IPA Software Engineering Center
Information-technology Promotion Agency, Japan

アジャイル型開発における
プラクティス活用事例調査
調査概要報告書

独立行政法人 情報処理推進機構
平成25年3月

目次

目的

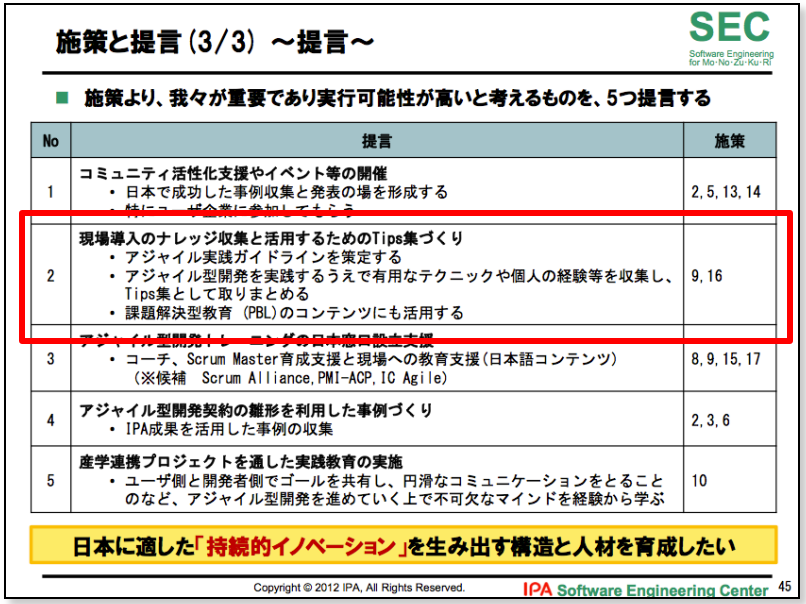
SEC
Software Engineering for Mo·No·Zu·Ku·Ri

■現場導入のナレッジ収集と活用するためのTips集づくり

- 2011年度の非ウォーターフォール型開発の調査の中で、アジャイル型開発を実践するうえで有用なテクニックや個人の経験を収集し、Tips集として取りまとめる必要があるという提言がなされた。
- 国内外の実際の非ウォーターフォール型開発の代表的な手法であるアジャイル型開発の適用プロジェクトにおける「プラクティス」（チーム運営からプログラミングまでの幅広い手法、活動、など）の具体的な活用事例を幅広く収集し、開発対象ソフトウェアの特徴や開発組織・プロジェクトの状況（以下「コンテキスト」という）の違いの観点で分類・整理した上で、ソフトウェア開発の現場で利用できるレファレンスのためのガイドとして取りまとめることとする。

■ 海外の書籍の翻訳に留まらず、日本の現場での実践事例を広く紹介することで、国内でのアジャイル型開発のより広い普及を目指す

- 日本で普及段階に入ったアジャイル開発をより広く普及させるため、「アジャイル開発に関心を持ちはじめて、自分で試してみようと思っている人」に向けたガイドを作成する。
- ガイドでは、プロジェクト独自の工夫や日本国内でアジャイル型開発を実践する際に留意しなければならない点を紹介する。



施策と提言（3/3）～提言～

SEC
Software Engineering for Mo·No·Zu·Ku·Ri

■ 施策より、我々が重要であり実行可能性が高いと考えるものを、5つ提言する

No	提言	施策
1	コミュニティ活性化支援やイベント等の開催 ・ 日本で成功した事例収集と発表の場を形成する ・ 特に[illegible]参加してもらう	2, 5, 13, 14
2	現場導入のナレッジ収集と活用するためのTips集づくり ・ アジャイル実践ガイドラインを策定する ・ アジャイル型開発を実践するうえで有用なテクニックや個人の経験等を収集し、Tips集として取りまとめる ・ 課題解決型教育（PBL）のコンテンツにも活用する	9, 16
3	アジャイル型開発トレーニングの日本語版設立支援 ・ コーチ、Scrum Master育成支援と現場への教育支援（日本語コンテンツ） （※候補　Scrum Alliance, PMI-ACP, IC Agile）	8, 9, 15, 17
4	アジャイル型開発契約の雛形を利用した事例づくり ・ IPA成果を活用した事例の収集	2, 3, 6
5	産学連携プロジェクトを通した実践教育の実施 ・ ユーザ側と開発者側でゴールを共有し、円滑なコミュニケーションをとることのなど、アジャイル型開発を進めていく上で不可欠なマインドを経験から学ぶ	10

日本に適した「持続的イノベーション」を生み出す構造と人材を育成したい

IPA Software Engineering Center 45

出典：非ウォーターフォール型開発の普及要因と適用領域の拡大に関する調査 調査概要報告書（IPA）

背景

SEC
Software Engineering for Mo·No·Zu·Ku·Ri

■ 世界的にアジャイル型開発が主流になっている

Forrester2010調査:「メインストリームとなったアジャイル型開発」

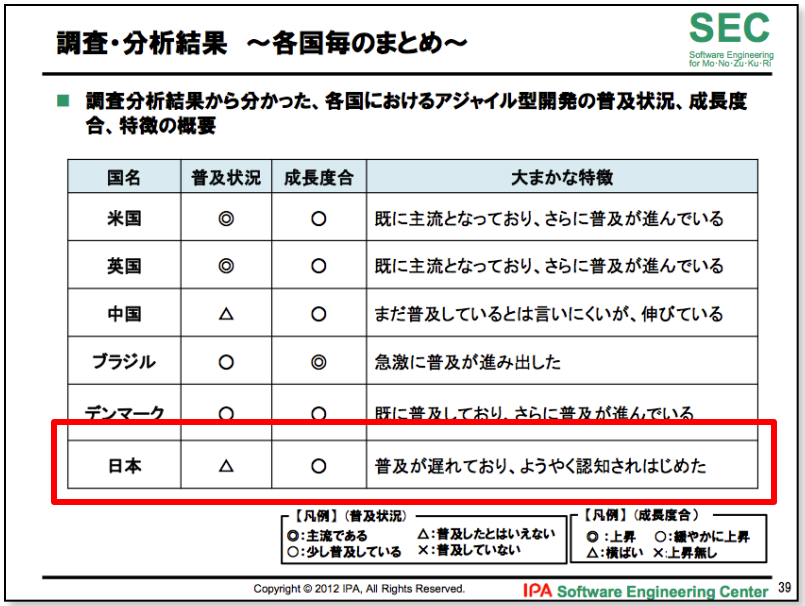
“2009年の第3四半期に実施した調査で、約35%の開発プロジェクトでアジャイル型開発手法を採用しているという結果が示されている。

… アジャイルと非アジャイルの技術とプラクティスを組み合わせてより大きな組織にあうようにハイブリッドにすることに苦闘している。”

(出典 eWeek 2010/1/22: が報じた、Forrester 2010: “AGILE DEVELOPMENT: MAINSTREAM ADOPTION HAS CHANGED AGILITY”)

※ 上記の調査で、アジャイル型開発はウォーターフォール型開発を初めて上回った。

■ 国際競争力の強化のため、世界的に主流になっているソフトウェア開発手法であるアジャイル型開発に日本でも取り組む必要がある。しかし、現状、日本では「普及が遅れており、ようやく認知されはじめた」段階であるとされている。



調査・分析結果 ～各国毎のまとめ～

SEC
Software Engineering for Mo·No·Zu·Ku·Ri

■ 調査分析結果から分かった、各国におけるアジャイル型開発の普及状況、成長度合、特徴の概要

国名	普及状況	成長度合	大まかな特徴
米国	◎	○	既に主流となっており、さらに普及が進んでいる
英国	◎	○	既に主流となっており、さらに普及が進んでいる
中国	△	○	まだ普及しているとは言いにくいが、伸びている
ブラジル	○	◎	急激に普及が進み出した
デンマーク	○	○	既に普及しており、さらに普及が進んでいる
日本	△	○	普及が遅れており、ようやく認知されはじめた

【凡例】(普及状況) ◎:主流である ○:少し普及している △:普及したとはいえない ×:普及していない

【凡例】(成長度合) ◎:上昇 ○:緩やかに上昇 △:横ばい ×:上昇無し

出典:非ウォーターフォール型開発の普及要因と適用領域の拡大に関する調査 調査概要報告書 (IPA)

課題

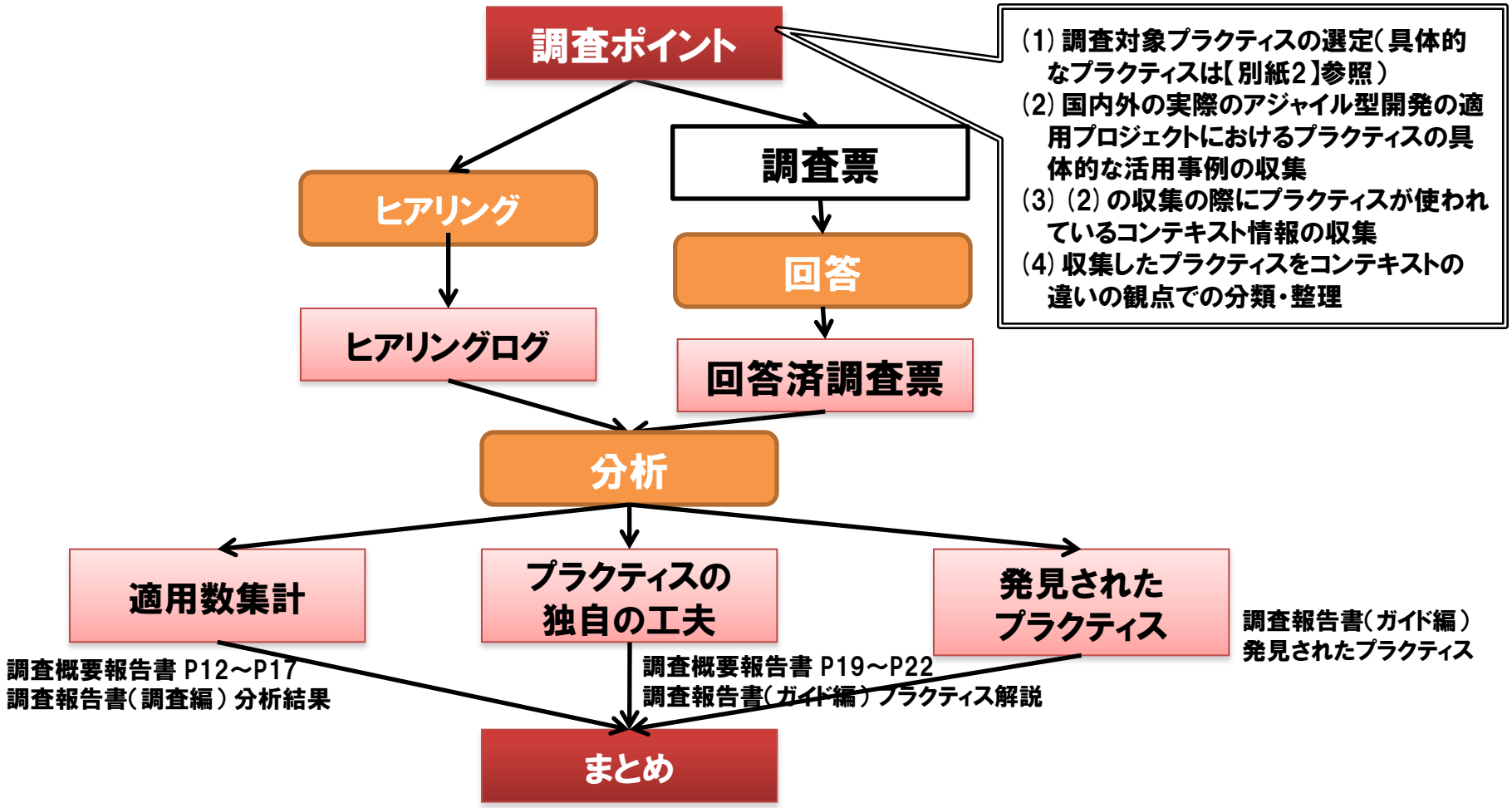
- 日本でアジャイル型開発の普及が阻害されている要因
  - 2011年度の非ウォーターフォール型開発の調査より、海外と国内では下記のような環境の違いがあることが分かった。

海外	国内
他国に比べて同一組織（企業など）内でソフトウェア開発が行われることが多い（米国）	顧客と開発チームのあいだに契約をはさむ受託開発が多い
オフショアであっても、顧客と開発チームがいつでもコミュニケーションをとれる環境を作ってる（米国−ブラジル）	契約の壁があったり物理的に離れていたりで、コミュニケーションがとりにくい
ユーザ企業にIT技術者が多い（米国）	IT技術者はSIerやソフトハウスに所属していることが多い

- 海外と国内でこのような環境の違いがあり、海外の書籍に書かれていることをそのまま実践しても、上手くいかないことがある。日本におけるアジャイル型開発の普及を図るため、本調査では、このような環境の違いを踏まえた国内のプラクティス利用例（独自の工夫・留意点）を紹介する。

調査報告書 目次

調査方法



<<プラクティスの選定基準>>
調査対象のプラクティスを選定するにあたり、以下の資料を参照した。
●『What Do We Know about Scientific Software Development 's Agile Practices?』(Slethold, M.T. 他, 2012)
●『CHAOS Manifesto』(Standish Group, 2011)
●『非ウォーターフォール型開発に関する調査』(IPA, 2009)
上記の資料に掲載されているプラクティスをすべて洗い出し、重複しているものや類似のものはまとめて扱うこととした。

調査対象 (1)

方針1：できるだけ多様なプロジェクトの事例を収集

方針2：国内の事例を収集

日本国内でアジャイル型開発を実践する際の工夫や留意点が重要

施策a） 下記の特徴があるプロジェクトの事例を含める。

中大規模適用プロジェクト

本調査では、プロジェクトに参加した人数が30名以上のプロジェクトについて調査する[※1]。

なお、「プロジェクトに参加した人数」はのべ人数ではなく、各自の参画した期間に係らずプロジェクトに参画した実人数をさすものとする。

複数手法の組み合わせプロジェクト

Scrum+XP、Scrum+WF(ウォーターフォール)のように複数の手法を組み合わせたプロジェクトについて調査する。

施策b） システム種別や契約形態もなるべく様々なものを収集対象とする。

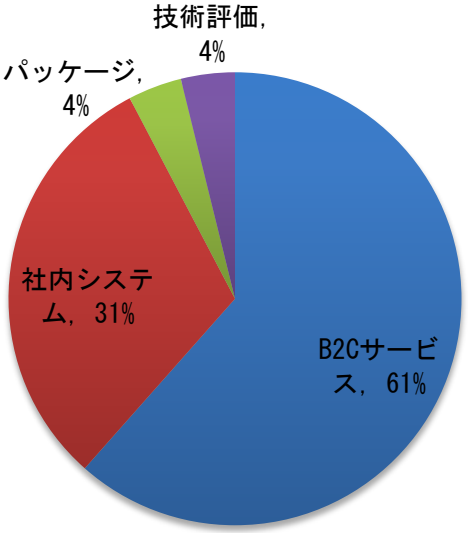
※1 人数が増えることによりコミュニケーション面に及ぼす影響を見たかったため、規模を表す基準として、チームの人数を採用した。
小規模 → プロジェクトに参加した人数が29名以下
中規模 → プロジェクトに参加した人数が30名以上99名以下
大規模 →プロジェクトに参加した人数が100名以上

調査対象（2）

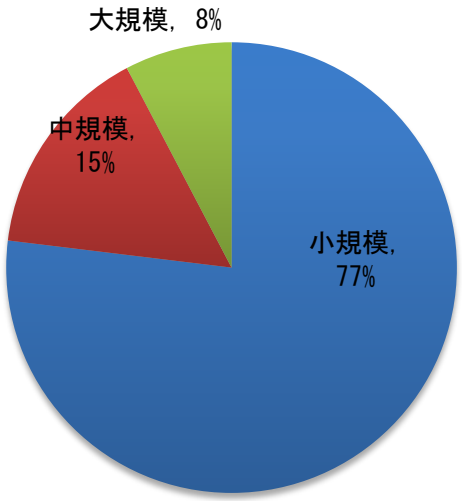
収集した事例をシステム種別・規模・手法・契約のそれぞれの切り口で分析した。

- 今回、調査対象としたプロジェクトの約60%がインターネット上のB2Cサービス開発にアジャイル型開発を適用している。
- 80%近くがチームメンバー数30名未満の小規模開発であった。
- Scrumの採用率が、他の手法と組み合わせている事例もあわせると、4分の3を超えていた。社内で様々な手法が乱立しているという例はあまりなく、会社全体でどの手法を使うのか統一している例が多かった。
- 半数が自社組織内に開発部隊を抱える形態（自社開発もしくは一部オフショアを含む自社開発）を採用していた。

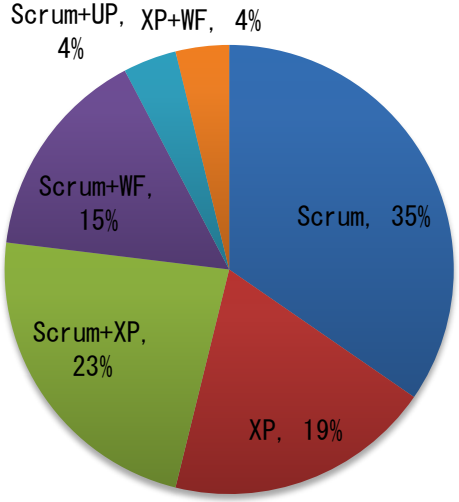
システム種別 （n=26）



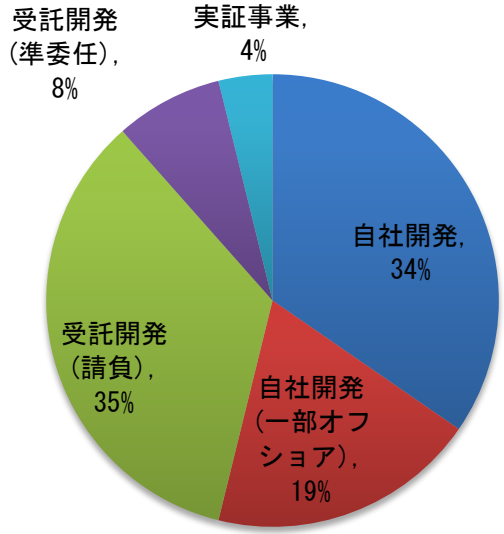
規模 （n=26）



手法 （n=26）

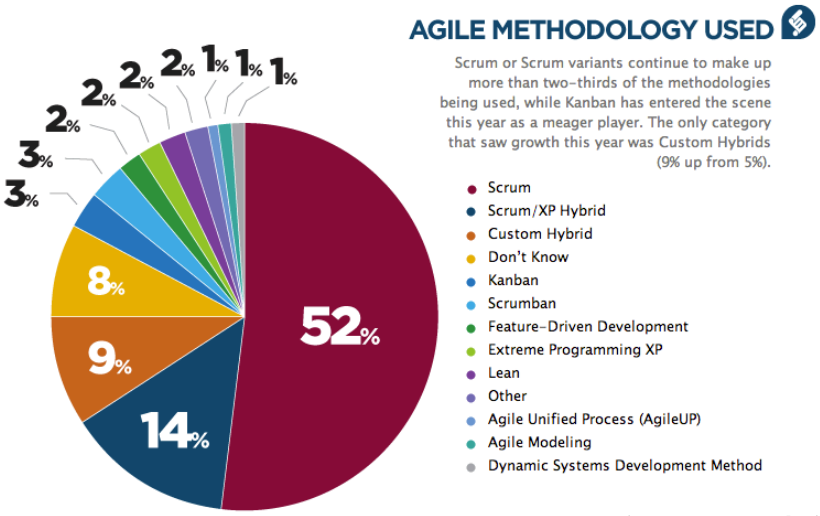


契約 （n=26）



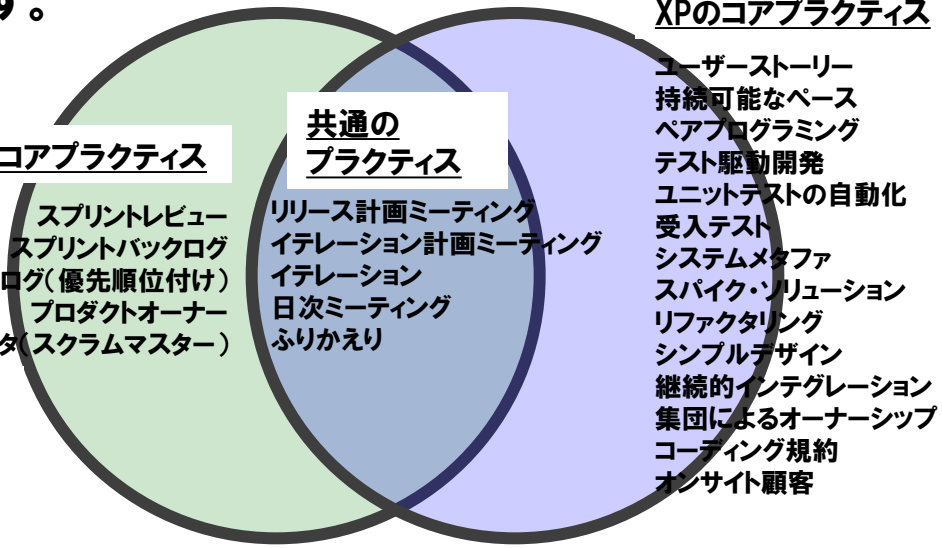
【参考】アジャイル型開発の手法

2011年にVersionOne社が行ったワールドワイドのアジャイル型開発実態調査で、最もよく用いられる手法として、Scrumおよび Scrum+XPの組み合わせがあげられており（合わせて全体の66%）、本調査でも調査対象事例として、ScrumとXPを採用している事例を中心に取り上げることにした。



出典：State of Agile Development Survey 2011（VersionOne社）

両アジャイル開発手法の特徴と代表的なプラクティスを以下に示す。



Scrumとは、マネジメント・プロセスをシンプルなフレームワークで提供する軽量開発手法

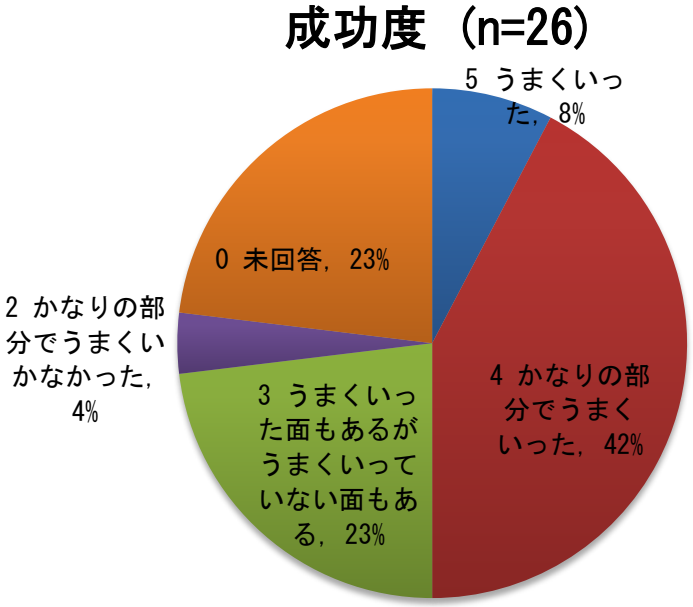
XPとは、エンジニアリング・プロセスとマネジメント・プロセスをバランスよく価値／原則／プラクティスというフォーマットで提供する開発手法

調査対象（3）

SEC
Software Engineering
for Mo·No·Zu·Ku·Ri

調査対象となるプロジェクトの成功度[※1]について、以下の5段階で回答を得た。

- 5：うまくいった
- 4：かなりの部分でうまくいった
- 3：うまくいった面もあるがうまくいっていない面もある
- 2：かなりの部分でうまくいかなかった
- 1：うまくいかなかった



「うまくいった」もしくは「かなりの部分でうまくいった」と回答した半数を占めるプロジェクトにおいては、主にプラクティスの有効な**利用例**を収集した。

また、「上手くいった面もあるがうまくいっていない面もある」、「かなりの部分でうまくいかなかった」に該当する約1/4の回答からは、主にプラクティスを活用する際の**留意点**を収集した。

※1 成功の尺度としては以下の観点をあげる企業が多かった。

- 顧客にとってのビジネス価値・利便性の高いシステムを実現できたか？
- 顧客からの要求の充足度が高かったか？顧客の望むシステムが実現できたか？
- 費用対効果が高く、ムダのない開発ができたか？
- 顧客満足度・開発者満足度が高かったか？
- 顧客の参画度が高かったか？顧客からの信頼を獲得できたか？
- メンテナンスしやすく、変更容易性の高いシステムが構築できたか？
- 品質・納期・コストは守れたか。生産性は向上したか？

事例概要

<<中大規模適用プロジェクトの事例>> 事例(5) C社

SEC
Software Engineering for Mo·No·Zu·Ku·Ri

プロフィール

既存のサービスのリプレイス開発。単純なサービスのリプレイスではなく、新しい要件も加えながら開発したいとの要望があり、C社から顧客にアジャイル型開発を提案して開始した。リプレイスといいながらも、顧客から要件を聞き出しながら開発を進めていった。要件が固められない部分のみアジャイル型開発を行い、要件が明らかな部分についてはウォーターフォール型開発を実施した。

システム種別	B2Cサービス
規模	中規模 開発者 32名 インフラ 4名 管理その他 23名 計 59名
手法	XP
契約	準委任契約（四半期毎に更新）
期間	2年
開発拠点	東京、地方を含めた3拠点

特徴的なプラクティス

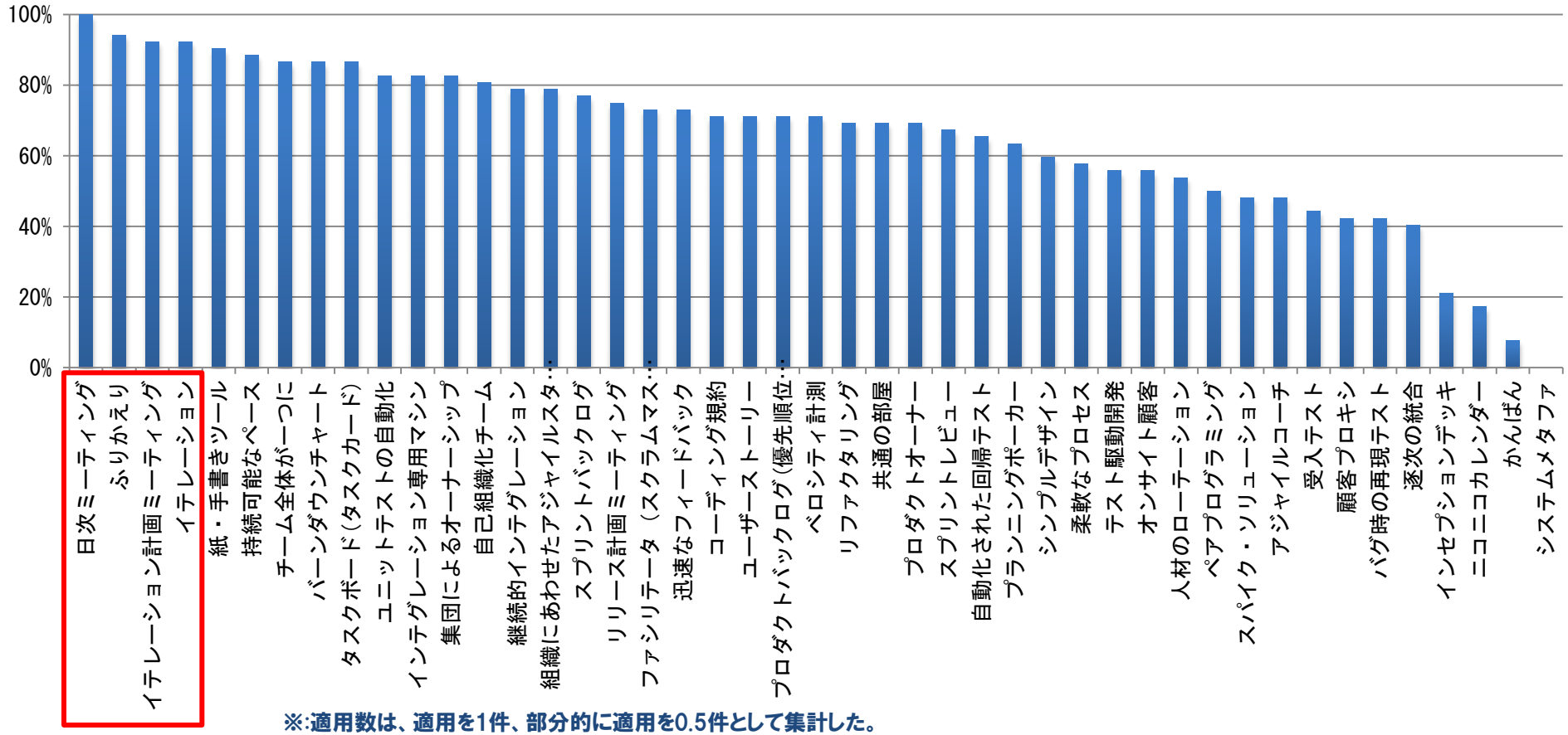
- **日次ミーティング**: 複数のチームが存在したため、二段階の構成で実施していた。(チーム間→チーム毎)。
- **ふりかえり**: チーム毎で実施した場合には、他のチームへの不満などばかりになってしまい機能しなかった。そのために、複数チームの混成で実施することで、問題へ集中するように意識を変えさせた。また、反復毎のふりかえりとは別に、四半期単位でも実施して大きな改善点について話しあった。
- **顧客プロキシ**: 当初は顧客に要件管理をしてもらっていたが、機能しなくなったため、C社の社員が顧客の会社へ出向して顧客プロキシとなり全面的に支援した。

適用プラクティス（全体）

SEC
Software Engineering
for Mo·No·Zu·Ku·Ri

日次ミーティング、ふりかえり、イテレーション計画ミーティング、イテレーションの順に適用率が高く、これらはアジャイル型開発を行う上でのほぼ必須のプラクティスであると言える。これらのプラクティスはScrumとXPに共通するプラクティスである（P9「【参考】アジャイル型開発の手法」）。

プラクティス適用率（n=26）



※:適用数は、適用を1件、部分的に適用を0.5件として集計した。
※ システムメタファは国内の26事例の中で活用されている事例はなかった。『ガイド編 プラクティス解説』では、海外の事例を調査した。

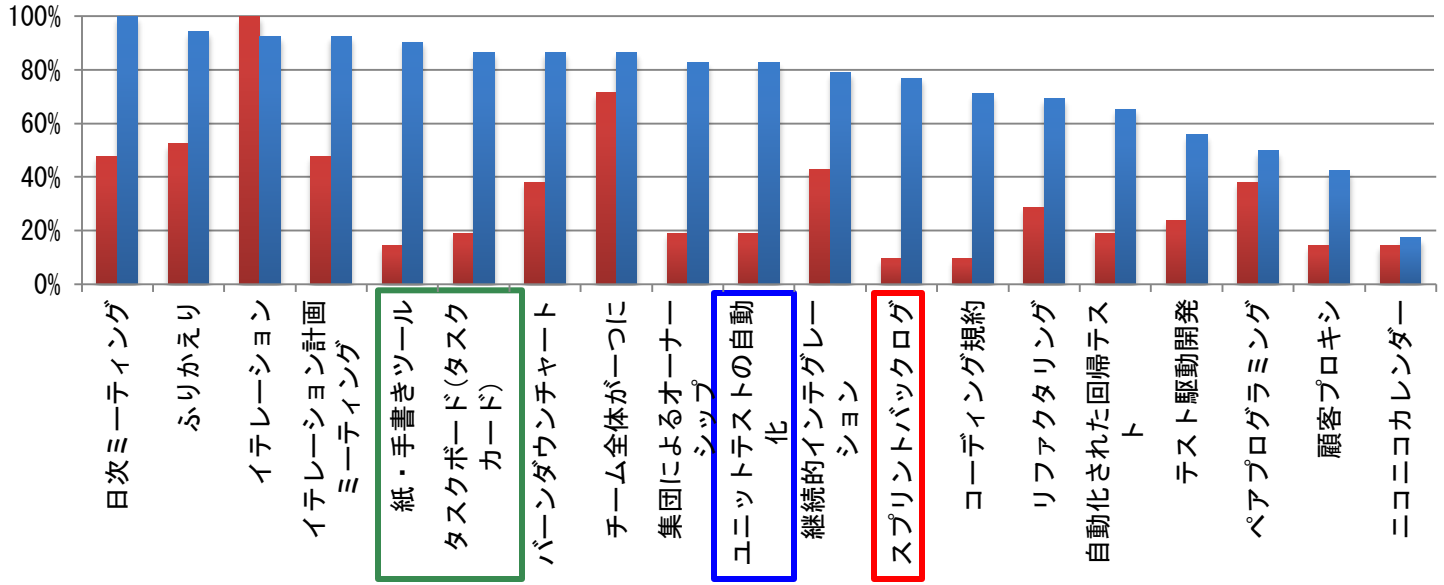
適用プラクティス（過年度調査との比較）

SEC
Software Engineering
for Mo·No·Zu·Ku·Ri

スプリントバックログはScrumのプラクティスであるが、2009年度調査から適用率が8倍以上、上昇している。これは、認定スクラムマスター研修が国内で開催されるようになったことにより、Scrumが普及したためであると思われる。

紙・手書きツールやタスクボード（タスクカード）についても、適用率が上昇している。これは、アジャイル型開発の認知度が高まり、社内にこれらのアナログツールを設置する障壁が減ったことや、事業者と開発者が密にコミュニケーションをとる必要が認識されたことにより、事業者と開発者が同一拠点で開発に取り組む事例が多く[※1]、アナログツールが活用される場面が増えたためであると思われる。

ユニットテストの自動化についても、2009年度から適用率が4倍以上、上昇している。これは、アジャイル型開発が一般的になるにつれ、プロセスの側面だけでなく、「技術・ツール」カテゴリのプラクティス[※2]を適切に実践することの重要性が認識されたためであると思われる。



※1 調査対象26事例中18事例が事業者の代表であるプロダクトオーナーと開発チームが同一拠点で開発

※2【別紙2】プラクティス一覧参照

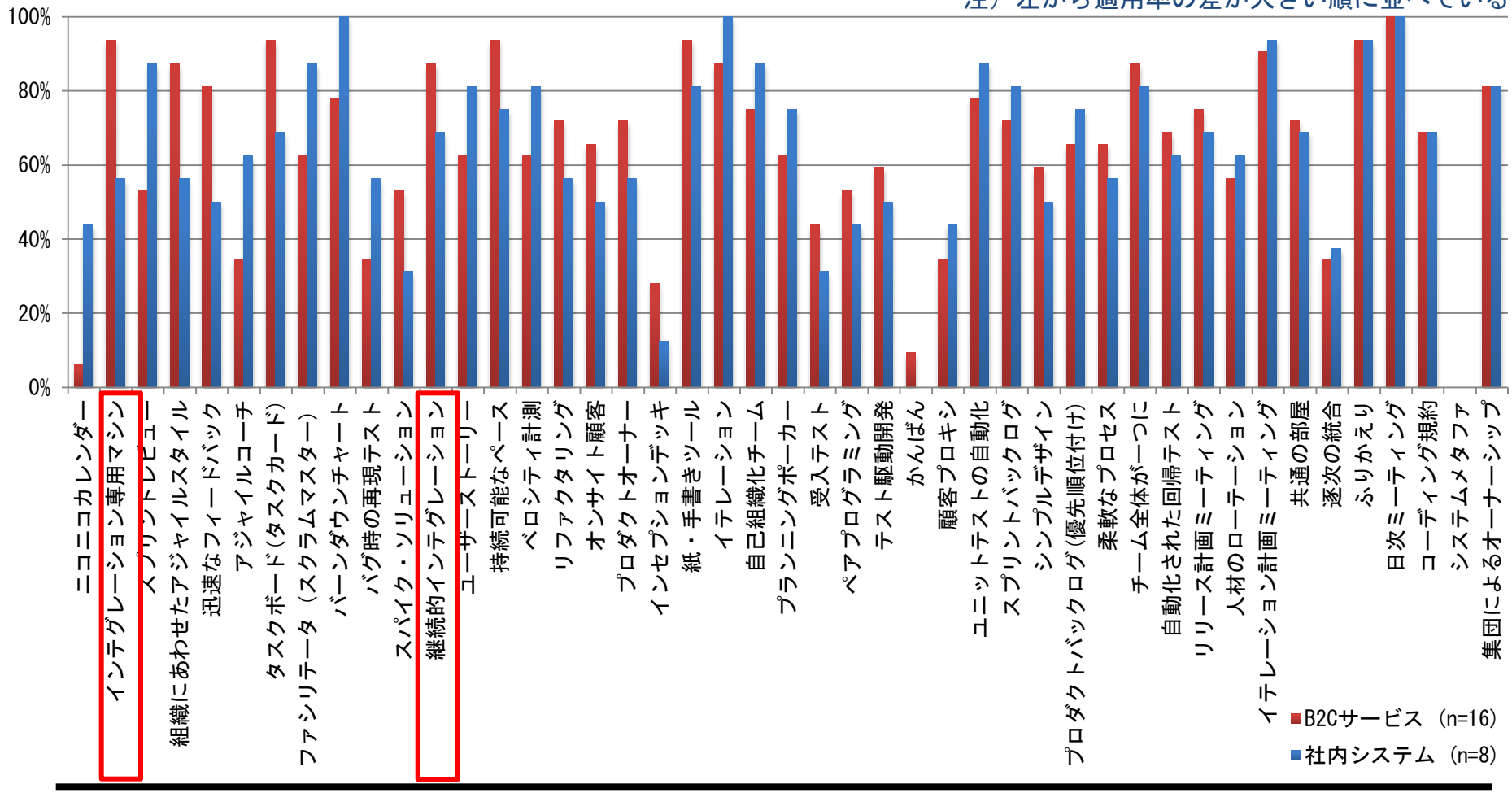
注）本調査と2009年度の調査では、調査対象とした事例の特性（システム種別、規模、手法、契約）が異なるため、単純に比較することはできないが、2009年度からの変化の傾向を示すために本節をもうけた。

適用プラクティス（システム種類別）

SEC
Software Engineering
for Mo·No·Zu·Ku·Ri

B2Cサービスでは、インテグレーション専用マシンや継続的インテグレーションが上位に入っている。これは、社内システムに比べてリリース後に変更が入る率が高く、迅速なフィードバックが求められるためであると思われる。

注）左から適用率の差が大きい順に並べている

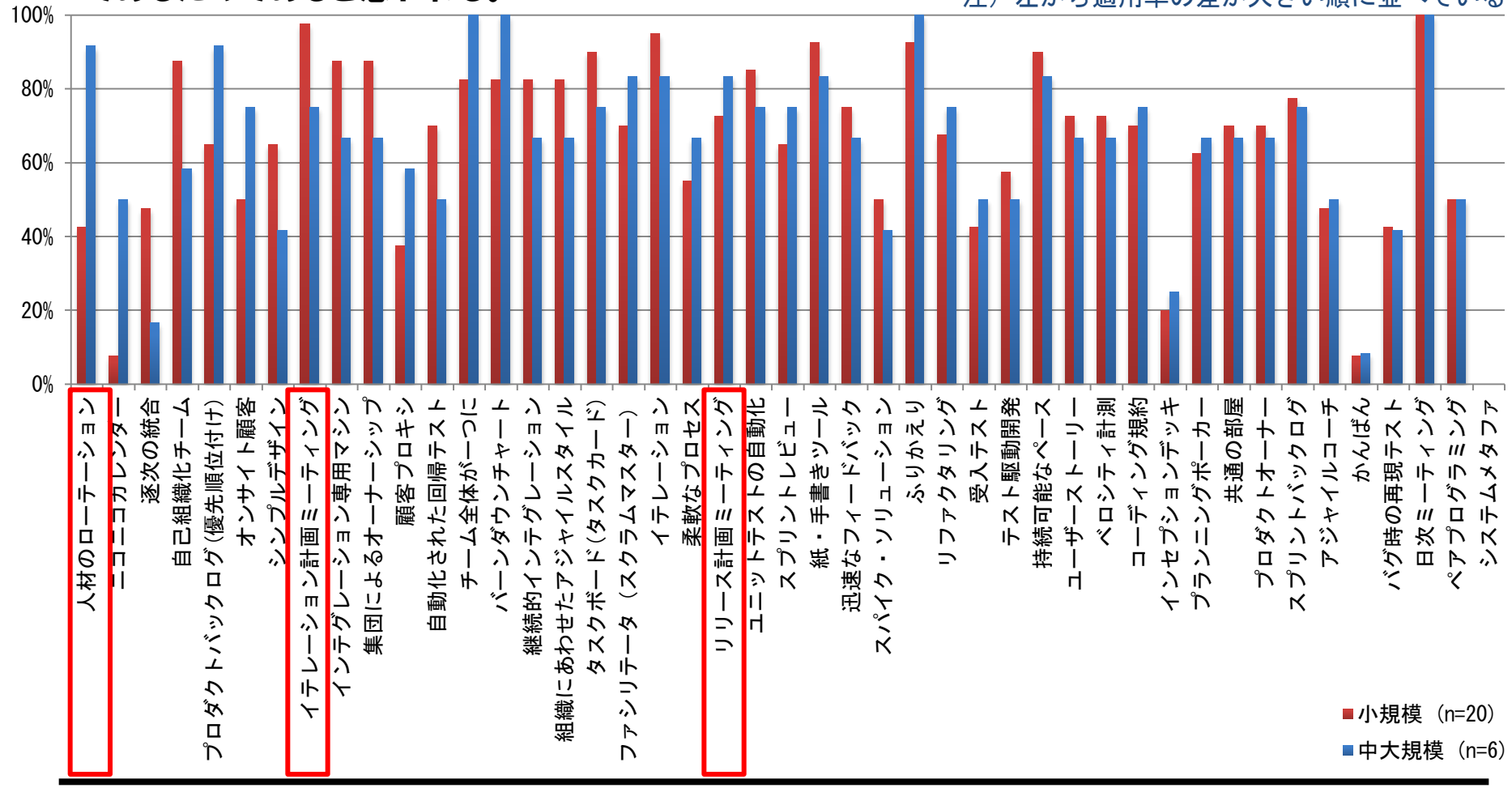


適用プラクティス（規模別）

SEC
Software Engineering for Mo·No·Zu·Ku·Ri

小規模ではイテレーション計画ミーティングがリリース計画ミーティングよりも適用率が高いのに対して、中大規模ではこれが逆転している。このことから、中大規模では中長期の計画がより重視されていることが分かる。中大規模では人材のローテーションが適用される率が高い。これは、中大規模プロジェクトにおいて、人材を流動的にして、ナレッジやノウハウをいかに流通させるかが課題であるためであると思われる。

注）左から適用率の差が大きい順に並べている

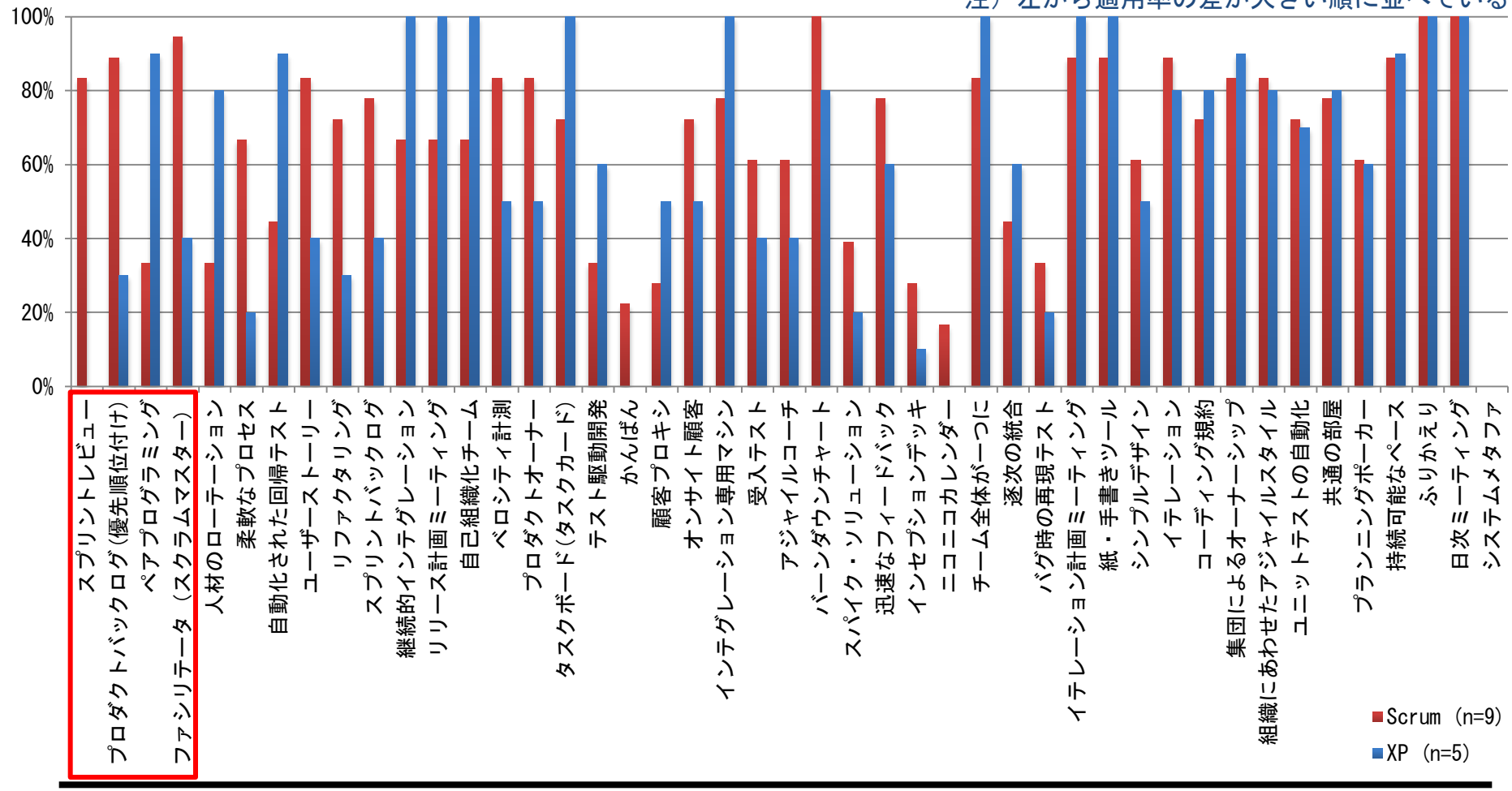


適用プラクティス（手法別）

SEC
Software Engineering
for Mo·No·Zu·Ku·Ri

Scrumプロジェクトではスプリントレビュー**や**ファシリテータ（スクラムマスター）**、**プロダクトバックログ（優先順位付け）**といったScrum発祥のプラクティスの適用率が高くなっている。一方、XPプロジェクトではXP発祥のプラクティスである**ペアプログラミング**の適用率が高くなっている（XPでの適用率は90%、Scrumでは33%）。**

注）左から適用率の差が大きい順に並べている

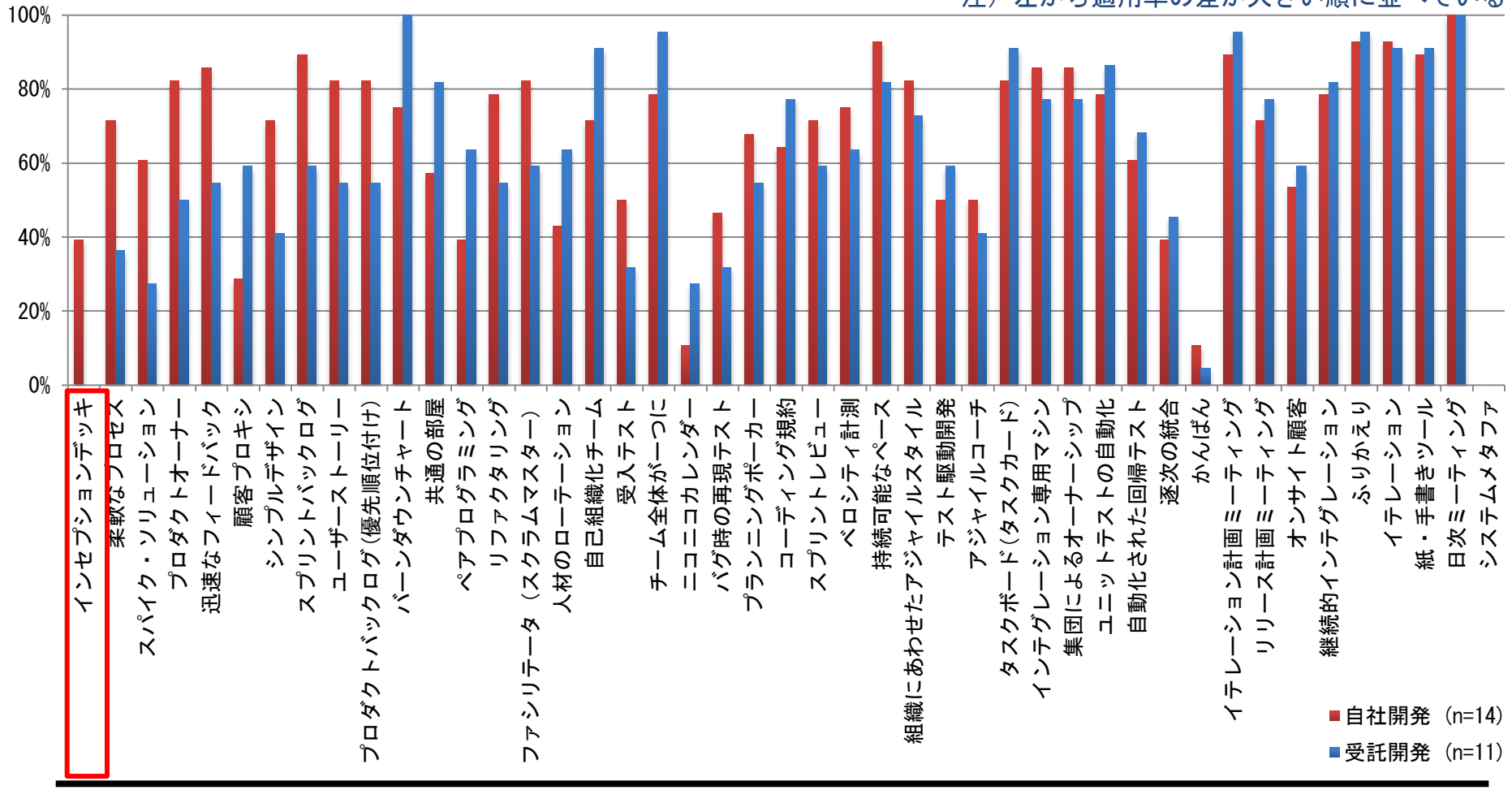


適用プラクティス（契約別）

SEC
Software Engineering
for Mo·No·Zu·Ku·Ri

受託開発では10の質問によりプロジェクトのビジョンやゴールを明らかにするインセプションデッキを使っている事例は皆無であったことに関して、受託開発では、インセプションデッキのような手法は開発者から事業者への押し売りになりやすく、理解のある事業者の下でしか実践できないという傾向があるものと思われる。

注）左から適用率の差が大きい順に並べている

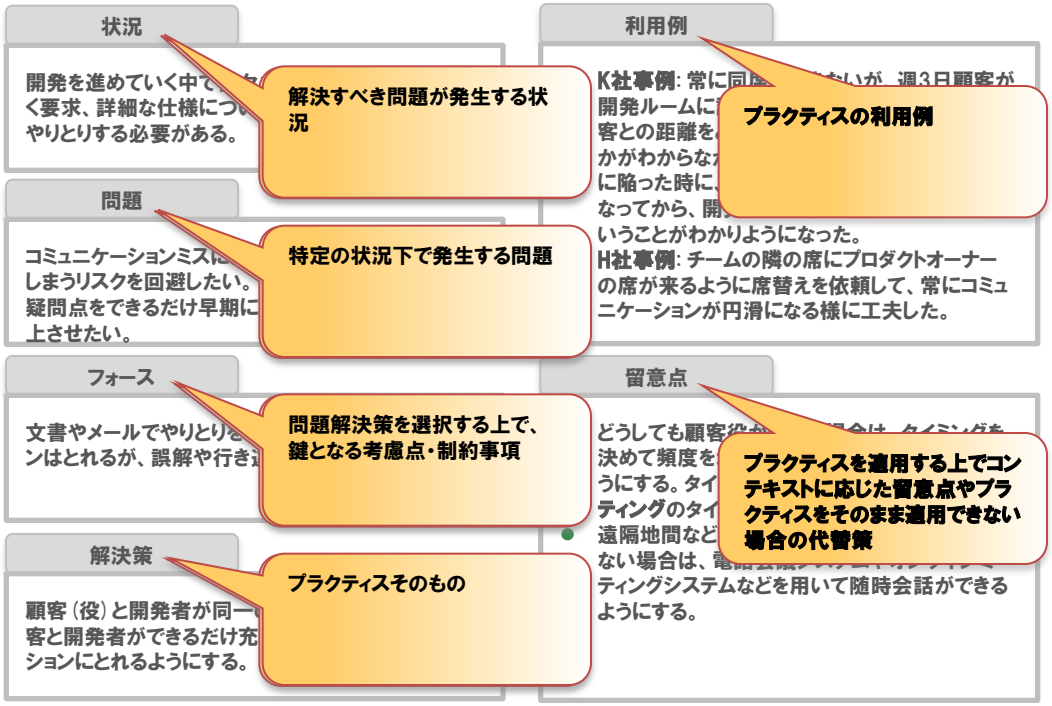


プラクティスの実態

適用事例数の多い上位3プラクティスについて、プラクティスの実態（どのようなコンテキストでどのように使われていたか）を示す。

1. 日次ミーティング
2. ふりかえり
3. イテレーション計画ミーティング

なお、説明には以下のフォーマットを用いた。



プラクティスの実態 - 日次ミーティング

SEC
Software Engineering for Mo·No·Zu·Ku·Ri

状況

チームは、プロジェクトのタスクをこなすためにほとんどの時間を使い、状況や情報の共有のために取れる時間がほとんどない。

問題

情報の共有遅れが問題を大きくする。
情報共有の時間が取れないまま、状況認識と問題対処への判断が遅れると、問題が大きくなるなど、より深刻な状況を招いてしまう。

フォース

関係者が多忙なため、情報共有のための時間が取れない。
情報共有の間隔が空いてしまうと、情報量が増え、共有に必要な時間が余分にかかる。

解決策

必要な情報を短い時間で毎日共有する。
関係者が長時間、時間を取れないようであれば、短い時間（15分を目安に）で済むように、共有を必要な情報に絞る。

利用例

- **事例（9）**：遠隔地にいるメンバーも日次ミーティングに参加するため、チャットツールや電話会議システムを利用した。
- **事例（17）**：一日3回（朝、昼、夕）10分程度のミーティングを実施。問題を報告/解決するためのリズムが開発メンバー全員に浸透して、短期での問題提起ができている。

留意点

- 必ずしも朝の時間帯にこだわらず、関係者が集まりやすい時間帯に開催する（例えば、終業近い時間帯に開催する夕会）。

プラクティスの実態 - ふりかえり

状況

イテレーション毎に、チームは動くソフトウェアとして成果を出そうとしている。イテレーションを繰り返して、チームはソフトウェアを開発していく。

問題

開発チームは、そこに集まったメンバーにとって最適な開発プロセスを、最初から実践することはできない。

フォース

イテレーションでの開発はうまくいくこともあるが、うまくいかないこともある

解決策

反復内で実施したことを、反復の最後にチームでふりかえり、開発プロセス、コミュニケーション、その他様々な活動をよりよくする改善案をチームで考え実施する機会を設ける。

利用例

- **事例 (25)**：当初はKPT[※1]を用いてふりかえりを行なっていたが、ファシリテータの技量にふりかえりの質が依存してしまう、声の大きいメンバーに影響を受けてしまうことに気づいた。そのため、意見を集めるやり方として、555 (Triple Nickels) [※2]を用いることにした。

留意点

- ふりかえりにチームが慣れていない場合は進行で各人の意見をうまく引出すようにしないとうまくいかない。
- 問題点を糾弾する場にしてしまうと、改善すべき点を積極的に話し合えない場になってしまう。
- 改善案を出しても、実際に実行可能なレベルの具体的なアクションになっていないと実施されない。

※1 メンバー全員でKepp（よかったこと・続けたいこと）、Problem（問題・困っていること）、Try（改善したいこと・チャレンジしたいこと）を出し合い、チームの改善を促す手法。

※2 アクションや提案に対するアイデアを出すための手法。5人程度のグループで、各人が5分間ブレインストーミングをしてアイデアを書き出す。5分経過したら紙を隣の人にまわし、新しいアイデアを書き加える。

プラクティスの実態 - イテレーション計画ミーティング

SEC
Software Engineering for Mo·No·Zu·Ku·Ri

状況

開発を一定期間のサイクル（イテレーション）で繰り返し行っている。
プロダクトバックログの内容をチームとプロダクトオーナーのあいだで合意している。

問題

リリース計画は遠い未来の目標のため、それだけではイテレーションで何をどのように開発すれば良いか分からない。

フォース

ユーザーストーリーのまま、イテレーションの詳細な計画を立て、開発を進めていくのは難しい。

解決策

イテレーションで開発するユーザーストーリーと、その完了までに必要なタスクおよびタスクの見積りを洗い出すミーティングを開く。

利用例

- G社事例（9）：ペーパープロトタイピング[※1]を用いたUIデザインの共有と受け入れ条件の確認をイテレーション計画ミーティングで行っていた。そのため、計画にはかなり時間を要していたが、見積りの前提が具体的になったため、見積り作業時間の削減に繋がった。

留意点

- 見積りに関してチームが水増しする懸念を持つかもしれないが、チームを信じるべきである。プロジェクトの目的を理解したチームは、見積りが大きく外れるようであれば、自らその原因を分析し、次の見積りに活かすはずである。

※1 紙などを使った試作品でユーザビリティテストを行うこと。

活用のポイント（1）

SEC
Software Engineering
for Mo･No･Zu･Ku･Ri

（1）短納期、開発期間が短い

開発対象のボリュームに比して、開発期間が短い場合、チームの開発速度を計測し、そのスピード感で、予定している開発量が期限内に完了するのか、常に点検する必要があるため、「ベロシティ計測」と、「バーンダウンチャート」を活用する。
ベロシティ計測は、関係者であるプロダクトオーナーが理解できる基準で計測する必要がある（H社事例（11））。バーンダウンチャートは、関係者と定期的に共有する機会を設けることが活用のポイントである（B社事例（2）、J社事例（17）（18））。

（2）スコープの変動が激しい

開発中に要求の変更が頻繁に発生するプロジェクトでは、チームが扱う要求の全体像と状態、直近のイテレーションで何を開発するかが分かっていて、柔軟に優先順位を変えられる必要があるため、「プロダクトバックログ（優先順位付け）」、「スプリントバックログ」および「プロダクトオーナー」を活用する。プロダクトバックログ（優先順位付け）は、イテレーション毎に整理を行い、チーム全員で優先順位と内容を合意すると良い（B社事例（2））。
プロダクトオーナーは、業務や全社的に全体最適となる判断を行うこと（G社事例（10））。

（3）求められる品質が高い

品質要求が高いプロジェクトでは、テストに関するプラクティスである「自動化された回帰テスト」、「ユニットテストの自動化」を活用する。
自動化された回帰テストやユニットテストの自動化は、プロジェクトの初期段階で、実施有無、実施のための取り決め、使用ツールを検討しておくことがポイントである。これを後回しにすると、必ず機能開発が優先され、自動化にたどりつかない（B社事例（2））。

活用のポイント（2）

SEC
Software Engineering
for Mo･No･Zu･Ku･Ri

（4）コスト要求が厳しい

必要のないものを作るムダをなくし、必要なものをより素早く提供することがROI（費用対効果）の向上につながり、コスト要求に応えることができる。そのためには、的確に顧客の要求を把握し、認識の相違をなくす必要があるため、「プロダクトバックログ（優先順位付け）」を活用する。
また、開発機能がプロダクトオーナーの意図通りになっているかの検証のために、「受入テスト」を活用する。「オンサイト顧客」には、優先順位や仕様の確認がその場で確認することができ、迅速に方針を決められるというメリットがある（K社事例（20））。

（5）チームメンバーのスキルが未成熟

スキル的に未成熟なメンバーが成長していく機会として、プロジェクトを計画する必要があるため、「ペアプログラミング」と「ふりかえり」を活用する。
ペアプログラミングは、ベテランとメンバーが一緒に仕事をすることで、技術的な指導を行うのに適したプラクティスである（C社事例（4））。
ふりかえりは、メンバーの成長の機会として捉えることができる。ふりかえりのやり方自体も見直しながらチームに適したやり方を模索すると良い（E社事例（6））。

（6）チームにとって初めての技術領域や業務知識を扱う

プロダクトの背景にある業界の知識や、要求の理解と実装に必要な業務知識の獲得が必要となるため、「スパイク・ソリューション」と「システムメタファ」を活用する。
スパイク・ソリューションを適用することは、リスクとなりそうな技術課題について、プロジェクトの初期段階で実験的に小さく試しておくことであり、チームとプロジェクトを後々助けることに繋がる（C社事例（4））。システムメタファは、開発者にとって、なじみの薄い業務知識を理解する手段として、有効と考えられる。

活用のポイント（3）

SEC
Software Engineering
for Mo･No･Zu･Ku･Ri

（7）初めてチームを組むメンバーが多い

初めてチームを組むメンバーが多い場合、チームが向かう方向を明確にすることと、チームビルディングが必要となるため、「インセプションデッキ」や「ニコニコカレンダー」を活用する。
インセプションデッキは、作成を通じて、プロジェクトの目的や目標が明らかとなる（B社事例（1））。
ニコニコカレンダーは、メンバーの感情や状況を可視化し、チームメンバーのことを知ることがポイントになる（E社事例（6））。

（8）オフショアなど分散開発を行う

プロダクトオーナーと開発チームが別の拠点にいる場合、オンラインでのコミュニケーション手段を検討し、頻繁にコミュニケーションが取れるようにする必要があるため、「日次ミーティング」や「顧客プロキシ」を活用する。
TV会議システムを使った日次ミーティングは、離れた者同士が毎日顔を合わせる機会として、ぜひ活用するべきである（G社事例（9））。顧客プロキシは、分散した環境下でも、迅速なフィードバックが得られる工夫を取らなければならない。

（9）初めてアジャイル開発に取り組む

初めてアジャイル開発に取り組む際には、書籍や文書だけではなく人から人にやり方を伝えることが有効であるため、社内にアジャイル型開発に取り組んだ経験のある人がいる場合はその人に、社内にない場合は、社外からアジャイルコーチを頼んで導入の手伝いをしてもらうのがよい。初めて取り組む場合は、イテレーション期間を短かくした上で、ふりかえりの中で改善点をチームで考え実行していくことが不可欠となる。

まとめ（1）

- 日次ミーティング、ふりかえり、イテレーション計画ミーティング、イテレーションの順に適用率が高く、これらはアジャイル型開発を行う上でのほぼ必須のプラクティスであると言える[※1]。これらのプラクティスはScrumとXPに共通するプラクティスである[※2]。
- 2009年度の調査結果と比較すると、スプリントバックログ、紙・手書きツール、タスクボード（タスクカード）、ユニットテストの自動化の適用率が4倍以上、上昇している[※3]。これは、ここ数年の日本国内でのアジャイルの動向を如実に表している。
  - 認定スクラムマスター研修が国内で開催されるようになったことにより、Scrumの認知度が向上した。
  - 事業者と開発者が密にコミュニケーションをとる必要が認識されたことにより、事業者と開発者が同一拠点で開発に取り組む割合が多く[※4]、紙・手書きツールやタスクボードなどのアナログのツールが活用される場面が増えた。
  - プロセスの側面だけでなく、「技術・ツール」カテゴリのプラクティスを適切に実践することの重要性が認識された。
- 現状、日本では「普及が遅れており、ようやく認知されはじめた」段階[※5]とされているが、アジャイル型開発を適用している企業を調査してみると2009年度の調査と比較して、プラクティスの適用率は総じて上昇している[※3]。今回、調査対象とした企業では、プラクティスを取り入れ、実践するまでに至っていた。
  - 一方ではアジャイル型開発を導入しさえすれば「生産性、品質、コスト」に関する問題が解決するという誤った認識も生まれ始めていることから、プラクティスへの正しい理解を広めると共に、「アジャイルコーチ」（外部からの変化）と「組織にあわせたアジャイルスタイル」（内部からの変化）をバランス良く取り入れながら、「正しい普及」活動を進めてゆくことが今後の課題となる。

※1 P12「適用プラクティス（全体）」　※2 P9「【参考】アジャイル型開発の手法」　※3 P13「適用プラクティス（過年度調査との比較）」
※4 調査対象26事例中18事例が事業者の代表であるプロダクトオーナーと開発チームが同一拠点で開発　※5 P3「背景」

まとめ (2)

- プロジェクト特性(システム種別、規模、手法、契約)の中でも特に規模の違いが活用するプラクティスに及ぼす影響が大きい。反面、契約やシステム種別の違いは他のプロジェクト特性に比べると、あまりプラクティスに影響を与えていないことが分かった[※1]。
  - 規模別(小規模と中大規模)で適用率に最も大きな差があったプラクティス(人材のローテーション)はその差が49.5%、契約別(自社開発と受託開発)で適用率に最も大きな差があったプラクティス(インセプションデッキ)では39.3%、システム種別(B2Cサービスと社内システム)で適用率に最も大きな差があったプラクティス(ニコニコカレンダーとインテグレーション専用マシン)では37.5%であった。
- 実際、契約の違いはあまりプラクティスに影響を与えていないが、海外と国内の環境の違い[※2]を踏まえると、日本の環境では特に以下のプラクティスを活用する必要がある。
  - ファシリテータ(スクラムマスター):契約絡みの問題のために、アジャイル型開発が進めにくくならないように、顧客や開発者とは別の組織に属する第三者を立てる。
  - 顧客プロキシ:顧客と直接対話するのが難しい場合は顧客プロキシを立てる。
  - 共通の部屋:物理的に近くで仕事をする、出来ない場合は電話会議ツールなどを活用する。
- 調査対象の事例それぞれの状況(コンテキスト)にあわせた工夫をしていることが分かった。現場での独自の工夫ができる余地があるところがアジャイル型開発の利点であり、本調査報告書のプラクティスを適用する状況・問題を参考にし、「なぜそのプラクティスを適用するのか」を考えて、それぞれの現場にあわせてカスタマイズして活用すべきである。
  - 独自の工夫を収集したところ、26事例で180以上の工夫が見つかった[※3]。

※1 P14～P17「適用プラクティス」　※2 P4「課題」　※3ガイド編「付録1」を参照

提言

日本でアジャイル開発をより広く普及させる上での施策を示す。

ファシリテータ・顧客プロキシができる人材の育成

「顧客と開発チームのあいだに契約をはさむ受託開発が多い」「契約の壁があったり物理的に離れていたりで、コミュニケーションがとりにくい」といった環境では、ファシリテータ（スクラムマスター）や顧客プロキシといった、顧客と開発チームのあいだに立って両者の調整をする役割が非常に重要になる。

プラクティス（パターン）を現場から発信できる場づくり

今回は調査としてプラクティスのパターンを収集したが、時間的制約があったため、現場の状況情報などの吸い上げが充分ではなかった。日本独自の工夫を収集するには、トップダウンによる調査だけでなく、もっと日常的に現場の開発者が自分達の工夫を言葉にして、それを社内・社外関わらず共有しながら、現場の知を交流させて、新しい知を生成していく必要がある。そうすることによって、海外からの輸入されたプラクティスではなく、日本、あるいは個々の業界独自のプラクティスが生まれて流通することで、国内産業の競争力の向上に寄与できると考える。

【別紙1】事例一覧（1）

SEC
Software Engineering for Mo·No·Zu·Ku·Ri

調査先	事例番号	採用手法[※1]	特徴	システム種別	契約関係[※2]	開発言語
A社	0	Scrum+XP		B2Cサービス（広告配信）	自社開発	Java, PHP, Perl
	1	Scrum+XP		B2Cサービス（広告配信）	自社開発	Ruby
B社	2	Scrum+XP		B2Cサービス（SNS）	自社開発	Java
	3	Scrum+XP		B2Cサービス（メール配信）	自社開発	Java
C社	4	XP+WF	中規模	B2Cサービス（メール配信）	受託開発（準委任）	Java
D社	5	XP		B2Cサービス（SNS）	自社開発	Java, PHP, Ruby
E社	6	Scrum	初導入	社内システム	自社開発	C#
	7	Scrum+WF	中規模	社内システム	受託開発（請負）	Java, COBOL
F社	8	Scrum+WF	中規模	社内システム	自社開発	C#
G社	9	Scrum+XP	初導入	社内システム	実証事業	Ruby
	10	Scrum+XP		社内システム	受託開発（請負）	Ruby
H社	11	Scrum		B2Cサービス（音楽配信）	自社開発 + オフショア（準委任）	Java, C#, Objective-C
	12	Scrum		B2Cサービス（エンターテイメント）	自社開発 + オフショア（準委任）	Java, C#, Objective-C
	13	Scrum		社内システム	自社開発 + オフショア（準委任）	Java
	14	Scrum		B2Cサービス（ヘルスケア）	自社開発 + オフショア（準委任）	C#

【別紙1】事例一覧（2）

SEC
Software Engineering for Mo·No·Zu·Ku·Ri

調査先	事例番号	採用手法[※1]	特徴	システム種別	契約関係[※2]	開発言語
I社	15	Scrum	中規模（組織展開）	B2Cサービス（広告配信）	自社開発	Java, Objective-C
J社	16	XP		B2Cサービス（スマートフォンアプリ）	受託開発（請負）	Java
	17	XP		B2Cサービス（クラウド基盤）	受託開発（請負）	Java
	18	XP		B2Cサービス（クラウド基盤）	受託開発（請負）	Java
	19	XP		B2Cサービス（PaaS）	受託開発（請負）	Java
K社	20	Scrum		B2Cサービス（ECサイト）	受託開発（請負）	PHP
L社	21	Scrum+UP		社内システム	受託開発（請負）	Java
	22	Scrum+WF	大規模	社内システム	受託開発（準委任）	Java
	23	Scrum+WF		技術評価	受託開発（請負）	Java
	24	Scrum		パッケージ	自社開発 + オフショア（請負）	C#
M社	25	Scrum	大規模（組織展開）	B2Cサービス（ソーシャルゲーム）	自社開発	Perl

中大規模（30名以上）:6件

初導入:2件

全26事例

※1:XP:エクストリームプログラミング、Scrum:スクラム、
WF:ウォーターフォール、UP:統一プロセス、
もしくは、これらの手法の組み合わせ

※2:自社開発 → 自社組織内に開発部隊あり、一部パートナー（派遣）
受託開発 → 自社組織内に開発部隊なし、外部ベンダに発注している

【別紙2】プラクティス一覧（1）

SEC
Software Engineering
for Mo·No·Zu·Ku·Ri

カテゴリ	サブカテゴリ	プラクティス	説明
プロセス・プロダクト	プロセス	リリース計画ミーティング	プロダクトリリースのためのリリース計画ミーティング
		イテレーション計画ミーティング	イテレーション（スプリント）ごとのリリース計画やアクティビティなどを計画するミーティング。
		イテレーション	ゴールや結果にアプローチするプロセスをくり返すこと。
		プランニングポーカー	スプリント計画時のタスクを見積もるためのプランニングポーカー
		ベロシティ計測	プロジェクトベロシティの計測
		日次ミーティング	現在の問題を解決するための短いデイリーミーティング
		ふりかえり	前のスプリント（イテレーション）から学ぶためにふりかえる
		かんばん	ジャストインタイムの継続的なデリバリを強調した管理手法
		スプリントレビュー	完了した仕事を表明するスプリントレビューミーティング
		タスクボード（タスクカード）	ボードに貼られたメンバーが継続的に更新するタスク
		バーンダウンチャート	スプリント進捗をモニターするためのバーンダウンチャート
		柔軟なプロセス	状況や対応に対応できる柔軟なプロセスにしている、もしくは、プロセスを柔軟に変更している。
	プロダクト	ユーザーストーリー	要求についての会話を行うときの開発チームとプロダクトオーナーのあいだの合意事項
		スプリントバックログ	プロダクトオーナーとチーム間でのスプリントバックログへの相互コミットメント
		インセプションデッキ	10の質問によりプロジェクトの属性を明らかする
		プロダクトバックログ（優先順位付け）	プロダクトオーナーによる優先順位（プロダクトバックログ）の管理
	フィードバック	迅速なフィードバック	迅速なフィードバックを得られるような取り組みを行っている

【別紙2】プラクティス一覧（2）

SEC
Software Engineering
for Mo･No･Zu･Ku･Ri

カテゴリ	サブカテゴリ	プラクティス	説明
技術・ツール	設計開発	ペアプログラミング	すべての製品コードはペアプロで開発している
		自動化された回帰テスト	自動化された回帰テストを行っている
		テスト駆動開発	単体テストを書き、そのテストを通るようなコードを実装する
		ユニットテストの自動化	ユニットテストの自動化
		受入テスト	受入テストの実施と、その結果を公開している
		システムメタファ	関係者全員が、そのシステムがどのように動くかについて伝えることができるストーリー
		スパイク・ソリューション	リスクを軽減するために、かくれた問題を探索するための簡単なプログラム（スパイク・ソリューション）の試作
		リファクタリング	定常的なリファクタリング
		シンプルデザイン	設計をシンプルに保つ
		逐次の統合	一度に統合するコードはひとつだけとする
		継続的インテグレーション	継続的イテレーション、または頻繁なインテグレーション
		集団によるオーナーシップ	全員がすべてのコードに対して責任をもつ
		コーディング規約	同意された標準のためのコーディング規約
	障害対応	バグ時の再現テスト	バグが見つかったとき、そのテストがまず最初に作られる
	利用ツール	紙・手書きツール	ポストイット（付箋紙）やCRC（class-responsibility-collaboration）カードなどの使用

【別紙2】プラクティス一覧（3）

SEC
Software Engineering
for Mo･No･Zu･Ku･Ri

カレゴリ	サブカテゴリ	プラクティス	説明
チーム運営・組織・チーム環境	人	顧客プロキシ	要件や仕様をまとめるために顧客の業務に精通した顧客プロキシの設置
		オンサイト顧客	顧客といつでも／定期的にやりとりが可能である
		プロダクトオーナー	プロダクトオーナー役の設置
		ファシリテータ（スクラムマスター）	スクラムマスターによる開発プロセスとプラクティスのファシリテート
		アジャイルコーチ	アジャイルコーチがプロジェクトに参加している
		自己組織化チーム	チームメンバーがタスクに志願するなど自律的なチームになっている
		ニコニコカレンダー	ニコニコカレンダーを用いてメンバーの気持ちを見える化している
	進め方	持続可能なペース	継続的なペースで開発している
	組織導入	組織にあわせたアジャイルスタイル	組織にあった適切なアジャイルスタイルを用いるようにしている
	ファシリティ・ワークスペース	共通の部屋	オープンスペースがチームに与えられている
		チーム全体が一つに	チーム全員がひとつのゴールに向かうような取り組みを行っている
		人材のローテーション	多能工の育成などのため人材のローテーションを行っている
		インテグレーション専用マシン	特定のインテグレーション用コンピュータ

【別紙3】調査先企業概要（1）

SEC
Software Engineering
for Mo･No･Zu･Ku･Ri

A社	2事例	サービス事業者
•自社サービス開発の事例 •1人プロジェクトの事例と10人プロジェクトの事例の2事例を収集 •企業文化・風土がアジャイル型開発を後押し		

B社	2事例	サービス事業者
•自社サービス開発の事例 •チームの自主性を尊重し、楽しみながらプラクティスを工夫して実践		

C社	1事例	中小SIer・ソフトハウス
•サービスの受託開発の事例 •50人で1年半にわたる中規模開発		

【別紙3】調査先企業概要（2）

SEC
Software Engineering
for Mo･No･Zu･Ku･Ri

D社	1事例	サービス事業者

- 自社サービス開発の事例
- アジャイル型開発経験者がほぼいない10人以下のチーム
- エンジニアリングプラクティスの習得を重視

E社	2事例	ユーザー企業のシステム子会社

- 基幹システムリプレイス（中規模）の事例と社内システム開発の事例（計2事例）
- 中規模開発ではミニWFの中にアジャイル型開発のプラクティスを組み合わせて実践

F社	1事例	大手SIer

- 社内システムのScrum+TOC[※1]の事例
- WFからScrumへの移行
- インドとの分散開発
- 品質データなど豊富なデータを収集・分析し、社内に説明

※1 制約理論。エリヤフ・ゴールドラット博士が提唱した生産管理・改善のための理論体系。最適生産のためには工程全体のスケジュールをボトルネック工程の能力に合わせる必要があり、生産性向上のためにはボトルネック工程を重点的に改善すべきだとされている。

【別紙3】調査先企業概要（3）

SEC
Software Engineering
for Mo·No·Zu·Ku·Ri

G社	2事例	中小SIer・ソフトハウス

- 実証実験プロジェクトで初めてアジャイル型開発でシステム構築した事例
- アジャイルコーチが参画し、チームがプロセスとスキルを成長させていった
- 途中から分散拠点開発に移行
- 見える化ツールを顧客オフィスに携帯して持ち運び可能にした

H社	4事例	サービス事業者

- 自社サービス開発の事例
- 全社でScrumを導入
- 中国にオフショア

I社	1事例	サービス事業者

- 自社サービス開発（スマートフォン向け）の事例
- 横断部門が独自の開発プロセスを定義
- 比較的規模の小さいチーム（1～2ヶ月×5人）を十数チームまとめている

【別紙3】調査先企業概要（4）

SEC
Software Engineering
for Mo·No·Zu·Ku·Ri

J社	4事例	メーカーのシステム子会社

- 受託システム開発の事例
- 分散拠点（首都圏、東海地方）で開発
- 発注元の会社はアジャイル型開発に協力的であるが、実際にシステムを利用するエンドユーザーにはアジャイル型開発であることを伝えていない

K社	1事例	中小SIer・ソフトハウス

- ECサイトのリプレイスがうまく行かず途中からScrumを導入した事例
- 品質の作り込みと自己組織化によって顧客満足を向上させるのが目的
- 顧客からリファクタリングの工数を要求されるほどシステムの健康度を重視
- 教科書通りのアジャイルに囚われずに柔軟に改善している

L社	4事例	大手SIer

- 自社製品のScrum+オフショアの事例
- 顧客とパートナー企業がWFで、自社のみがScrumをやっているScrum+WFの事例
- 制約が多い一括請負の開発でやったScrum+UPの事例
- 大規模開発の事例（計4事例を収集）

【別紙3】調査先企業概要（5）

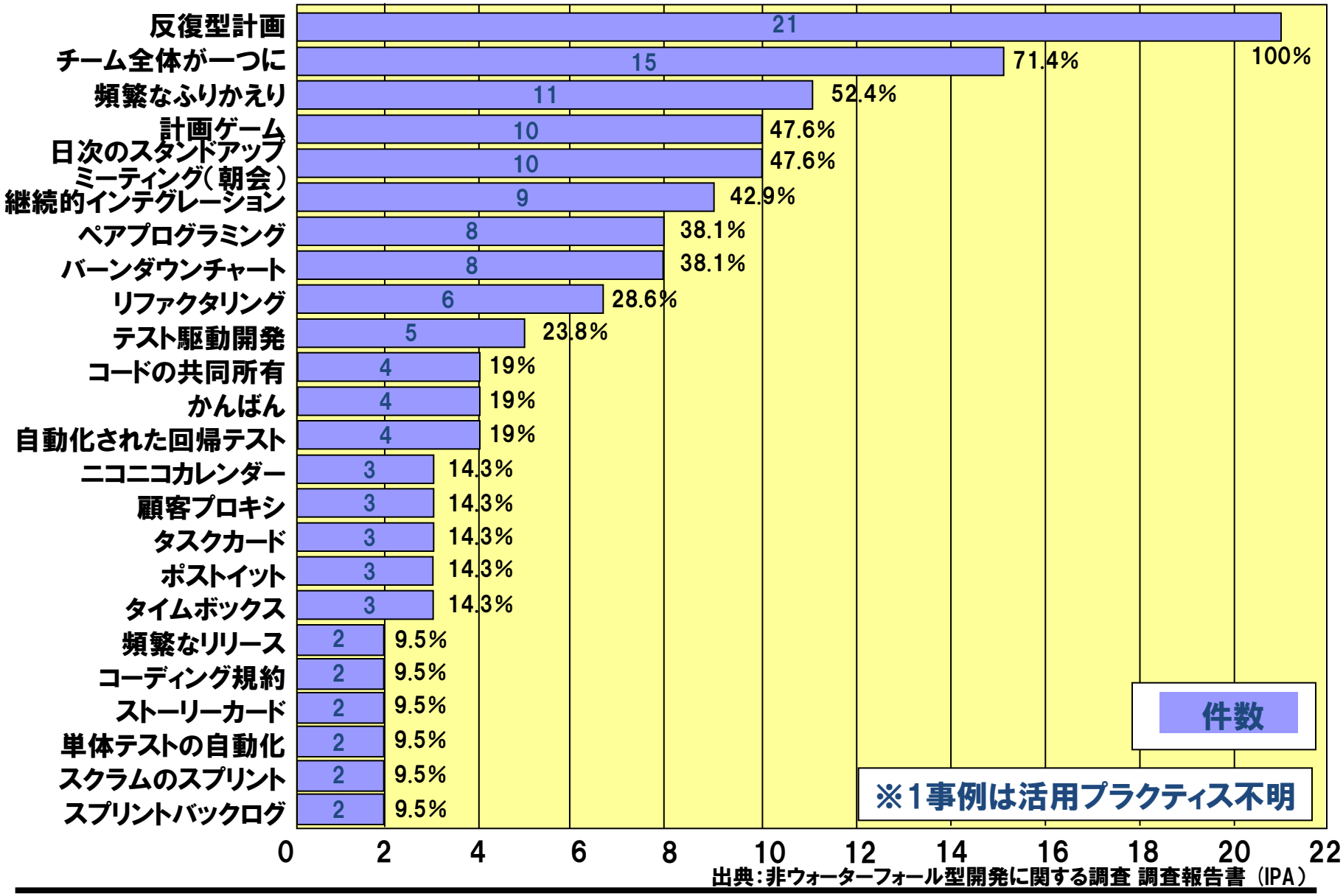
SEC
Software Engineering
for Mo･No･Zu･Ku･Ri

M社	1事例	サービス事業者

- ソーシャルゲーム開発の事例
- 5～6人のチームが数十チームある
- それぞれのチームの裁量に任せている面が大きいが、Scrumから逸脱しないようにアジャイルコーチが全体を見ている

【別紙4】2009年度の調査結果

SEC
Software Engineering
for Mo·No·Zu·Ku·Ri



出典:非ウォーターフォール型開発に関する調査 調査報告書 (IPA)