 新製品開発方法論としてのスクラム

製品開発の動きが早く、競争率が高い今日の世界では、速度と柔軟性が重要である。従来の直線的な方法では仕事を成し遂げることができず、ラグビーにおいてチーム内でボールをパスしながらチームが一群となって動くというような方法が必要である。日本や米国の先進的企業では、このような全体論的な開発方法を用いるようになってきている。この考え方に基づく開発方法をスクラムと名付けた。

by 竹内弘高、野中郁次郎

“The New New Product Development Game” Harvard Business Review, January-February 1986 pp.137-146

（2） システム開発方法論としてのスクラム

Ken Schwaber とJeff Sutherland は、この考え方をシステム開発方法論「スクラム」へと、発展させ、1995年のOOPSLA’95で、初めて公的に発表した。 また、Ken SchwaberとMike Beedleは、 これを次の著書にまとめた。“Agile Software Development with Scrum” （監訳）長瀬嘉秀/今野睦、（編集）株式会社テクノロジックアート、（出版）ピアソン・エデュケーション、2003年9月（原著は、2001年に出版）

4. アジャイル開発プロセス（XPとスクラム）

（3） スクラムの構成要素とその流れ

表4.1 スクラムのフレームワーク構成要素

構成要素	内容	詳細要素
スクラムチーム	プロダクトを反復的・漸進的に届けるチームである。	・プロダクトオーナー ・開発チーム ・スクラムマスター
作成物	作業や価値を表したものである。	・プロダクトバックログ ・スプリントバックログ ・インクリメント
イベント	透明性を維持し、検査・適応の機会である。	・スプリント ・スプリントプランニング ・デイリースクラム ・スプリントレビュー ・スプリントレトロスペクティブ
ルール	スクラムチームとその役割・イベント・作成物をまとめ、それらの関係性や相互作用を統括するものである。	全体で説明。

4. アジャイル開発プロセス（XPとスクラム）

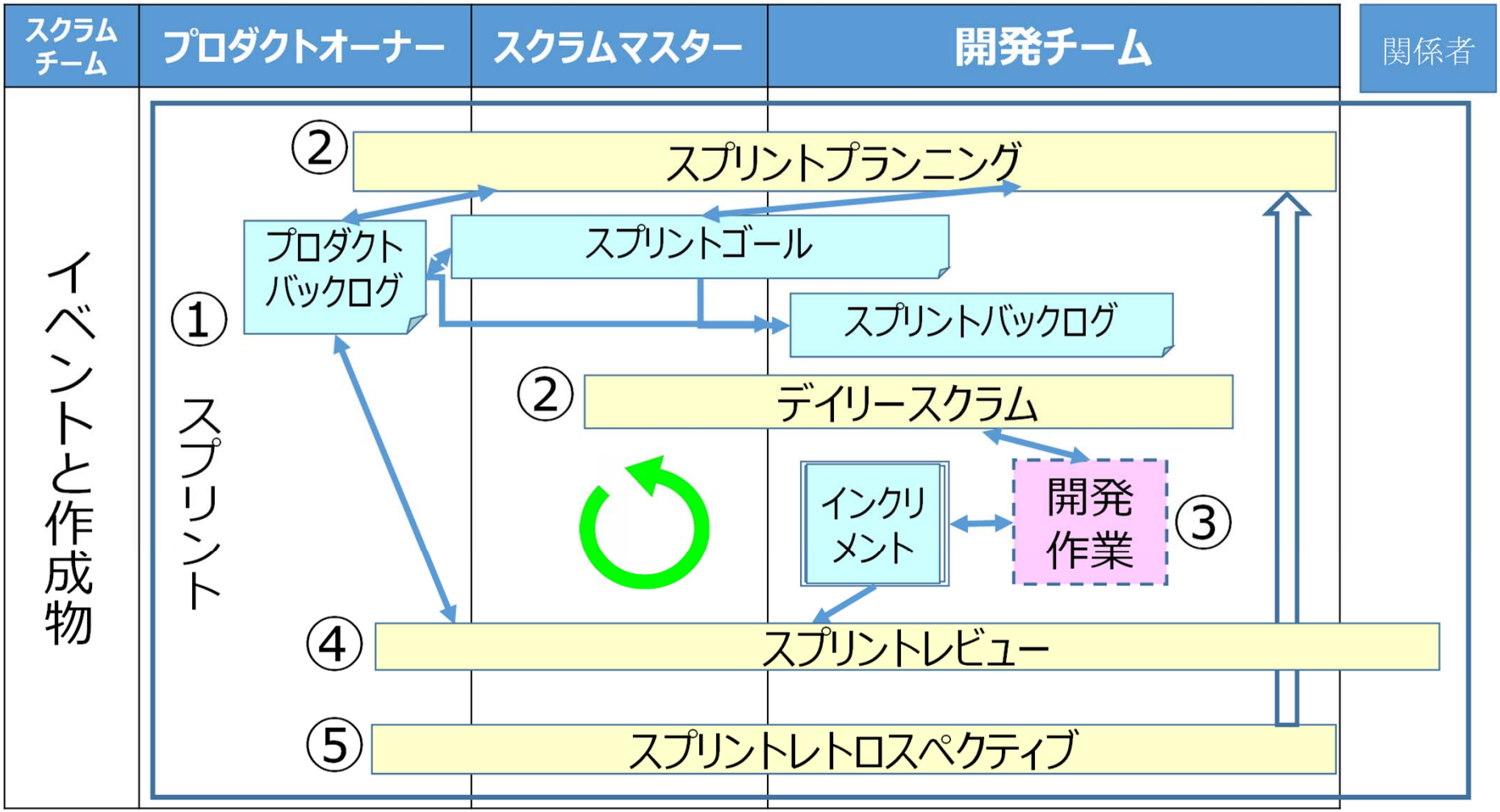


図4.3 スクラム構成要素の流れ

4. アジャイル開発プロセス（XPとスクラム）

（4）インセプションデッキ（プロジェクト憲章に相当）

表4.2　インセプションデッキの10の質問と課題

	質問と課題	目的
1	我々はなぜここにいるのか？	プロジェクトの目的を明確にする
2	エレベーターピッチ(elevator pitch)を作る	プロジェクト、プロダクトの概要を短時間で明確に説明する
3	パッケージデザインを作る	プロダクトの価値を明確にする
4	やらないことリストを作る	プロジェクトのスコープを明確にする
5	「ご近所さん」を探せ	ステークホルダーを明確にする
6	技術的な解決案を描く	アーキテクチャを具体化する
7	夜も眠れなくなるような問題は何だろう？	リスクを明確にする
8	期間を見極める	スケジュールを明確にする
9	何を諦めるのかをはっきりさせる	QCD、スコープの優先順位を明確にする
10	何がどれだけ必要か	リソースを明確にする

4. アジャイル開発プロセス（XPとスクラム）

（5） スクラムで使われる用語について（注意）

スクラムには、特徴的な用語が多い。日本国内のアジャイル開発では、XP の用語が使われることが多い。

スクラムの用語	一般的な用語	意味
スプリント	イテレーション	繰り返しの開発期間、リリース計画
プロダクトバックログ	ストーリー	顧客が選択する製品の機能を整理したもの
スプリントバックログ	タスク（チケット）	開発チームが実施する作業
スクラムマスター	アジャイルマスター	アジャイルプロセスのファシリテータ

5. アジャイル開発の効果とリスク

5.1 アジャイル開発により期待される効果

アジャイル開発では次の観点から著しい改善効果があげられる。

（1） 顧客満足度の向上

顧客が開発に深く関与し、顧客満足度の向上に寄与している。

（ｉ） 開発計画への顧客の参加、機能の開発優先度を顧客が指定
（ii） 短期開発により動作するソフトウェア機能を早期に顧客へ提供
（iii） 顧客による受け入れテスト

（2） 開発期間の短縮

アジャイル開発は、開発期間を二つの観点で短縮する。

（ｉ） 全体開発期間（開発開始から全体機能のリリースまで）の短縮
（ii） 優先度の高い機能の早期提供による実質効果上の短縮

（3） 開発効率の向上

（ｉ） ソフトウェアを早期に動かして理解を深める。
（ii） 分からないこと、新しいことを理解するために、仲間の知恵を借りる。このための、5～15分程度の短期の打ち合わせを頻繁に行う。
（iii） 自動化ツールを活用して自動化する。

5. アジャイル開発の効果とリスク

（4） 製品品質の改善

アジャイル開発では製品品質の確保の観点から次の施策を実施している。

（ｉ） 単体テストの徹底

（ii） 顧客テストの徹底

すなわち、アジャイル開発では、内部部品の積み上げである単体テストと顧客へのリリース機能に基づく外部機能の２通りのテストを行っていく。そして、これらのテストは開発と共に積み上げられ、再テストは自動化されて品質を保証していく。アジャイル開発での品質管理はテスト資産の着実な積み上げと、そのための自動化に支えられている。

（5） 参加メンバーの技量の向上

アジャイル開発チームでの働き方の基本は「自主独立」である。各メンバーは自分で考え、自分で開発し、誰かの指示や、命令で開発するのではない。同じチーム内の他のメンバーの助けを借りることはできるし、逆に、他のメンバー作業を支援することもある。しかし、他のメンバーから指示を受け、他のメンバーに指示をすることはない。各々のメンバーは独立であり、自分で自分の仕事を行う。

5. アジャイル開発の効果とリスク

5.2 アジャイル開発における開発リスクへの対応

アジャイル開発は従来開発の延長線上にはない。飛躍が必要。

表5.2 アジャイル開発のリスク

項番	アジャイル開発のリスク	説明
1	アジャイル文化の継承なし	チームにアジャイル開発の経験者がいない
2	顧客の参加なし	具体的な要求仕様を顧客から直接に聴けない
3	ドメインモデルと用語集なし	設計仕様の基礎が固められていない
4	機能分割が不十分	機能（ユーザーストーリー）分割が不十分で、小規模くり返し型開発が出来ない
5	一つの部屋で作業できない	チーム全体が一つの部屋でまとまって作業できない
6	チケットによる進捗管理をしていない	チケットを用いたITS(Issue Tracking System)が、アジャイル開発の進捗管理に必須
7	自動化ツールの活用が不十分	アジャイル開発では自動化ツールの積極的活用が前提
8	大規模ソフトウェア開発への適用	大規模ソフトウェア向きのアジャイル開発が確立されていない

5. アジャイル開発の効果とリスク

（1） チームにアジャイル開発の経験者が不在

アジャイル開発を始めるに当たっては、開発チームの中にアジャイル開発の経験者がいることが望ましい。ウォーターフォール開発とアジャイル開発との間にあるギャップを乗り越える必要がある。

－文化的ギャップ（アジャイル開発の「自主」、「試してみる」精神）

－技術的ギャップ（アジャイル開発で「標準化」、「自動化」がMust）

（2） 顧客の参加なし

アジャイル開発では、チームへの顧客の参加が前提になっている。

① 業務仕様は顧客が一番知っている

② ドキュメントによりこれを伝えるには膨大な記述工数がかかる。

③ 開発したソフトウェアを顧客にすぐに見てもらいFBしてもらう。

日本では対米国比、顧客側にIT技術者が少ないという問題がある。

（3） ドメインモデルと用語集なし

システムの概念やオブジェクト名称が決まっていないと、ソフトウェアの開発はうまく進まない。これは、アジャイル開発だけでなくウォーターフォール開発においても言える。開発に当たっては、ドメインモデルとこれにリンクした用語定義集を作成しなければならない。

5. アジャイル開発の効果とリスク

（4） 機能分割が不十分

アジャイル開発では、開発項目をユーザストーリーと呼ばれる小機能に展開し、作業管理対象としていく。ウォーターフォール開発においても機能分割が必要である。しかし、アジャイル開発での機能分割では、その分割したそれぞれを短期間のうちに動作させなければならない。

アジャイル開発では、各リリース対応の大きな機能分割をし、そのリリース対応の開始時点で反復開発期間毎の小さな機能に分割する。

（5） 一つの部屋で作業できない

アジャイル開発では、チームメンバー全員がフェースツフェースでコミュニケーションできる環境で作業する。このことを前提に小規模な非公式ミーティングを頻繁に行い、また、ドキュメントの量を減らしている。

（6） チケットによる進捗管理をしていない

アジャイル開発では、高速開発での正確な作業管理のためにすべての開発項目（ユーザストーリー）にIDを付ける。ユーザストーリーに展開されない非機能項目であっても、その調査、検討に時間を取られるのであればIDを付ける。すなわち、すべての作業にIDを付ける。IDのない作業は、プロジェクトとして認められていない非公式の作業となる。

5. アジャイル開発の効果とリスク

（7） 自動化ツールの活用が不十分

アジャイル開発では自動化ツールを積極的に活用することが当たり前になっている。そして、アジャイル開発では、自社固有のツールは殆ど使わない。オープンの世界から外部調達する。しかも、大半のツールはOSS(Open Source Software)であり、無料で調達できる。例えば、表3.4 に紹介した自動化ツールはすべてがOSSである。

（8） 大規模ソフトウェア開発への適用

アジャイル開発は、提案され始めた当初は小規模なシステム開発に適用されてきた。大規模なシステム開発では、過去の文化の慣性が強く、アジャイル開発の適用はなかなか進まなかった。このために、アジャイル開発は小規模システム開発のプログラミング工程以降を主たる対象としてきた。

しかし、工期短縮、開発効率向上の観点から、アジャイル開発へのニーズは高く、大規模システム開発への適用が検討され、その成果が出てき始めている。具体的には、要求分析・定義、概念設計、等の上流工程へのアジャイル開発の適用が検討されるようになってきている。このような新しい流れについては、次の6章で解説する。

6. 上流工程を組み込んだ拡張アジャイル開発

表6.1 上流工程を組み込んだAgile開発技法

Agile開発技法名称	開発・提案者	概要
Agile Modeling (AM)	Scott Ambler	AMは、軽量な設計方法を提案している。設計法自体は既存のものから選択して用いる。AMはそれを軽量に、短期間で用いる方法を提案している。AMは、設計方法のみを提案していて、実際の開発ではAMとXP、RUPなどと組み合わせて使う。
Agile ICONIX	Doug Rosenberg	ICONIXは、オブジェクト指向ベースの開発法であり、ユースケースを系統的に変換してコードを得る。上流工程での曖昧性を早期に取り除くのが特徴である。Agile ICONIXは、それをアジャイル化したもの。XPを強く批判していて、XPの悪いところを補ったアジャイル技法を目指している。
Scaled Agile Framework (SAFe)	Dean Leffingwell	①単独チームによるアジャイル開発から、 ②複数チームによるアジャイル開発、 ③企業ポートフォリオレベルのアジャイル開発へとスケールアップしていく方法を提唱する。その中で、ART(Agile Release Train)、アーキテクチャ滑走路(Architectural Runway)などの大規模システムのためのアジャイル技法を提案している。

6. 上流工程を組み込んだ拡張アジャイル開発

(1)SAFeSAFe(Scaled Agile Framework)とは

SAFeとは、名前の通り、小規模システム開発向けとみなされてきたアジャイル開発技法を、大規模システム開発向けに拡張した技法である。SAFe開発・普及活動のリーダーであるDean Leffingwellは、一貫して大規模システム開発領域で活動してきた。

(2)スケールアップ可能なアジャイルプラクティス

Leffingwellは、大規模システムにもスケールアップが可能なプラクティス7件と大規模システムアジャイル化特別プラクティス7件を抽出した。スケールアップ可能なプラクティスは下記の通りである。

① 定義/ビルド/テストコンポーネントチーム
② 2 レベル計画作りと追跡
③ 反復型開発の習得
④ 頻繁な小規模リリース
⑤ コンカレントテスティング
⑥ 継続的インテグレーション
⑦ 継続的な考察と適応

６．上流工程を組み込んだ拡張アジャイル開発

（３）大規模システムのアジャイル化のための特別プラクティス

① 意図的なアーキテクチャ

（アーキテクチャ滑走路（Architectural Runway））

② リーン要求開発のスケールアップ

－ビジョン：開発構想、共通フィーチャー

－ロードマップ：個別フィーチャーをリリース計画に展開

－ジャストインタイムの詳細化：ユーザーストーリーなどで展開

③ システムオブシステムとアジャイルリリーストレイン

－アジャイルリリーストレインART(Agile Release Train)

（すべてのコンポーネントのリリース日程を同期化）

④ 高度に分散したチームの管理

－インスタントメッセージ、Eメール、カンファレンスルーム

－CI(Continuous Integration)ツール

⑤ 顧客とオペレーションへのインパクトの調整

⑥ 組織を変化させる

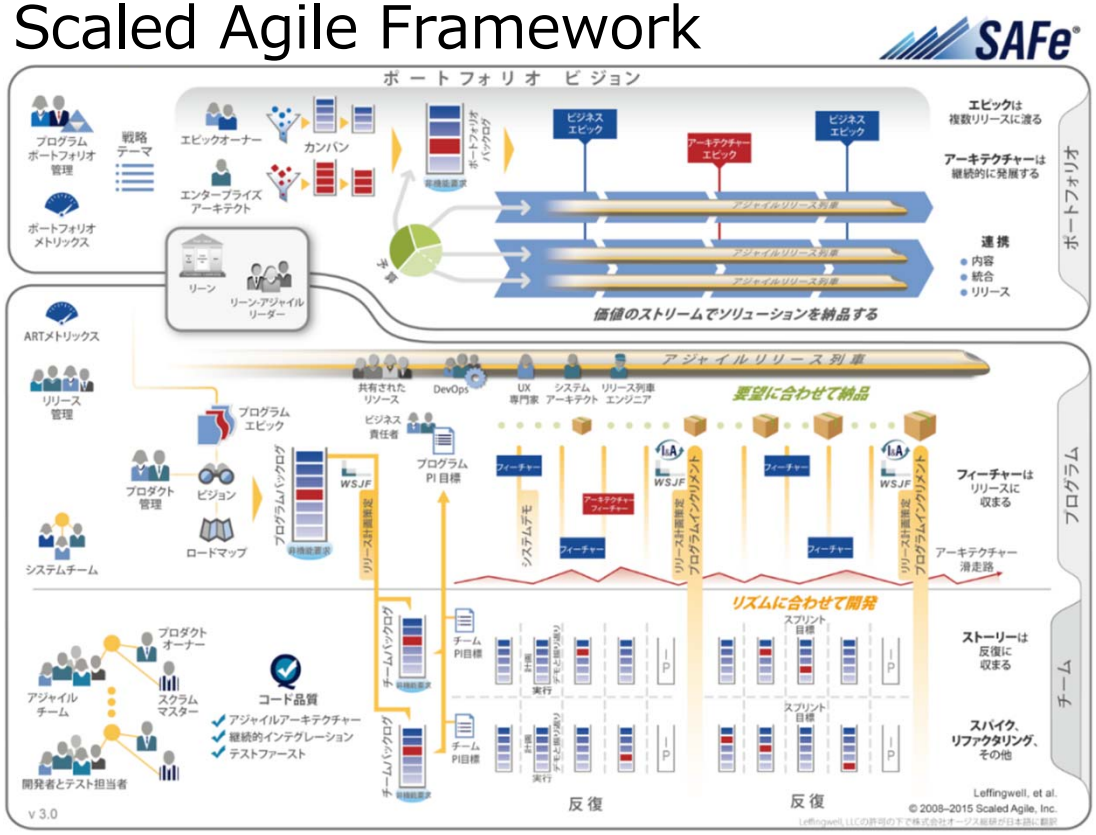
－トップレベルの決断、トップの説得

⑦ ビジネスパフォーマンスを計測する

6. 上流工程を組み込んだ拡張アジャイル開発

アジャイル開発とポートフォリオ・プログラムマネジメントを連携させたフレームワークを実践した事例が多数発表されている。**経営層とチームが連動**。

Scaled Agile Framework



ポートフォリオレベル：
　経営陣の**戦略的な投資**判断に基づいて企画を行います。

プログラムレベル：
　企画に対して**複数チーム**で開発する計画を立案し、プロダクトとして同期してリリースします。

チームレベル：
　チームが分担した要求を**5-9名のチームで**プロダクトを**開発**します。

（演習＃2）アジャイル開発への疑問の見直し

P23で挙げた、アジャイル開発に対する懸念・疑問を見直して下さい。

１．P23の懸念・疑問のキーワードを分類して下さい。
- 懸念・疑問が解消した。（×、横線を引いて下さい）
- 懸念・疑問が残っている（そのままにしてください）

２．新たな懸念・疑問を追加して下さい。

３．お隣の方と上記に関して話してください。

7. アジャイル開発の事例

7．１ アジャイル開発事例の説明方法

（1）事例一覧

事例		利用者	P.O.	開発チーム	特徴
事例1	新しいサービスの共創	プロ野球球団	大手ITベンダー		多様なチームの共創による新たなサービス構築
事例2	新しい業務への挑戦	一般住民	自治体	大手ITベンダー	業務ノウハウを活用した開発による新たな業務の実現
事例3	請負開発での導入	社員等	顧客企業	独位系ITベンダー	開発ノウハウの活用と顧客の参加の喚起による請負開発での導入
事例4	米国ﾊｲﾌﾞﾘｯﾄﾞｱｼﾞｬｲﾙ	社員	顧客企業	ｸﾞﾛｰﾊﾞﾙベンダー	大規模チームへのハイブリッドアジャイルの導入
5	新商品開発	消費者	社長	社員	35年間継続のIT以外の新製品開発

7. アジャイル開発の事例

（1）事例におけるアジャイルの特徴の実践

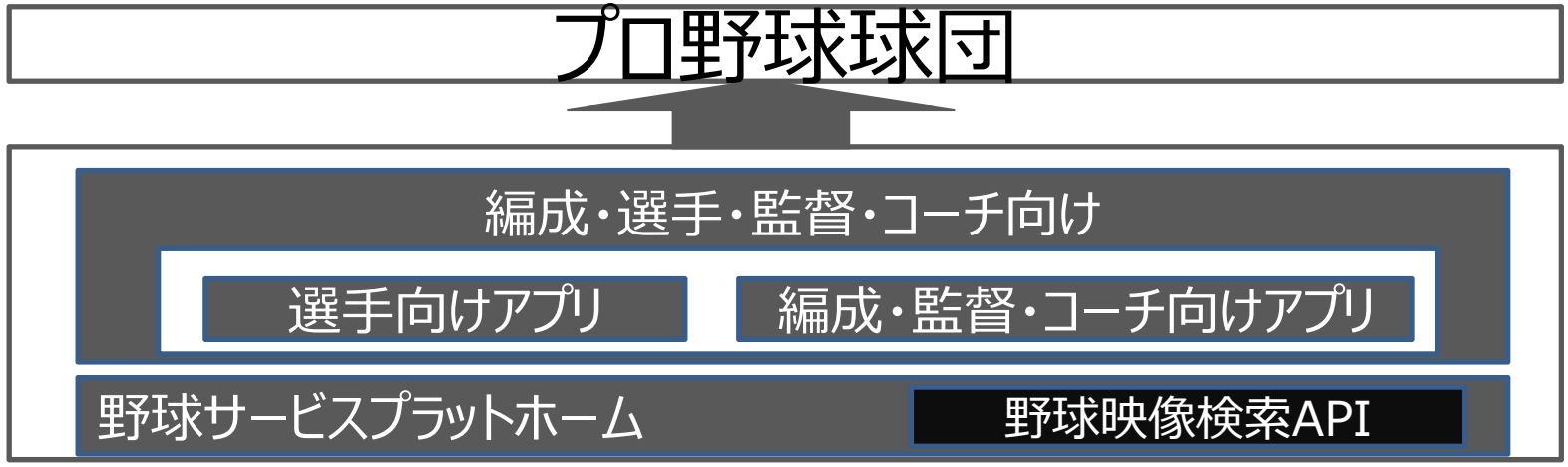
質問視点	事例1	事例2	事例3	事例4	事例5
高重要度から優先開発	チームで議論	機能、業務の両面から設定	顧客と価値を掘り起こす	基盤となる機能を優先	注力事業を優先
動くソフトで顧客と確認	称賛をフィードバック	利用者との協調	体感と連動させた	領域毎の利用者代表が参加	試作品で確認
開発期間を固定化	2週間 ＋スプリント0	2週間（初回のみ4週間）	2週間	3週間	1週間 (35年間継続)
開発項目は固定しない	P. Oの一貫した優先付け	10ｲﾃﾚｰｼｮﾝまでに基本機能	試使用スプリントを活用	20%を可変とした	フィードバックに対応
開発対象機能の分割	リーダーの自律的分割	業務と処理・操作で分割	チーム全員での計画	全員での計画立案	伴走方式で対応
全作業をチケット管理	5つのステータスで管理	プロセスとツールを連動	ﾀｽｸﾎﾞｰﾄﾞ+ﾊﾞｰﾝｱｯﾌﾟﾁｬｰﾄ	PMOが管理	チームで管理
顧客、チームの直接対話	定期的なメーティング実施	検証イテレーションを設定	試使用結果のフィードバック	非公式に利用者代表と会話	プレゼン会議で社長確認
自動化ツールの高度に活用	Atlassianの適用	Git, Jenkins, Redmine等	Redmine, TestLink等	ﾊﾞｰﾝﾀﾞｳﾝﾁｬｰﾄ	TV会議を活用
ドキュメント量を最小限に	AtlassianのWikiを活用	運用・保守ﾌｪｰｽﾞを分離	受託契約に従う	ｳｫｰﾀｰﾎｰﾙ開発規約を準拠	1枚のみ

7. アジャイル開発の事例

7．２　新しいサービスの共創

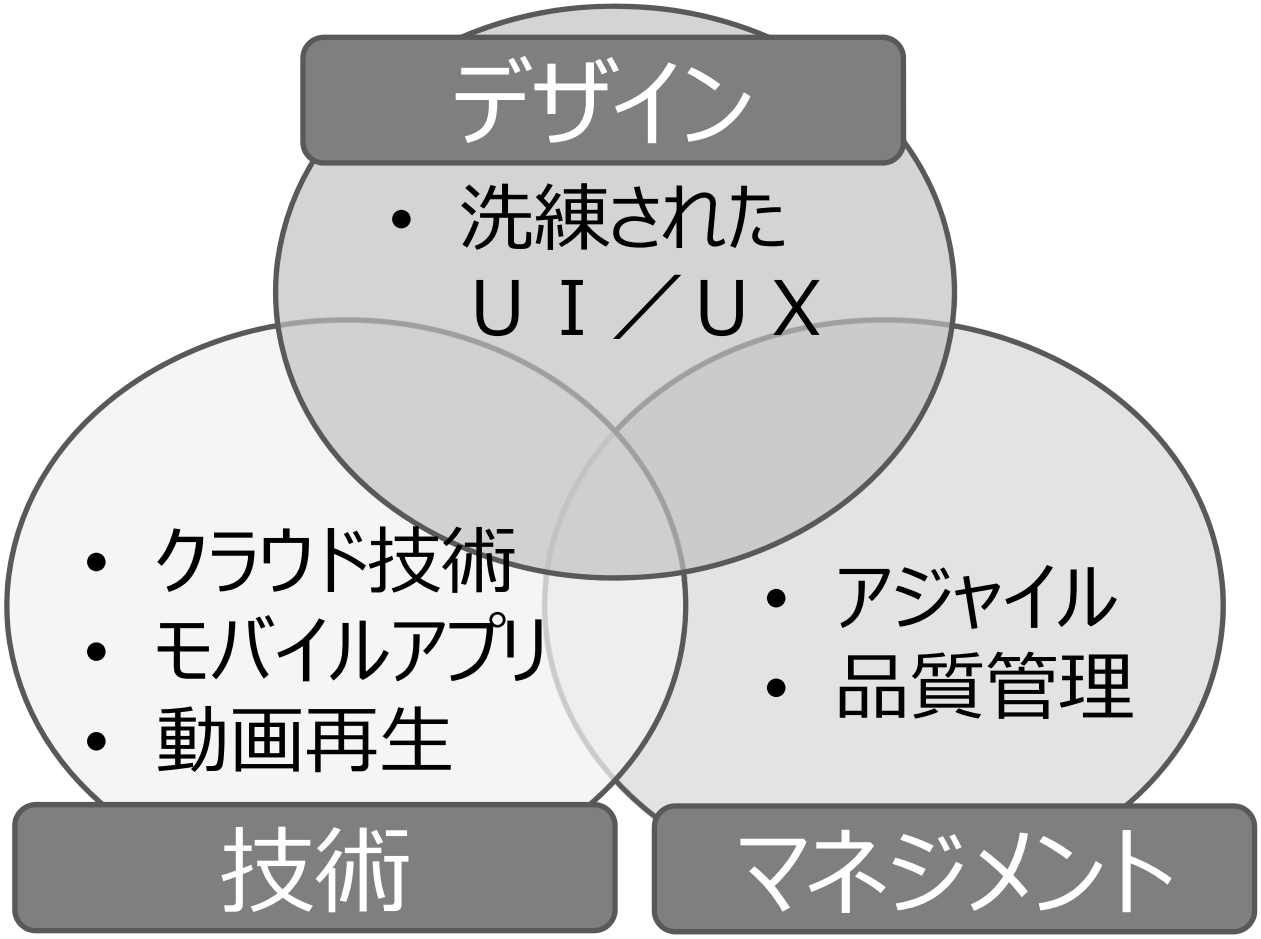
（1）概要

目的	プロ野球の価値を高める。（映像提供、選手の価値）
サービス	プロ野球ファン、プロ野球球団向けの新たな野球映像ソリューション
時期	2015年4月プロジェクト開始、2016年5月サービス提供
体制	20名／6社（デザイン、技術、マネジメントの各専門家）、3チーム
開発環境	CI環境@渋谷シェア・オフィス
推進方法	2週間のスプリントを推進、一部の遠隔メンバーと協働



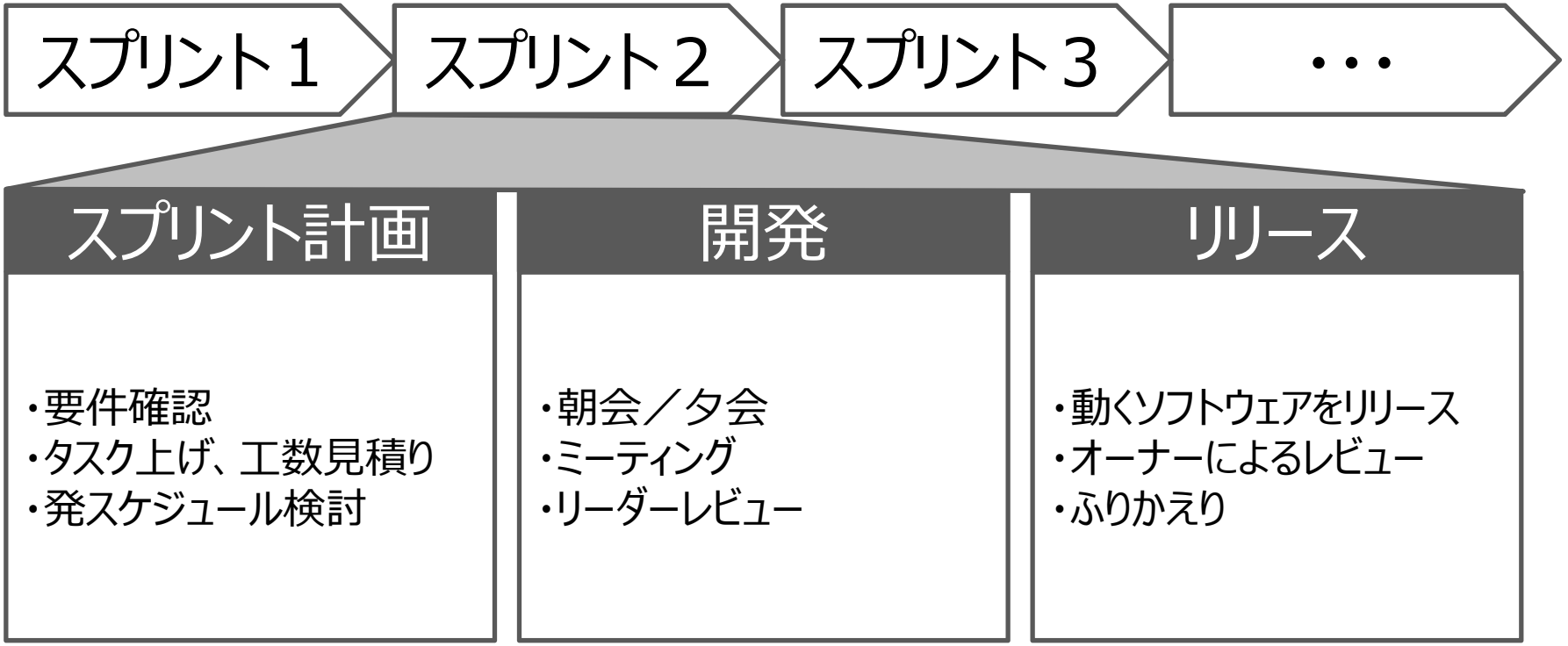
7. アジャイル開発の事例

（2）体制



7. アジャイル開発の事例

（4）推進



7. アジャイル開発の事例

7. 3　新しい業務への挑戦

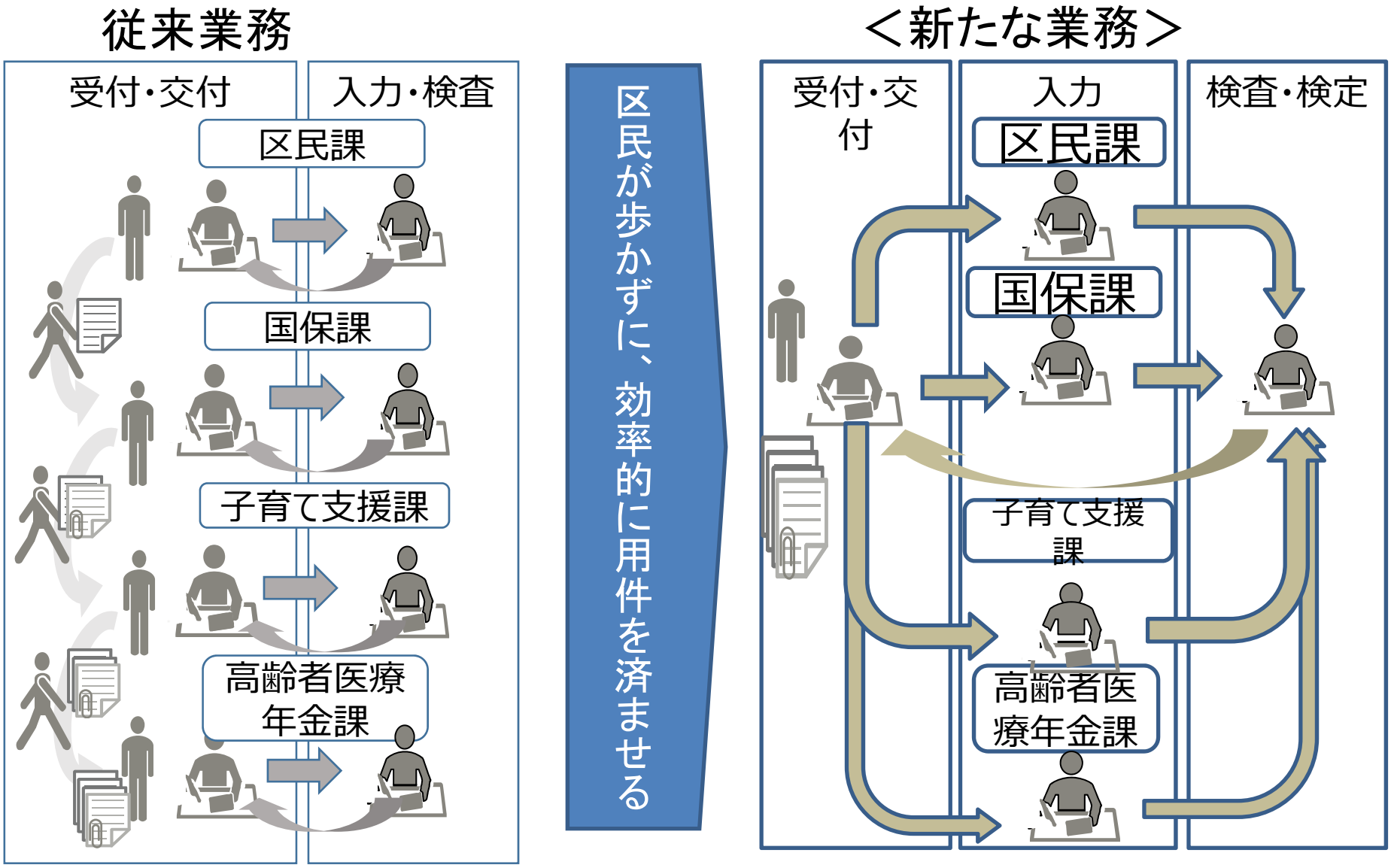
(1)概要

目的	区民をできるだけ庁舎内を歩かずに効率的に用件を済ませる窓口の提供
サービス	複数の手続きを短時間で効率よく済ませられるサービス
時期	2013年2月プロジェクト開始、2014年2月サービス提供
体制	利用者:　名／　　部門(豊島区役所) 開発チーム:5名／1チーム(富士通株式会社)
開発環境	CI環境@富士通開発センター
推進方法	2週間のイテレーションX25回(1年間) 初回のイテレーション完了時から「動くソフトウェア」を提供

75%の区民の満足度が2倍、不満が3分の1に

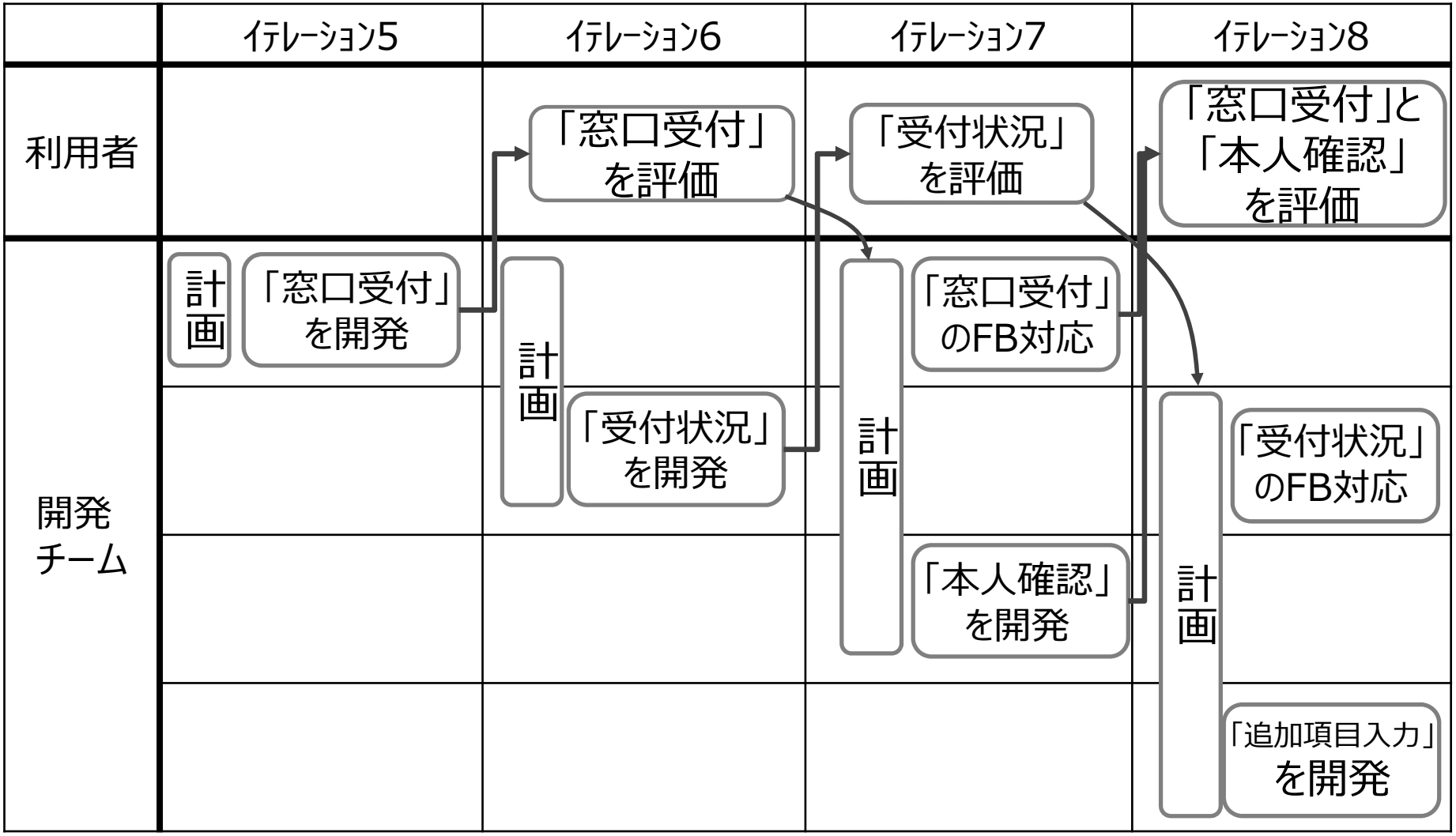
7. アジャイル開発の事例

（2）新たな業務



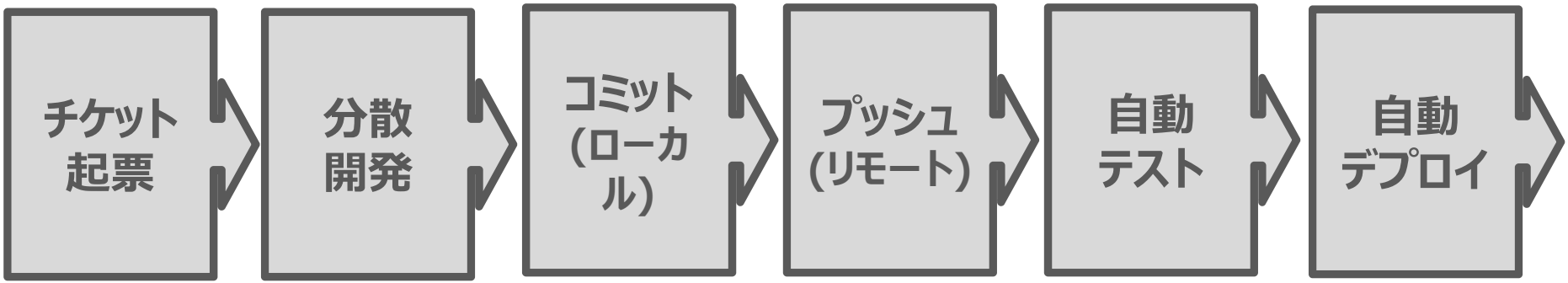
7. アジャイル開発の事例

（3）利用者と共創するイテレーション



7. アジャイル開発の事例

(4)チケット管理の流れとツール



タスク管理ツール
Redmine

分散バージョン管理ツールGit

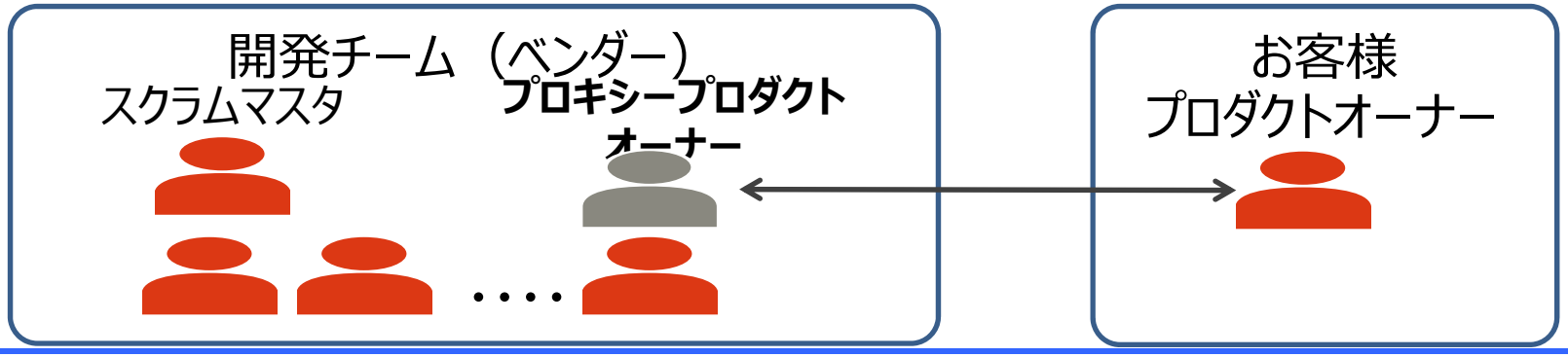
CIツールJenkins

7. アジャイル開発の事例

7．4　請負開発でのアジャイル開発の導入

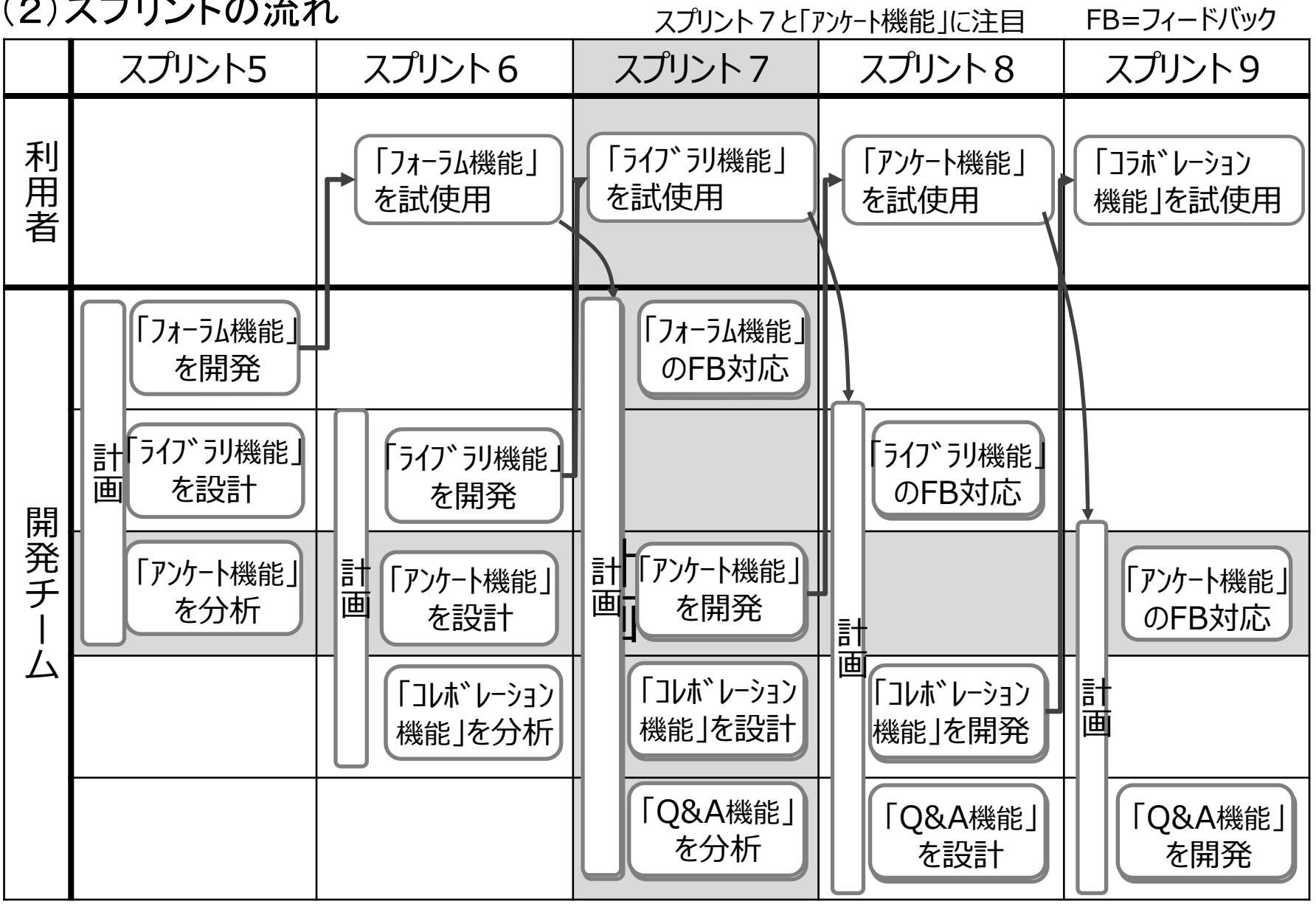
（1）概要

目的	「ソフトウェア技術を通じて、社員と家族、そして関係するすべてのお客様の幸せを追求する」（ASE社経営理念より）
サービス	請負契約におけるアジャイル開発　20案件以上
時期	2011年3月より、スプリントは原則2週間
体制	社内およびビジネスパートナー
開発環境	1ケ所での開発または、ニアショアでの開発
推進方法	要件定義：お客様が真に必要としてい価値を掘り出す。 テスト：CI環境は必須とし、自動テストを原則とする。 スプリント計画：メンバー全員でタスクを洗い出し、チケット化する。 作業：チケット駆動型で行う。 進捗：バーンダウンチャートをアナログ（手書き）更新して確認する。



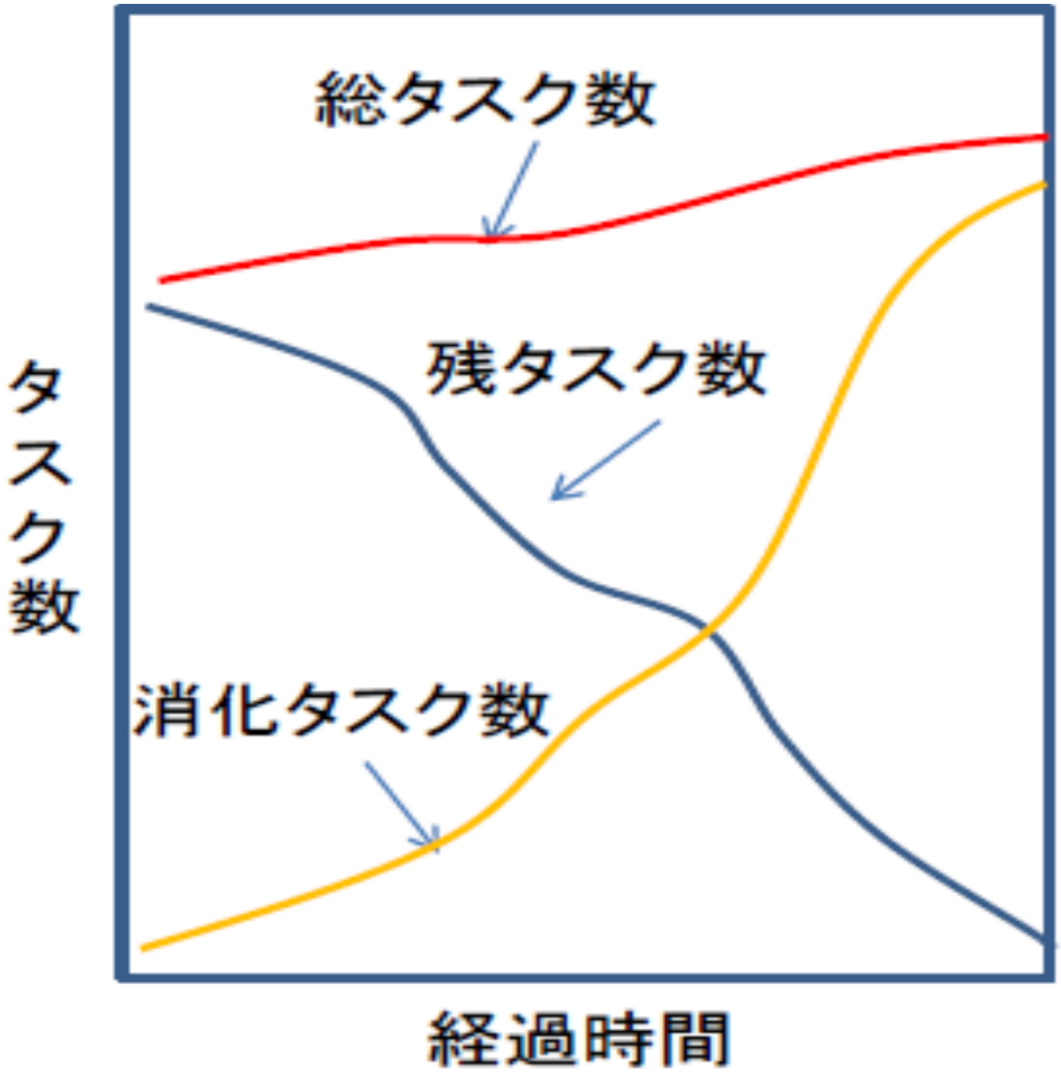
7．アジャイル開発の事例

（2）スプリントの流れ



7. アジャイル開発の事例

（3）バーンアップチャート



7. アジャイル開発の事例

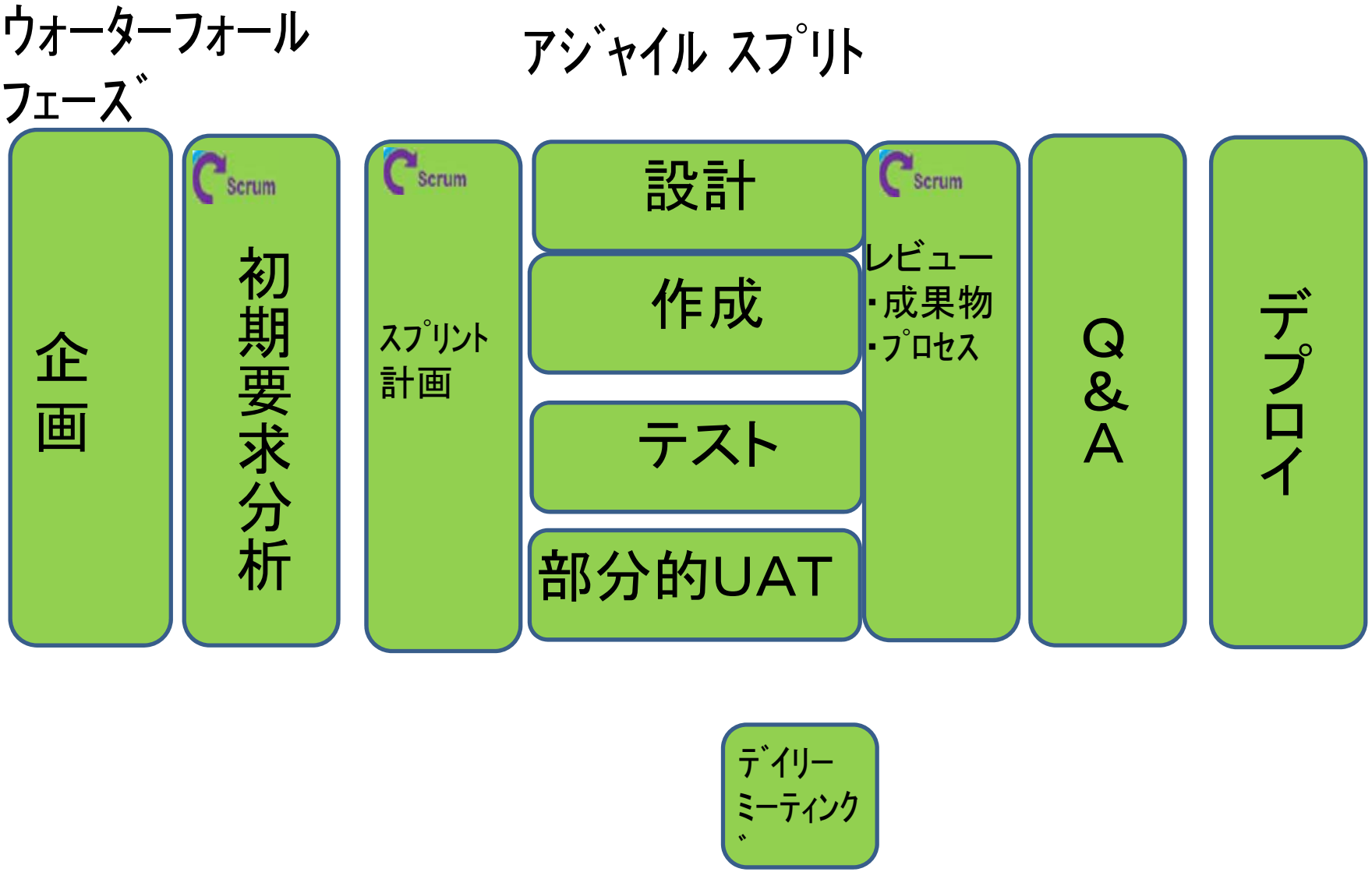
7．５　米国におけるハイブリッドアジャイルの採用

（1）概要

目的	大規模公共システム
サービス	複数システムと連携し、データ移行したセキュリティの高いサービス
時期	全体＝1年半、アジャイルスプリント＝2012年5月から7ケ月
体制	大規模ウォーターフォール開発体制とスプリント時の体制の両立
開発環境	1ケ所での開発
推進方法	企画、要件定義：ウォーターフォール開発（一部、スプリント） アジャイルスプリント（2～3週間：7ケ月）：アジャイル開発 品質確認、デプロイ：ウォーターフォール開発

役割	責任
スクラムマスター/副ﾌﾟﾛｼﾞｪｸﾄﾏﾈｰｼﾞｬｰ	・スプリント計画とイテレーションバックログ定義を管理 ・リスクと懸念事項の軽減と必要に応じたエスカレーション ・日々のスクラムミーティングを管理
開発チーム	・開発タスクの作成と実施 ・日単位のタスク進捗の更新
テストチーム	・テストケース作成と実施 ・リグニッションテストの定義とユーザー受入テスト(UAT)の定義
PMO	・作業計画とタスク割り当ての管理 ・ステークホルダーへのプロジェクト状況報告
全員	・スプリント計画とレビューへの参加 ・日々のステータスミーティングの参加

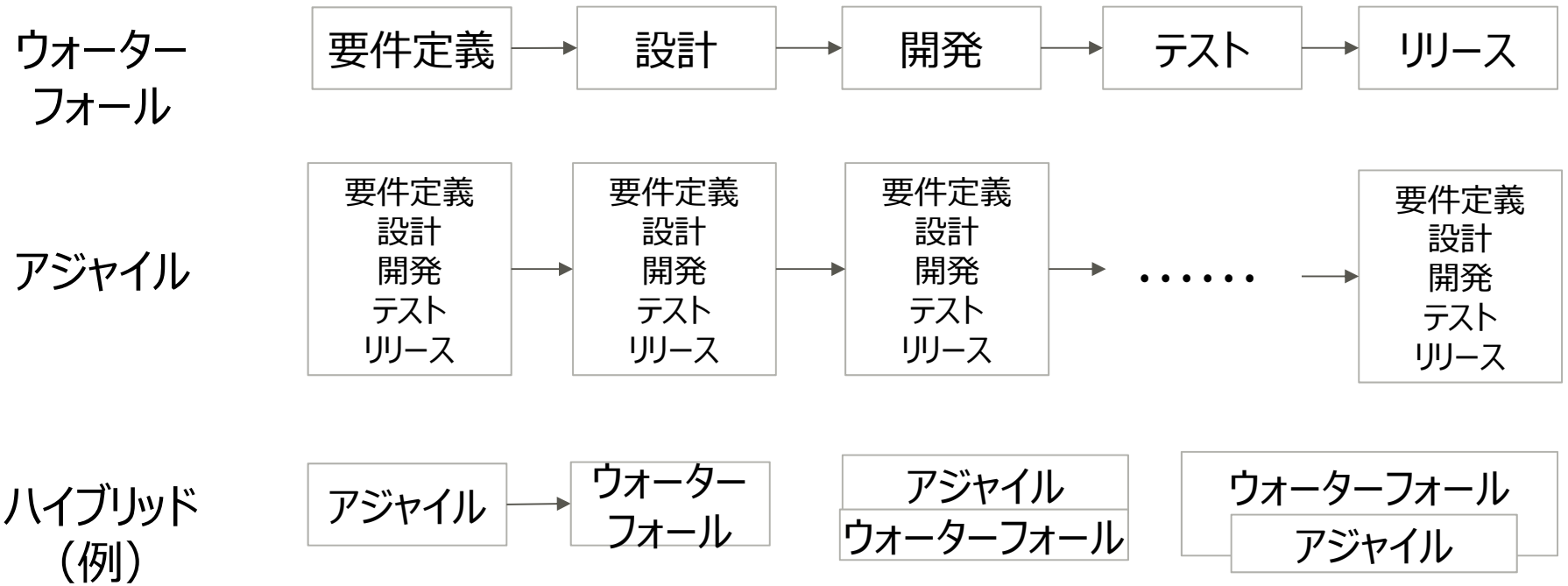
7. アジャイル開発の事例



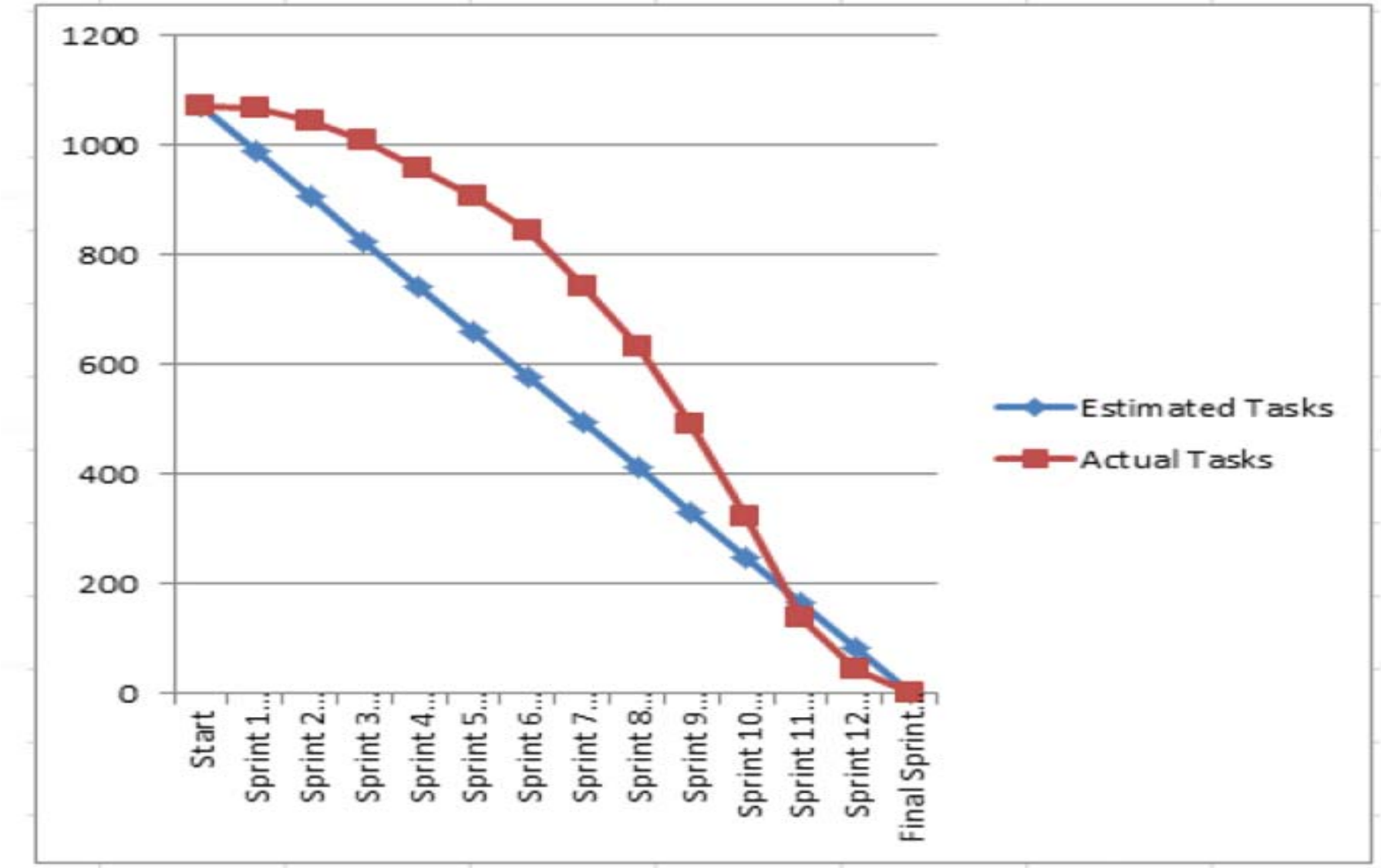
（参考）ライフサイクル

- ウォーターフォール：要件が固定＆全体で１回実行・リリース
- アジャイル　　：要件が変動＆頻繁で小さな反復・リリース
- ハイブリッド　　：上記の複合

参考：アジャイル実務ガイド、**ＰＭＩ**、**2016**



7. アジャイル開発の事例



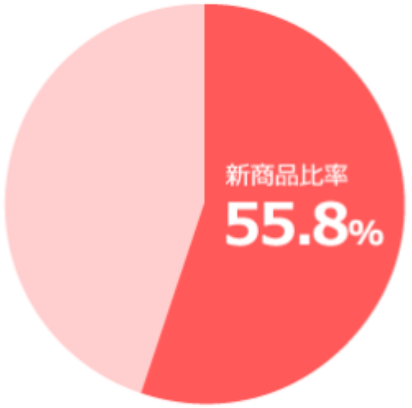
7. アジャイル開発の事例

7．６　変化に対応する新製品開発

（1）特徴

目的	移り変わる生活者ニーズに常に応える
サービス	現実の生活者が困っている悩みを解決する新製品
時期	35年前（1970年代）から開始、毎週月曜日（朝～夕方）
体制	社長、各部門の責任者、開発チーム
開発環境	専用のプレゼン室、TV会議（動画）
推進方法	資料は1画面。「要点から話す」。途中で合否判定。5～10分／件

新製品を出し続ける

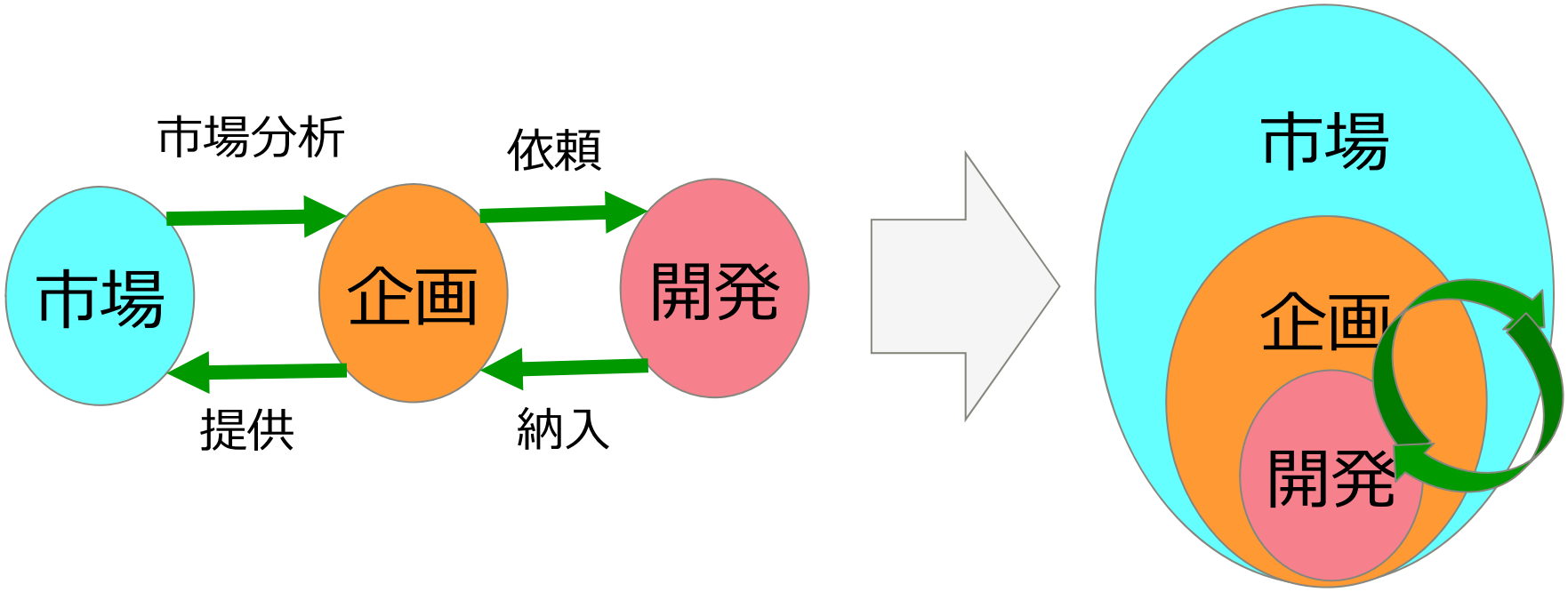


（参考）組織アジリティ

部門毎の完了の連携では、市場変化に対応できないことが増えている。

市場変化に対応するため、市場～企画～開発のサイクルを早める必要がある。

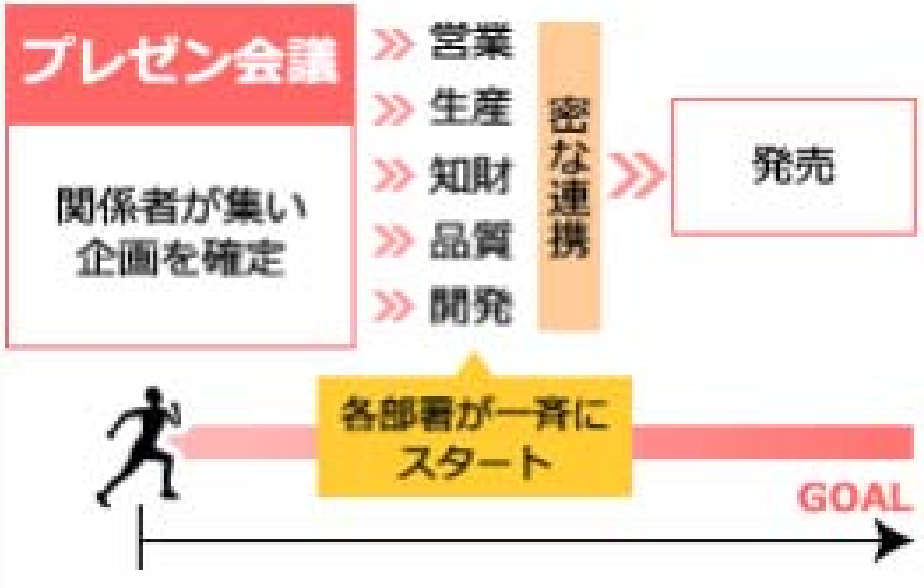
➡ 組織としての俊敏性（組織アジリティ）を高め、”デジタル企業”へ



参考：経営者のための「DX時代のイノベーション戦略」（第2回）平鍋 健児／2017.12.20
https://jbpress.ismedia.jp/articles/-/51870?page=4

7. アジャイル開発の事例

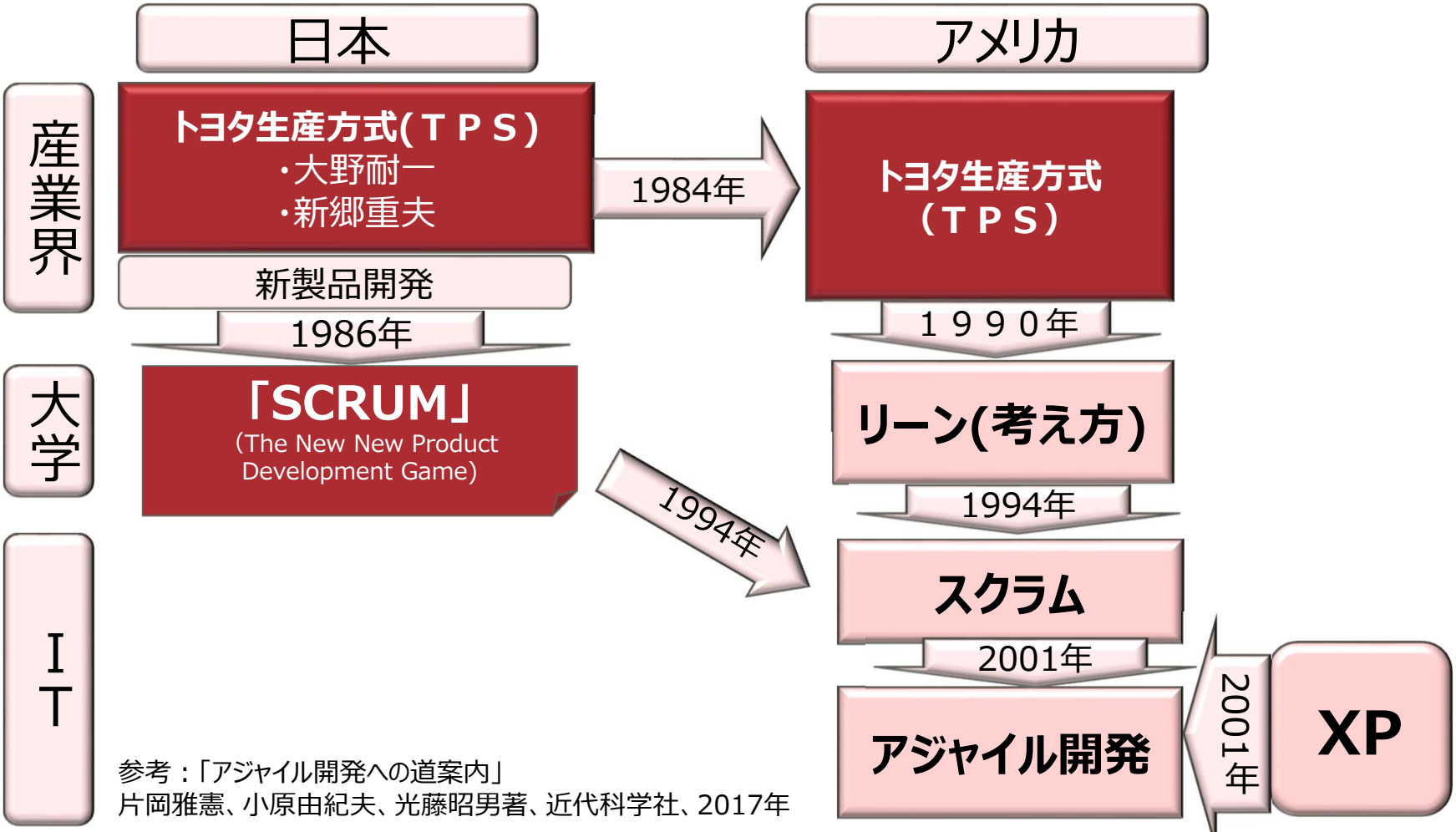
（3）35年間継承している日本の実践ノウハウ



7. アジャイル開発の事例

(3) 日本の実践ノウハウとスクラム

アジャイル開発は日本の実践ノウハウを触媒とした。



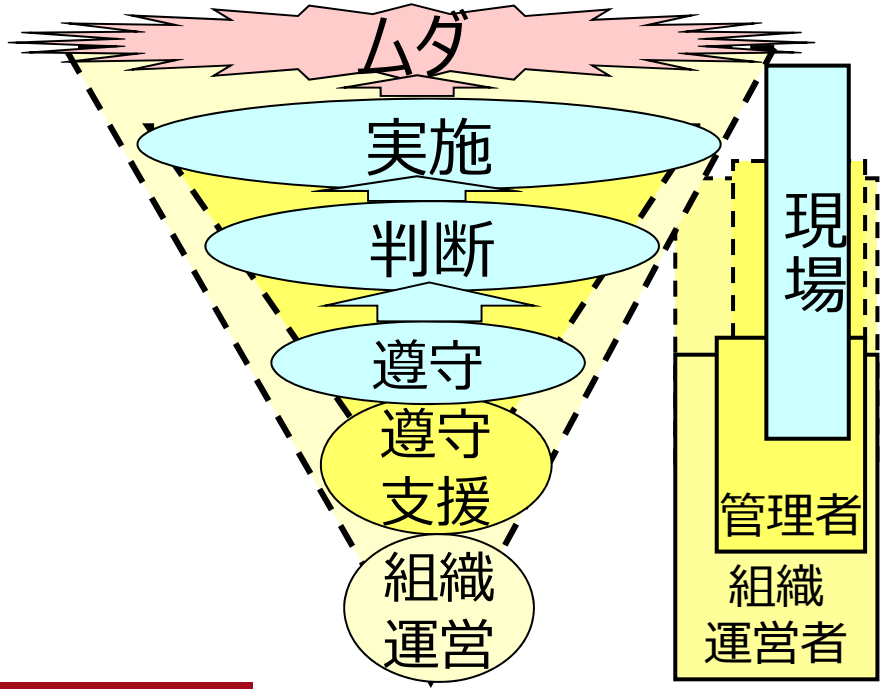
参考：「アジャイル開発への道案内」
片岡雅憲、小原由紀夫、光藤昭男著、近代科学社、2017年

（参考）リーン・スタートアップとトヨタ生産方式

『リーン・スタートアップ』 エリック・リース著, 日経BP社、2012　世界で100万部

- 「日本のトヨタ生産方式の各種の概念をうまく応用すれば、企業に伴う様々な問題を解き明かせる」
- 「リーン・スタートアップの手法のすべてが５回のなぜから導ける」
- 「５回のなぜとは、単に真因を調べることではなく、チーム内に共通の理解と視点を醸成する方法である」

なぜなぜ５階層モデル



参考：「アジャイル開発への道案内」 片岡雅憲、小原由紀夫、光藤昭男著、近代科学社、2017年

シリコンバレーも、トヨタ生産方式を学んでいる。

補遺：P2Mプログラムマネジメントとアジャイル開発

・P2Mもアジャイル開発も「顧客価値の向上」を目指す
・プロジェクト（スプリント）の実施を繰り返し、その経過や個別のプロジェクト（スプリント）の結果を評価しながら、次第に目標を明らかにする。その目標を達成するために複数のプロジェクト（スプリント）を繰り返し実施する。

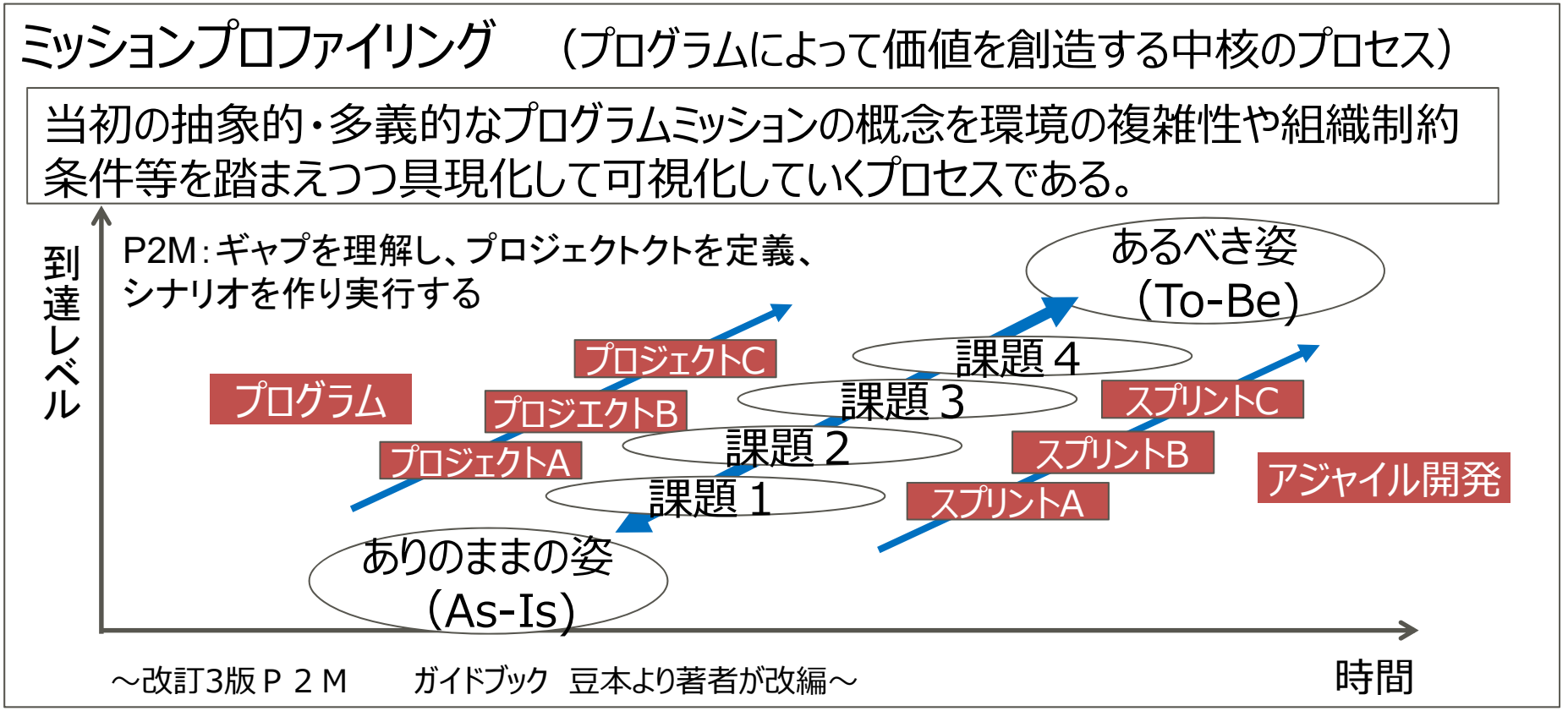


図7.13 P2Mとアジャイル開発

演習#3アジャイル開発への疑問の見直し

P23で挙げた、アジャイル開発に対する懸念・疑問を見直して下さい。

１．P23の懸念・疑問のキーワードを分類して下さい。
- 懸念・疑問が解消した。（×、横線を引いて下さい）
- 懸念・疑問が残っている（そのままにしてください）

２．新たな懸念・疑問を追加して下さい。
３．お隣の方と上記に関して話してください。
４．残った、懸念・疑問に優先順位１～３(１が優先度高)を振って下さい。

おわりに

チャールズ・R・ダーウィンは、**環境変化に適応できる者だけが生き残れる**名言を残した。**アジャイル開発は環境適応型**の開発方法論だ。短期間に高い効率で改善することができ、顧客にも受け入れられ、結果として**企業や組織は成長できる**。

ハードウェア資源では、資産が経費に変わるイノベーションが起きている。**「クラウド化」**である。**ソフトウェア資源でも**、利用者自らが所有せず、ITサービス企業と連携したアジャイル開発を採用したサービスとして提供することを受ける選択をすれば、**資産が経費となる**。

・大規模システム開発プロジェクト
・IOTの応用面で急拡大を続けるエンベデットシステム

へのアジャイル開発の適用が広がり、アジャイル開発を前提としたWebアーキテクチャーが整備され、**企業活動を大きく革新させる**ことになるだろう。