IPA Better Life with IT

進化するアジャイル，IoT時代のビジネスを担う

～ 四国で活躍する技術者への期待を込めて ～

Agile Japan 2016 高松サテライト
2016年6月25日

独立行政法人情報処理推進機構 (IPA)
技術本部 ソフトウェア高信頼化センター (SEC)
山　下　博　之

http://www.ipa.go.jp/sec/index.html

本日の講演内容

SEC
Software Reliability Enhancement Center

1. アジャイル開発プラクティス活用リファレンスガイド
2. モチベーションとアジャイル人材
3. アジャイル開発の今後：IoTの時代に向けて

付録

アジャイル開発手法について、IPA/SECにおける取組みを概観した後、エンタープライズアジャイルへとその適用か広がってきた状況を踏まえ、IoT時代のビジネスのために果たす重要な役割への期待を述べます。

本日の講演内容

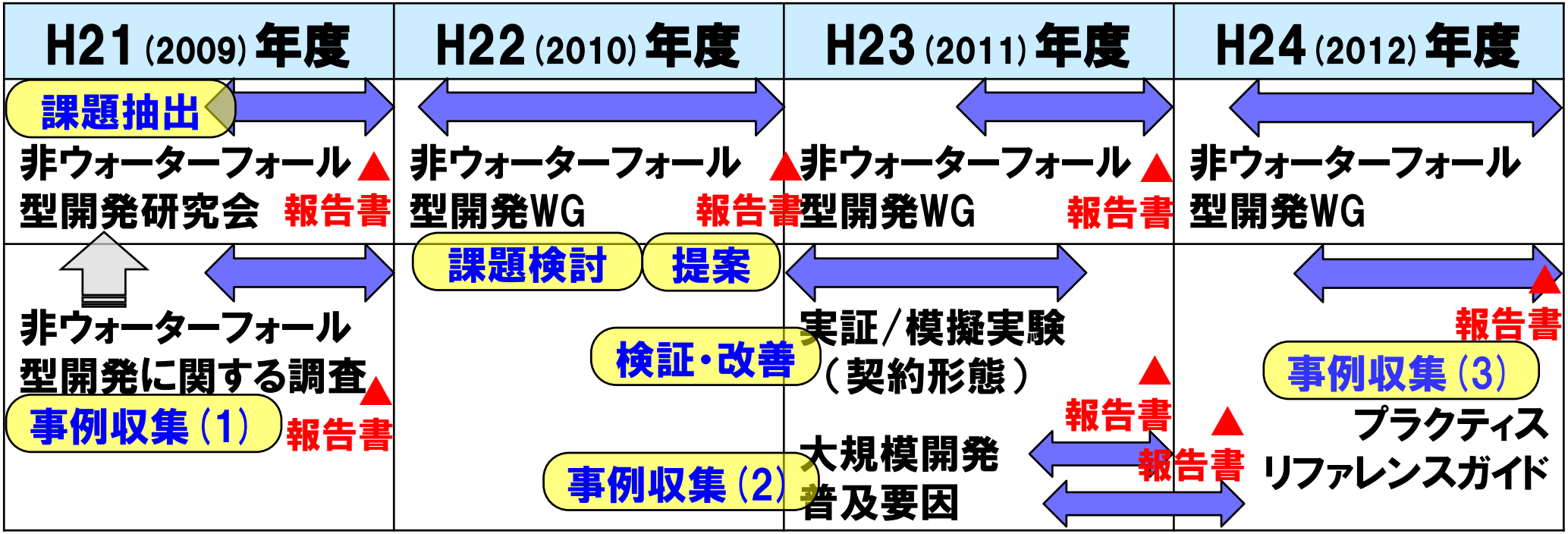
SEC
Software Reliability Enhancement Center

1. アジャイル開発プラクティス活用リファレンスガイド
2. モチベーションとアジャイル人材
3. アジャイル開発の今後：IoTの時代に向けて

付録

ふり返り：アジャイル開発に関するIPA/SECの取組み

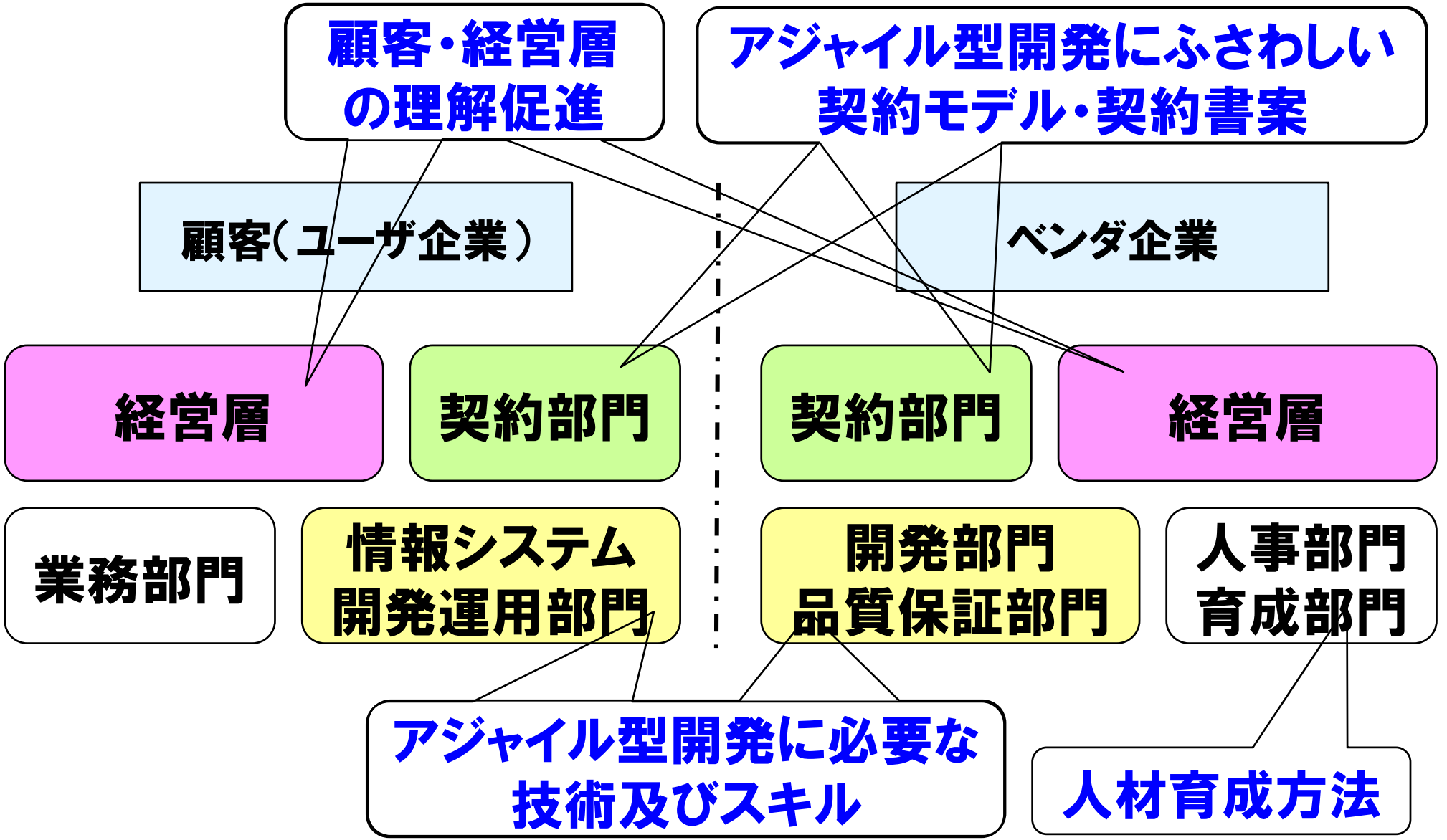
SEC Software Reliability Enhancement Center



報告書（公開中）

H21年度版	***http://www.ipa.go.jp/sec/softwareengineering/reports/20100330a.html***
H22年度版	***http://www.ipa.go.jp/sec/softwareengineering/reports/20110407.html***
H23年度版	***http://www.ipa.go.jp/sec/softwareengineering/reports/20120326.html***
（大規模開発）	***http://www.ipa.go.jp/about/press/20120328.html***
（普及要因）	***http://www.ipa.go.jp/sec/softwareengineering/reports/20120611.html***
（プラクティス）	***http://www.ipa.go.jp/about/press/20130319.html***

IPA/SECの取組み:ステークホルダと検討項目



※プラクティス：アジャイル開発を実践する活動項目

http://www.ipa.go.jp/about/press/20130319.html
http://www.ipa.go.jp/sec/softwareengineering/reports/20130319.html

ガイドの特徴

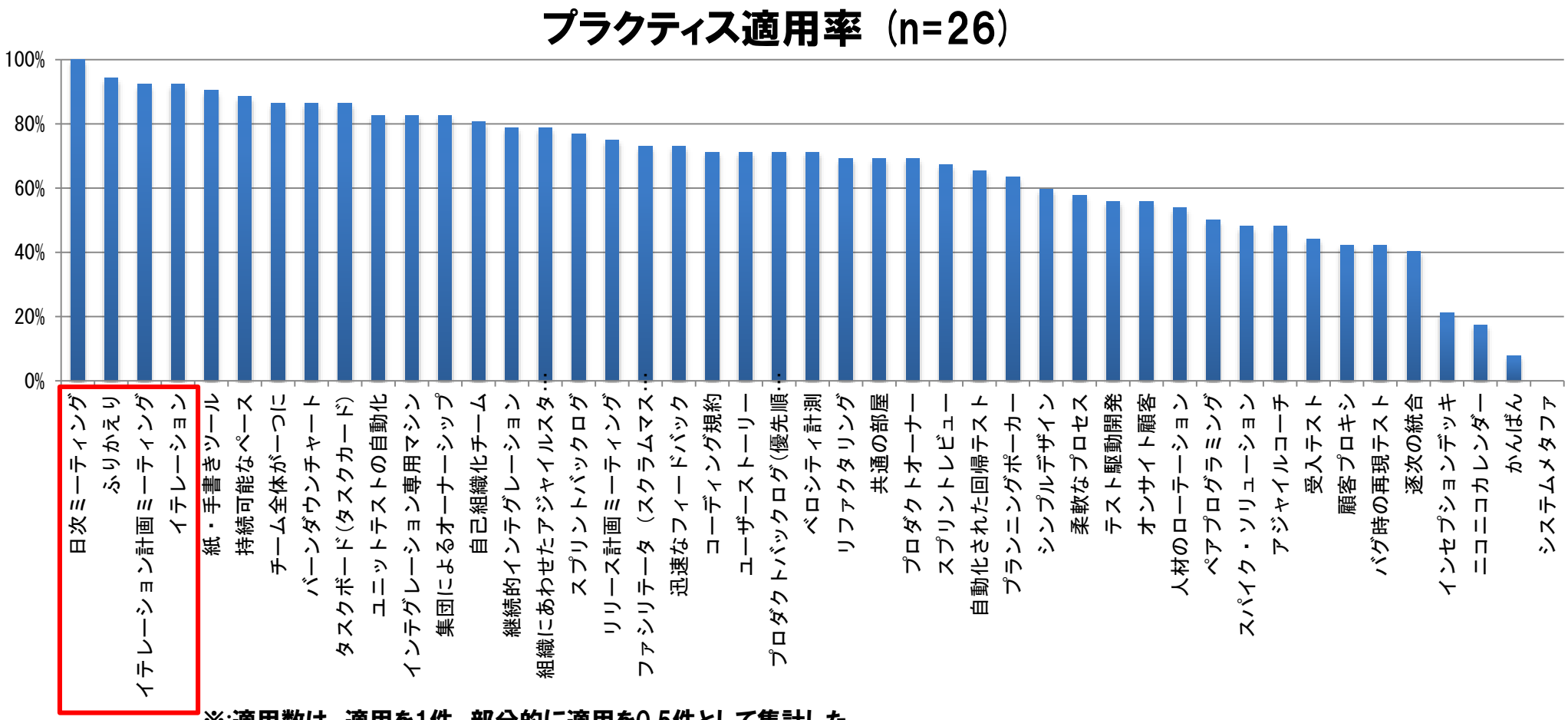
アジャイル開発を実践する活動項目

- 55個*のプラクティス，26個の事例，9つの活用ポイント　計224ページ
- 日本国内の開発現場からのヒアリングにより収集した知見を，パターン記述形式で取りまとめ
- MS-Wordファイルを公開し，クリエイティブ・コモンズ・ライセンスの下に，改変自由・営利目的利用可で使用許諾

＊ 類似のものを統合し，最終的には45個

適用プラクティス（全体）

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。



※: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。

プラクティス例概要 - 日次ミーティング

SEC Software Reliability Enhancement Center

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）：1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）。

プラクティス例概要 - ふりかえり

SEC Software Reliability Enhancement Center

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。

※1 メンバー全員で、Keep（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。

利用例

- **事例（25）**：当初はKPT[※1]を用いてふりかえりを行っていたが、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くなってしまう。
- 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。

※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。

プラクティス例概要 - イテレーション計画ミーティング

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間で合意している。

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。

事例概要 <<中大規模適用プロジェクトの事例>> 事例(4) C社

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**: チーム毎に実施した場合には、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により、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。

活用のポイント (1)

SEC Software Reliability Enhancement Center

(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ことが予想され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おり、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 Reliability Enhancement Center

(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否かの検証のために、「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 Reliability Enhancement Center

(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しなければならない。

(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。

本日の講演内容

SEC
Software Reliability Enhancement Center

1. アジャイル開発プラクティス活用リファレンスガイド

2. モチベーションとアジャイル人材

3. アジャイル開発の今後：IoTの時代に向けて

付録

ウォーターフォール型とアジャイル型との手法の違い

SEC Software Reliability Enhancement Center

ウォーターフォール型

（開発が）

失敗しないための手法

「プロセス」重視

“計画”駆動型

文化が異なる

ケースバイケースで使い分け

アジャイル開発

（ビジネスが）

成功するための手法

「人」重視

（顧客）“価値”駆動型

作るものも使用する技術も明確

例）ビルや橋の建設

最初から綿密な計画を立て
計画に従って着実に進める.

計画時には，ビジネス上，システム上の課題が未解決，開始後も変更の可能性大

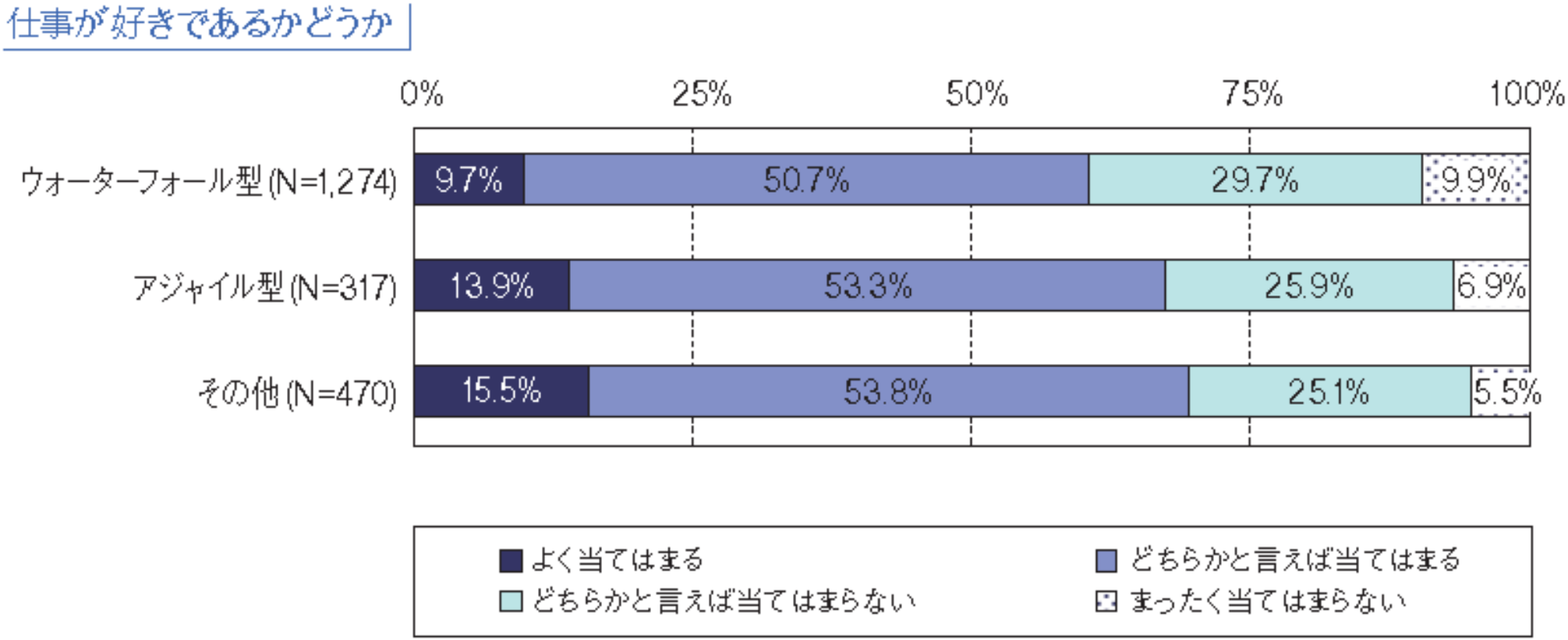
少し試して，その結果に基づいて
次のステップを進める.

多くの組織，チーム，個人にとって，アジャイル開発プロセスへの転換は“挑戦的”である.
それは，ある種の文化的変革を必要とするからだ.［Agile transformation, IBM］
http://www.ibm.com/smarterplanet/us/en/business_analytics/article/agiledevelopment.html

アジャイルは，プロセスではなく文化である.
Michael Sahota: “An Agile Adoption and Transformation Survival Guide: Working with Organizational Culture,” 2012.

参考データ

アジャイル開発技術者の仕事に対する感じ方・考え方 (1)



アジャイル開発技術者は，WF開発技術者に比べ，仕事が好きな割合が高い

出典:「IT人材白書2014」, IPA, 2014年4月25日.
http://www.ipa.go.jp/jinzai/jigyou/about.html

参考データ

アジャイル開発技術者の仕事に対する感じ方・考え方 (2)

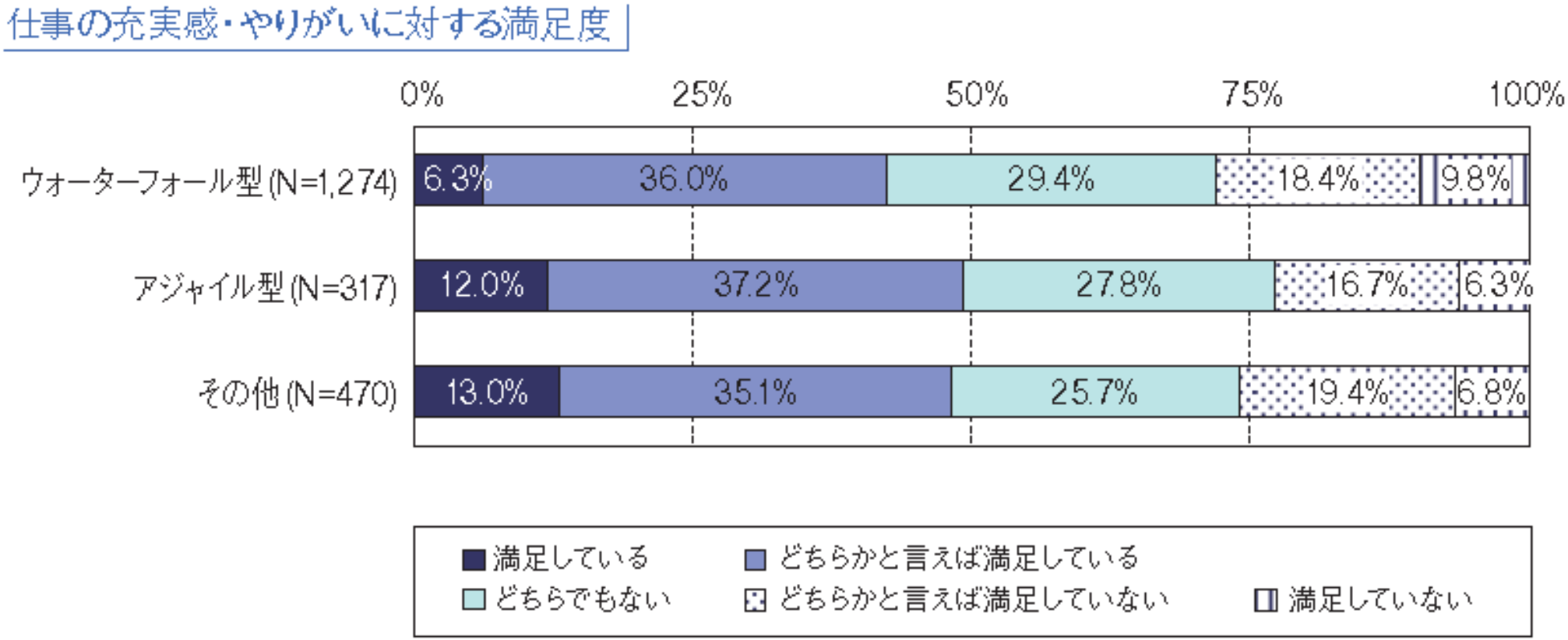


図２－３－19　仕事に対する意識【使用する開発手法別】

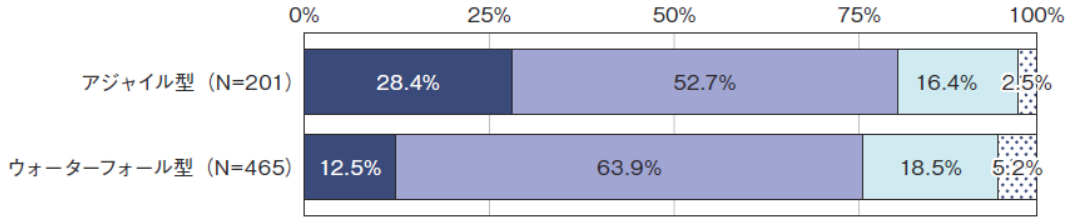
アジャイル開発技術者は，WF開発技術者に比べ，仕事に満足している割合が高い

出典:「IT人材白書2014」, IPA, 2014年4月25日.
http://www.ipa.go.jp/jinzai/jigyou/about.html

参考データ

アジャイル開発技術者の仕事に対する感じ方・考え方（3）

今の仕事に一生懸命取り組んでいる

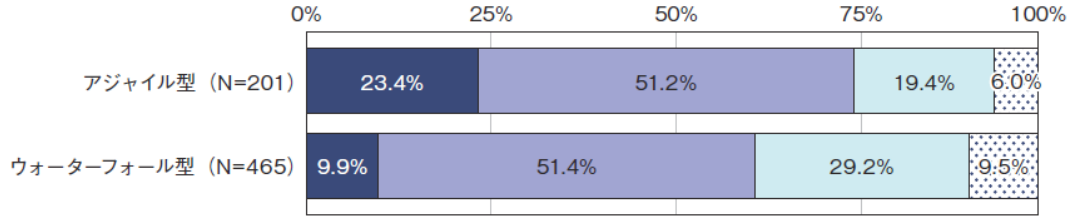


Q. 今の仕事に一生懸命取り組んでいる

アジャイル型　：28.4%

ウォーターフォール型：12.5%

仕事が好きである

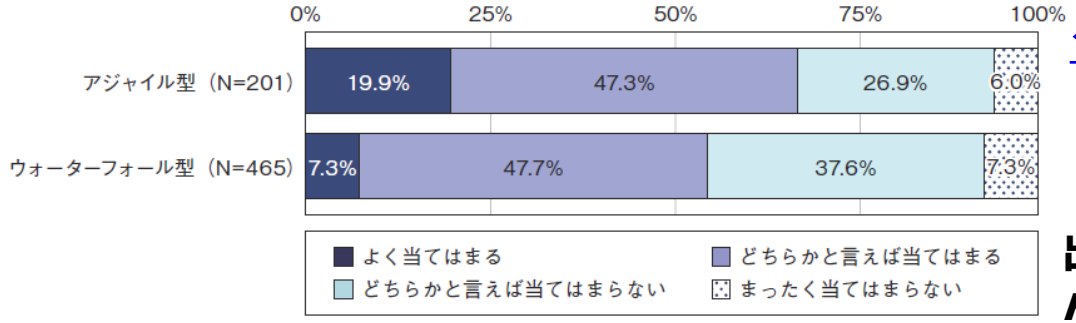


Q. 仕事が好きである

アジャイル型　：23.4%

ウォーターフォール型：　9.9%

この仕事をしていることを誇りに思う



Q. この仕事をしていることを誇りに思う

アジャイル型　：19.9%

ウォーターフォール型：　7.3%

<回答=「よく当てはまる」の割合>

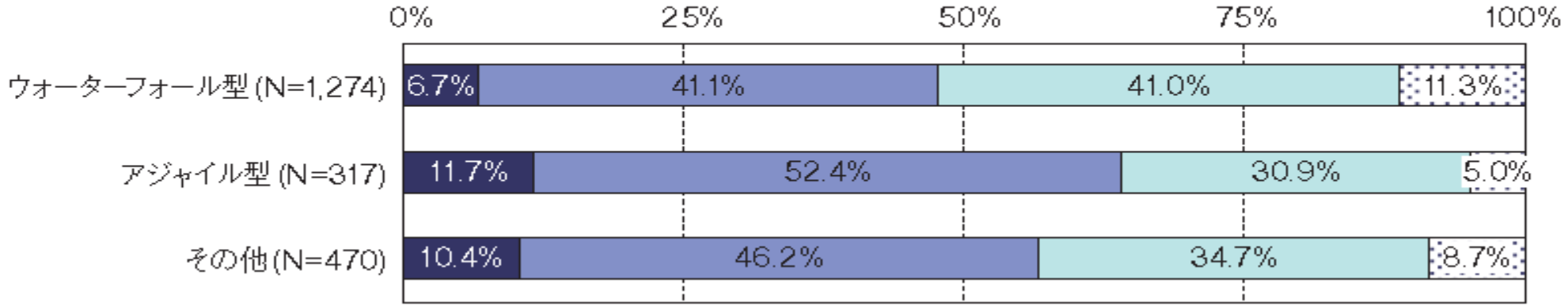
出典：「IT人材白書2013」, IPA, 2013/3/28.
http://www.ipa.go.jp/jinzai/jigyou/about.html

参考データ

スキルを学ぶ(1/2)

SEC Software Reliability Enhancement Center

新しい技術やスキルの習得のための勉強に自主的に取り組んでいるかどうか



次にどんな技術やスキルを学ぶべきかよくわかっているかどうか

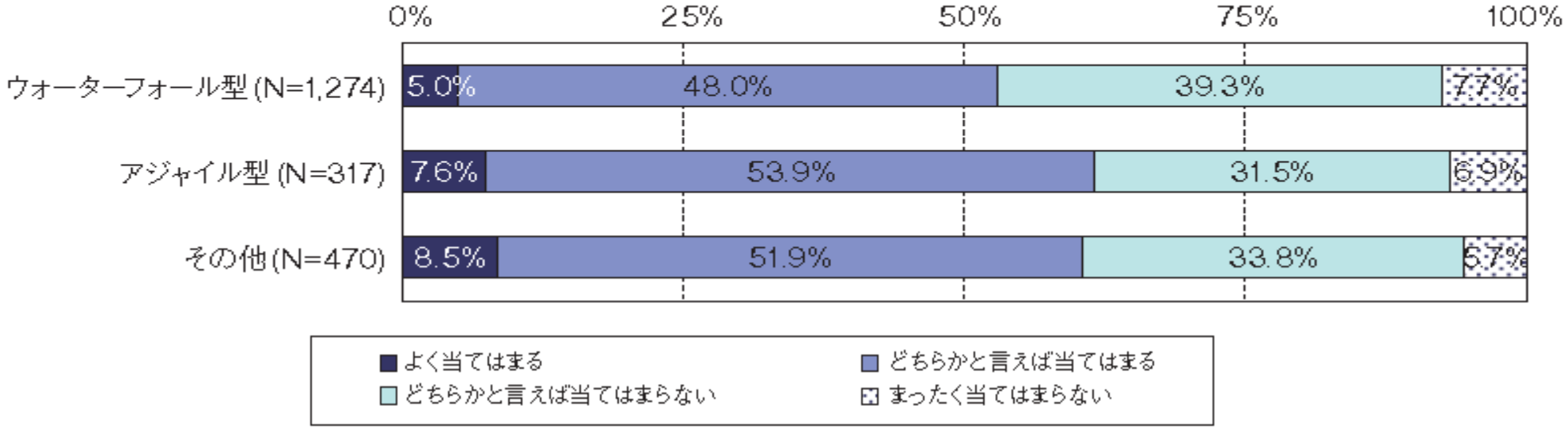


図2－3－20　勉強に対する取り組み姿勢【使用する開発手法別】

出典:「IT人材白書2014」, IPA, 2014年4月25日.

参考データ

スキルを学ぶ(2/2)

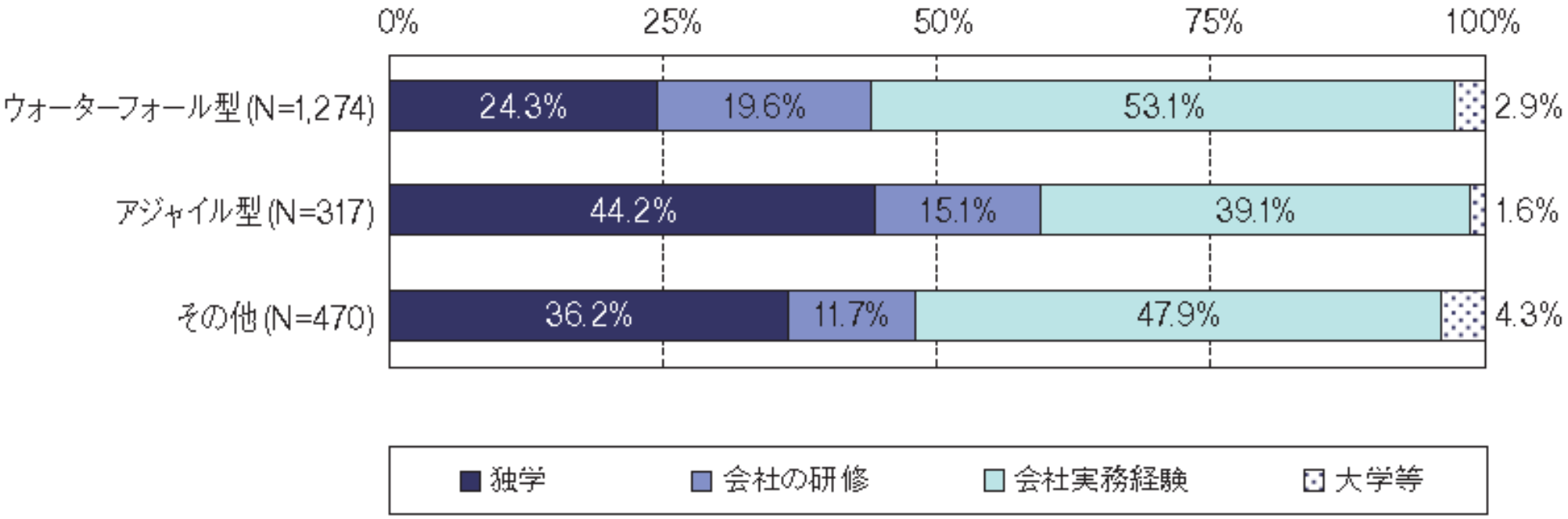


図2－3－21　開発手法を学習した場所【使用する開発手法別】

アジャイル開発技術者は，自主・独学で手法を学んでいる

出典:「IT人材白書2014」, IPA, 2014年4月25日.
http://www.ipa.go.jp/jinzai/jigyou/about.html

モチベーション…科学的実証の結果

SEC Software Reliability Enhancement Center

報酬のインセンティブは，視野を狭め，心を集中させることから，単純な仕事では効果があるが，そうでない創造的な仕事では逆効果．

成果を高めるのは，内的な動機付けに基づくアプローチ．

すなわち，重要だからやる，好きだからやる，面白いからやる，何か重要なことの一部を担っているからやる，というもの．

仕事において重要な要素は次の3つ：

- 自主性…自分の人生の方向は自分で決めたい
- 成長　…何か大切なことについて上達したい
- 目的　…私たち自身よりも大きな何かのためにやりたい

（ある程度の）裁量

スキルアップになる

顧客の”価値”を高める

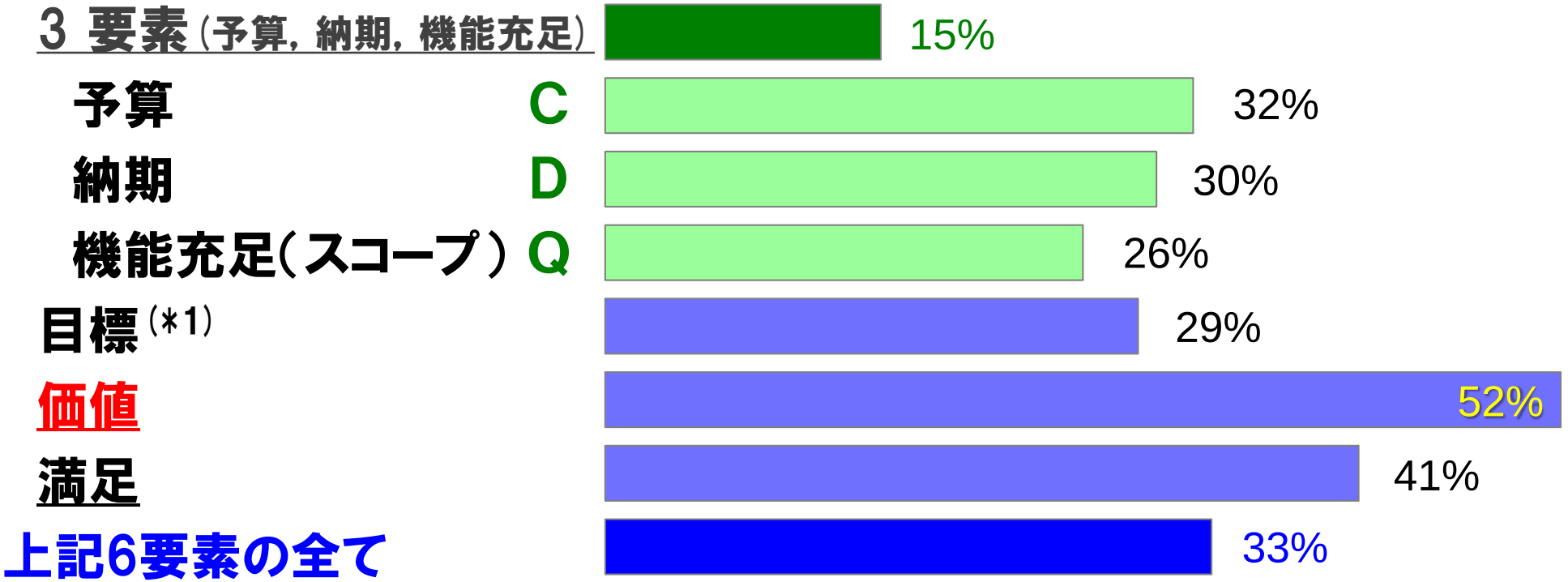
<出典> Dan Pink on the surprising science of motivation（ダニエル・ピンク「やる気に関する驚きの科学」）
http://www.ted.com/talks/lang/ja/dan_pink_on_motivation.html

上記要素は開発手法とは独立であるが，アプローチのし易さに特徴が反映される．

参考データ

ITプロジェクト成功の定義（あるアンケートの例）

<出典> CHAOS MANIFESTO 2014



(*1) 組織の戦略目標にどれだけ合致しているか?

6要素のうち，4つまで選択可．(回答数=309)

Triple constraints	15%
On budget	32%
On time	30%
On Target (scope)	26%
On goal	29%
Valuable	52%
Satisfied	41%
All of the above	33%

ＩＴプロジェクトの成功は，もはやQCDではなく，（顧客側の）価値や満足で決まる

★アジャイル開発に取り組むためにマインドを変える

SEC Software Reliability Enhancement Center

私見

メンバ間のスキル・能力にばらつきの多いソフトウェア開発チームにおいて，生産性を高め，所定の品質を確保するために，自由度と柔軟性を低くする方向，すなわち，プロセスの標準化や自動化が進められる．

このような創造性を発揮する機会が少ない状況において，「このままでよいのか？」という疑問を抱いたメンバは，自ら勉強する中で，顧客の「価値」を重視するアジャイル開発に取り組んできた．

標準化・自動化されたプロセスの下でのソフトウェア開発組織で“居心地の良さ”を感ずるメンバに対し，アジャイル開発の技術のみを訓練するとするならば，果たしてうまくいくのだろうか？

クリエィティブなソフトウェア技術者を目指す

SEC
Software Reliability Enhancement Center

ロバート・コール（日本の製造業の研究者）：
「日本のソフトウェア業はブルーカラーを作ってきた」よって，
「日本の製造業はソフトウェアでイノベーションは起こせない」
∵ソフトウェア技術者が知識労働者でなく，言われたことしかやらないブルーカラーだから．

<日本の現状>
コスト抑制のためにオフショアに依存し過ぎ，クリエイティブなエンジニアが育っていない．クリエイティブでないから守備範囲が狭い．
ホンハイ（鴻海精密工業）がシャープのエンジニア3千人のリストラを発表したのは，ホンハイから見ると，シャープのエンジニアの守備範囲が狭くて生産性が低いから．

<出典> つくだひとし:【講演採録】派生開発推進協議会代表・清水吉男氏 XDDPから「IoT」に挑む，IT記者会レポート，2016.6.14.

本日の講演内容

1. アジャイル開発プラクティス活用リファレンスガイド

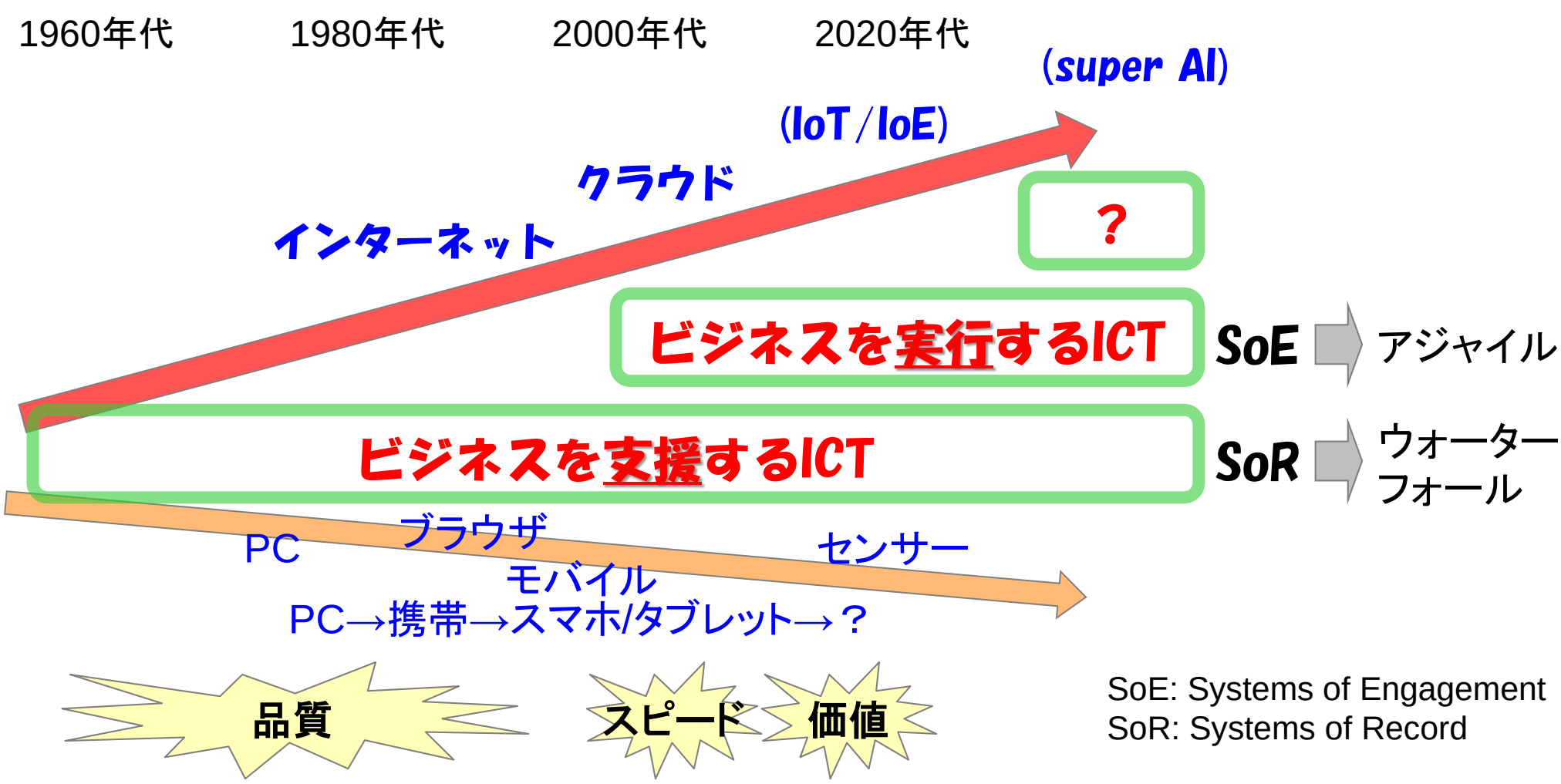
2. モチベーションとアジャイル人材

3. アジャイル開発の今後：IoTの時代に向けて

付録

http://i.creativecommons.org/l/by-nc-sa/3.0/es/88x31.png

ICTシステムの変遷と開発スタイル



変化に対応した迅速なシステム開発が求められる時代

SEC Software Reliability Enhancement Center

グローバル化に伴う変化
政治情勢、新興国の台頭、
政府政策の変更

マーケットの変化
顧客嗜好の変化、ニーズの多様化、
競合他社の動向、SNSの普及

技術の変化
HWの価格性能比の劇的改善、
IOTの進展、ビッグデータ、
クラウド、AI

経済情勢の変化
為替の変動、金利の変動、
景気動向

環境問題に関する変化
気候変動、資源枯渇、省エネ

自社の変化
経営方針、人的リソース、業績

さらに、IoT時代になると

様々なモノがつながる

異なる分野の
サービスがつながる

データを集めたり
モノを制御できる

サービス提供企業やユーザが
自由にモノをつなげられる

**製品やシステムの要件が
開発開始時には不確定**

アジャイル開発は必然

IoT時代のビジネス

SEC
Software Reliability Enhancement Center

IoT（Internet of Things）時代

モノ，ヒトとネットワークとソフトウェアを組み合わせた強みを活用し，さらにAIやビッグデータ分析等の技術を組み合わせた

商品／サービス競争の時代

- インターネット上で時々刻々と新デバイスが接続，新サービスが提供

  →現状や未来の全体像を見通すことはできない
- 顧客のニーズもそれらに誘発されて移ろいゆき，それがきっかけとなってさらにデバイスやサービスが更新

  →相互に深く影響し合いながら進展する
- ある日突然，新たなセキュリティ上の脅威が発生

  →対応の機敏さが死命を制する

進化し続けるアジャイル

IoT時代における環境の変化に機敏に応じることが可能な
商品／サービスを市場に投入

エンタープライズ・アジャイル

→企業活動そのもののアジリティ

→今や企業活動の根幹を担うICTシステムの
迅速でかつ確実な開発・更新

アジャイル開発手法

- 適用範囲がますます拡大
- 手法自身も進化し続けている
  （モデルやハードウェアをも対象とする，等）

ご清聴，ありがとうございました

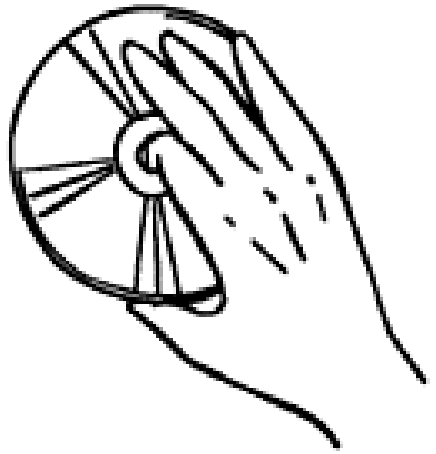
アジャイル開発に関するIPA/SECの検討結果等は：
http://www.ipa.go.jp/sec/softwareengineering/std/ent02-c.html

本日の講演内容

SEC
Software Reliability Enhancement Center

1. アジャイル開発プラクティス活用リファレンスガイド
2. モチベーションとアジャイル人材
3. アジャイル開発の今後：IoTの時代に向けて

付録



アジャイル開発プラクティス活用リファレンスガイド 事例一覧（1）

SEC Software Reliability Enhancement Center

調査先	No.	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#

アジャイル開発プラクティス活用リファレンスガイド 事例一覧 (2)

SEC Software Reliability Enhancement Center

調査先	No.	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

アジャイル開発プラクティス活用リファレンスガイド プラクティス一覧（1）

SEC Software Reliability Enhancement Center

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繰り返すこと
		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	状況や環境の変化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間の合意事項
		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にする
		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組みを行っている

アジャイル開発プラクティス活用リファレンスガイド プラクティス一覧（2）

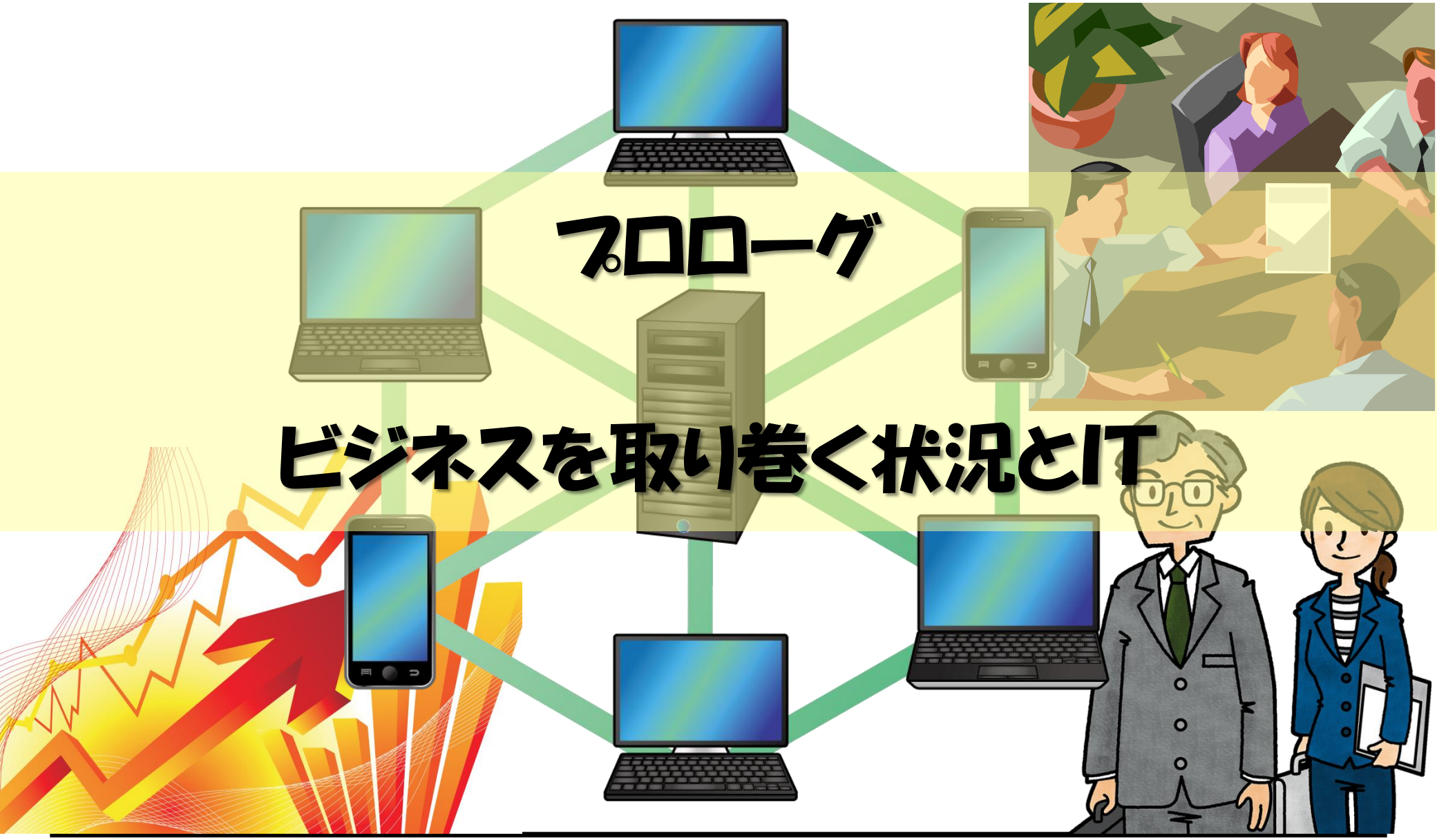
SEC Software Reliability Enhancement Center

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グラミングで開発している
		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、隠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	継続的インデグレーション、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持つ
		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

アジャイル開発プラクティス活用リファレンスガイド プラクティス一覧（3）

SEC Software Reliability Enhancement Center

カテゴリ	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ション
		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



プロローグ

ビジネスを取り巻く状況とIT

★ビジネスの3要素

5

ビジネスの3要素　ヒト
モノ
カネ
4要素　情報　*Business Intelligence BigData*
5要素　時間

変化対応の俊敏性（Agility）

参考データ

システム開発におけるQCDの優先順位

SEC Software Reliability Enhancement Center

システム企画工程におけるQCDの優先順位

- 品質：29 (28,29) %
- コスト: 23 (23,24) %
- **納期：48 (49,47) %**

調査で収集した1021 (918,801) プロジェクトのうち,「QCDのうちのどれかを優先した」という回答 (446 (377,313) プロジェクト) の内訳 [() 内は前年度,前々年度の結果]

<出典> ソフトウェアメトリックス調査2014 (2013,2012), 一般社団法人 日本情報システム・ユーザー協会 (JUAS).

そうした事業環境の中，いわゆる「QCD」のうち，特に納期を重視してものづくりを進めている．品質の確保は当たり前．開発・製造期間を短縮して製品の投入スピードをいかに速くできるかが，世界を相手に競争優位を築くカギになる．

<出典> CIOの哲学：三菱重工業 児玉敏雄氏，日経コンピュータ，2012.10.25.

開発(構築)手法の選択

SEC Software Reliability Enhancement Center

環境の変化に対する俊敏な開発(構築)が求められる場合

俊敏な開発(構築)手法

a. 非ウォーターフォール型開発(アジャイル開発) **少しずつ作って，確かめながら**

b. クラウドコンピューティング **作らないで，使う**

「超高速開発」

c. 自動コード生成／ビジネスルールマネジメントシステム(BRMS) **パラメータを変更するだけ**

1つのシステム全体を単一の手法で開発(構築)することが適切ではない(ケースもある)？

異なる手法で開発した部品の組合せ？

★重視事項は対象・ビジネス状況により異なる

SEC
Software Reliability Enhancement Center

ITシステムの対象や，
それを使うビジネスの状況に応じ，
重視する評価ポイントは異なる

“平均”で語ることは，
必ずしも適切ではない．

ITシステムのクラスと特徴

SEC
Software Reliability Enhancement Center

・比較的長期間，そのまま運用
・障害発生時の社会的影響大
・一般利用者の厳しい反応

高信頼 ← 重要インフラ等ITシステム

対象は拡大傾向

共通基盤系

短納期
（変化俊敏対応）
（高速開発）

ビジネス戦略ITシステム

サービス系

従来のアジャイル開発の主対象

業務支援ITシステム

・先を見通しにくい
・激しい環境変化
・競争優位の確保

各クラスに対応した

適切なアーキテクチャと，構築・運用体制及び手法

参考データ

システム構築時の重視事項

SEC Software Reliability Enhancement Center

システム構築時の重視事項(1,2位の合計%)

	基幹系	業務支援情報系	Web・フロント系	管理業務系
データ数	989	966	963	974
品質	76.8	59.2	59.3	76.9
コスト	41.2	54.8	53.1	50.2
開発スピード	14.3	35.9	43.5	12.3
変更容易性	27.7	33.1	32.4	23.6
継承性	34.9	14.5	7.3	33.0

品質，継承性重視

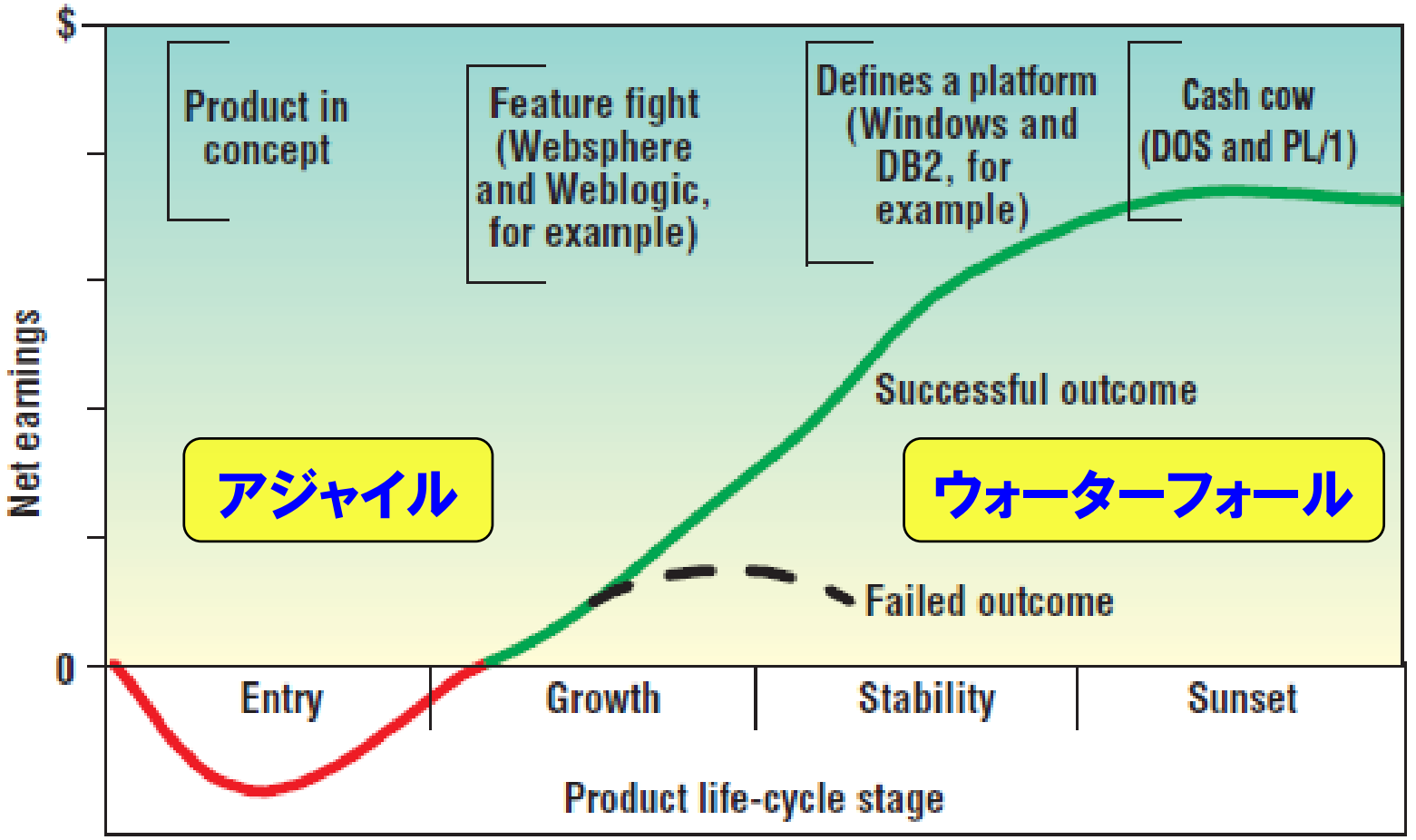
開発スピード，変更容易性重視

重要インフラ等ITシステム，Bsys(共通基盤系)

ビジネス戦略ITシステム(サービス系)

<出典> IT動向調査2014，図表8-3-1，一般社団法人 日本情報システム・ユーザー協会(JUAS)．

ビジネス・ステージと開発手法



A financial model of software product development.

ソフトウェア製品のライフサイクル・モデル例と開発手法

<出典> Ram Chillarege: The Marriage of Business Dynamics and Software Engineering, IEEE SOFTWARE, November/December 2002.

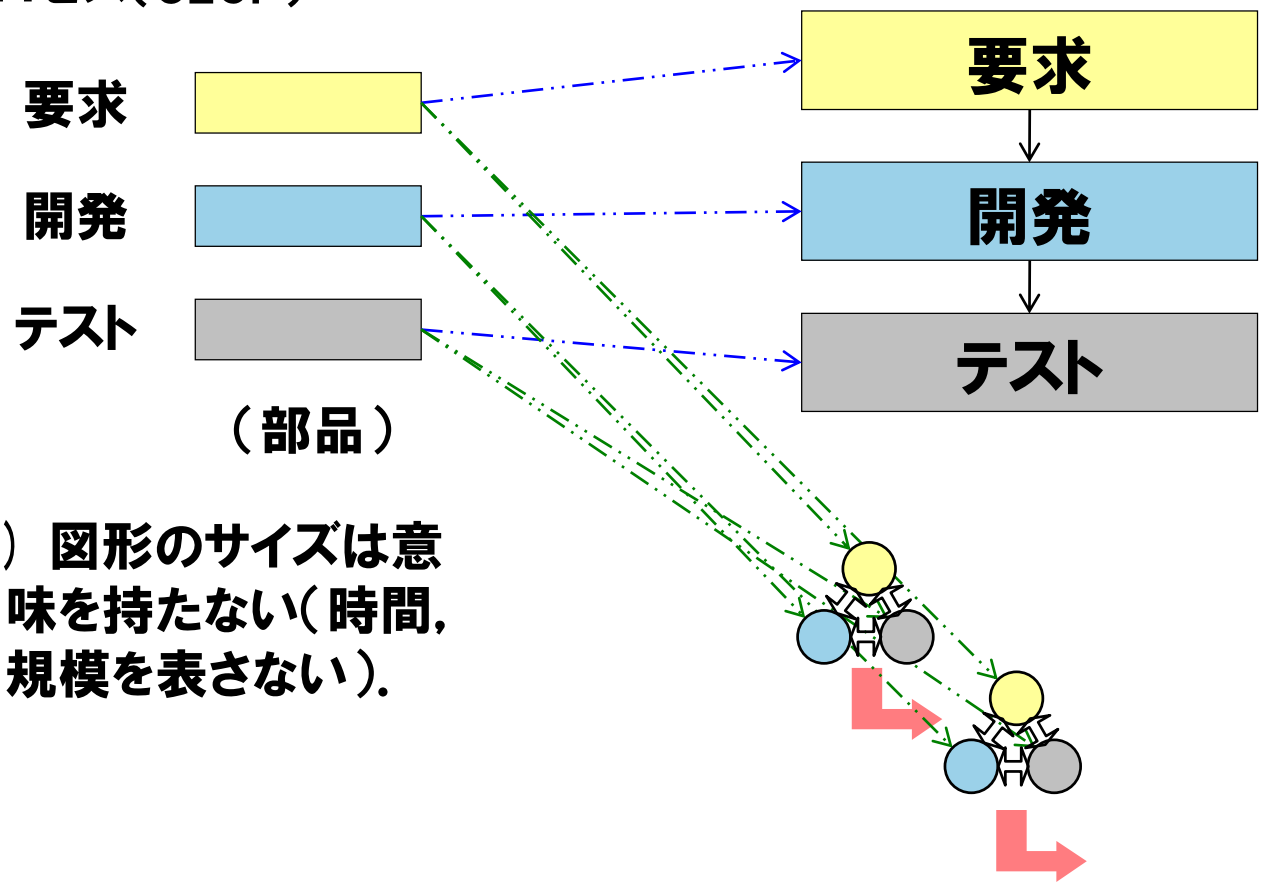
アジャイル開発の特徴

アジャイル開発のプロセスは，
大きく3つのモデルに分類される

アジャイル開発のモデル – プロセスの対応 –

<標準>
ソフトウェアライフサイクル
プロセス(SLCP)

<実際>



ウォーターフォール型

大きなプロセスを
順に実施し,
それを1回で終了

注) 図形のサイズは意味を持たない(時間,規模を表さない).

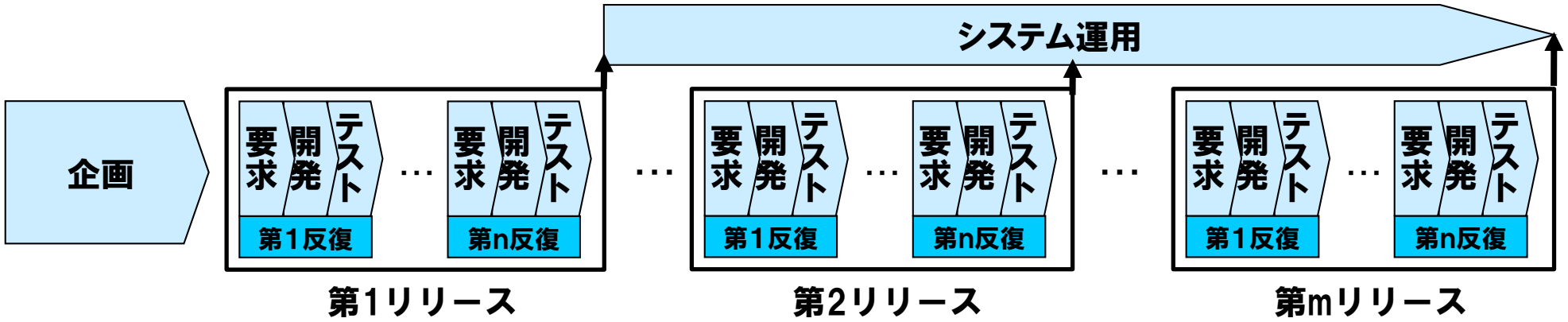
アジャイル型

小さなプロセスを
行き来しつつ実施し,
それを何回も反復

注) 図形のサイズは意味を持つ.

調査事例から導かれた開発プロセス・モデル（1）

モデル1



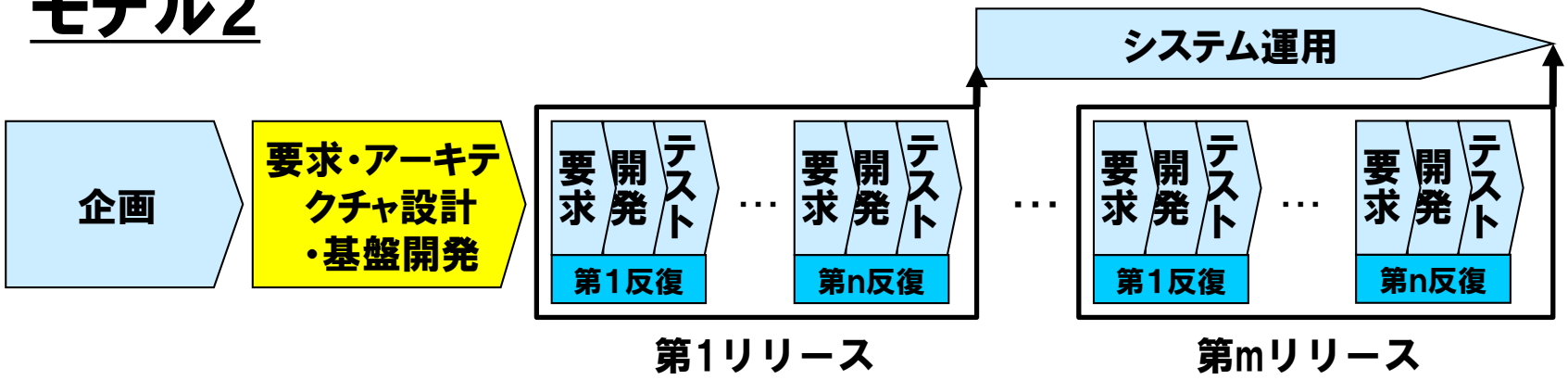
- n=1のケースもあり。

考え方
シンプルな基本形

調査事例から導かれた開発プロセス・モデル (2)

SEC Software Reliability Enhancement Center

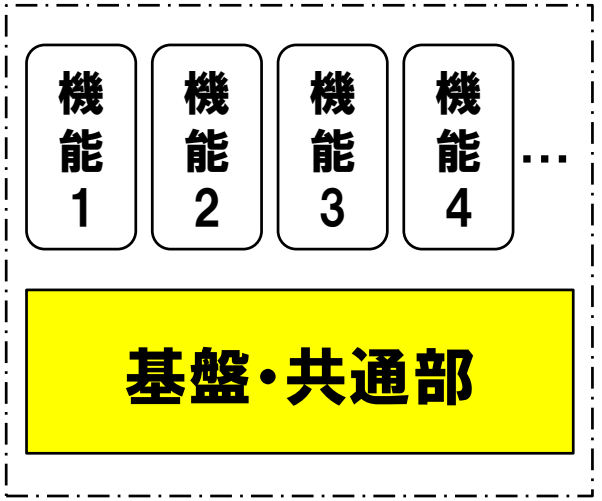
モデル2



- 比較的大規模システム／新規開発で全体のシステム構造が不明確なケースなど

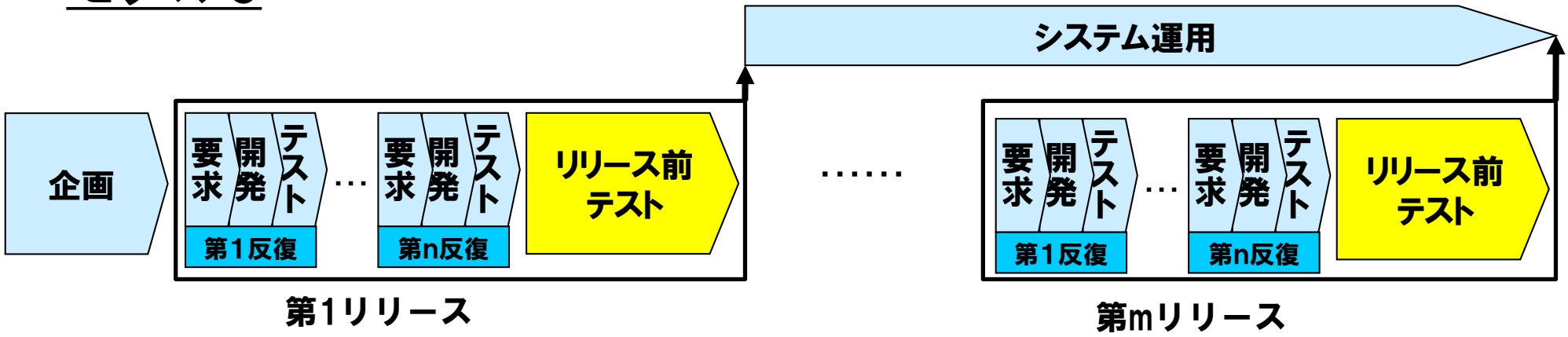
考え方

拡張形．基盤・共通部といくつかの機能部とから構成されるソフトウェア（右図）において，最初にまず，基盤・共通部の開発を終えた後，機能部群について，アジャイル開発を行う．基盤・共通部が確固としていないと，追加・変更時の機能部への影響が大きくなりすぎることを避ける．アジャイル開発では，基盤・共通部の変更は，原則として行わない．



調査事例から導かれた開発プロセス・モデル (3)

モデル3



- アジャイル開発では反復ごとにリリースできる品質までテストを行うことが原則だが、各リリース工程前に行う重点的なテストを実施することがある。
- リリースは複数回繰り返される

考え方

顧客やビジネスの特徴から，特に高い品質が求められたり，品質がクリティカルであったりする場合に，リリース前に品質確保のための特別のアクションを実施する．

開発プロジェクトのパラメータ間の関係

SEC
Software Reliability Enhancement Center

スコープ（要求）のサイズが品質に影響

固定

固定

固定

S:スコープ（R:要求）

C:コスト

D:納期

Q:品質

計画駆動

見積り→実際には変動

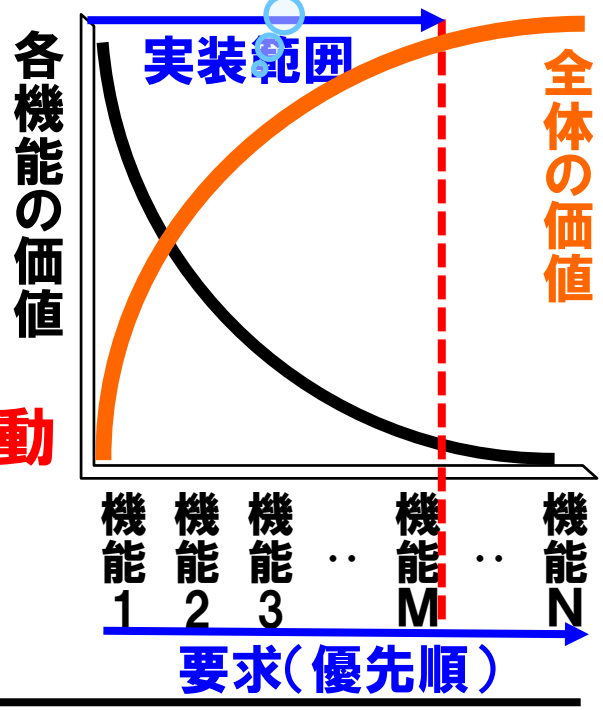
品質を維持しようとするとコストと納期に影響

価値駆動

優先順に従って変動

参考
QCDの優先順位

優先度の低い機能は実装しても結局は使われない
→無駄な実装はしない

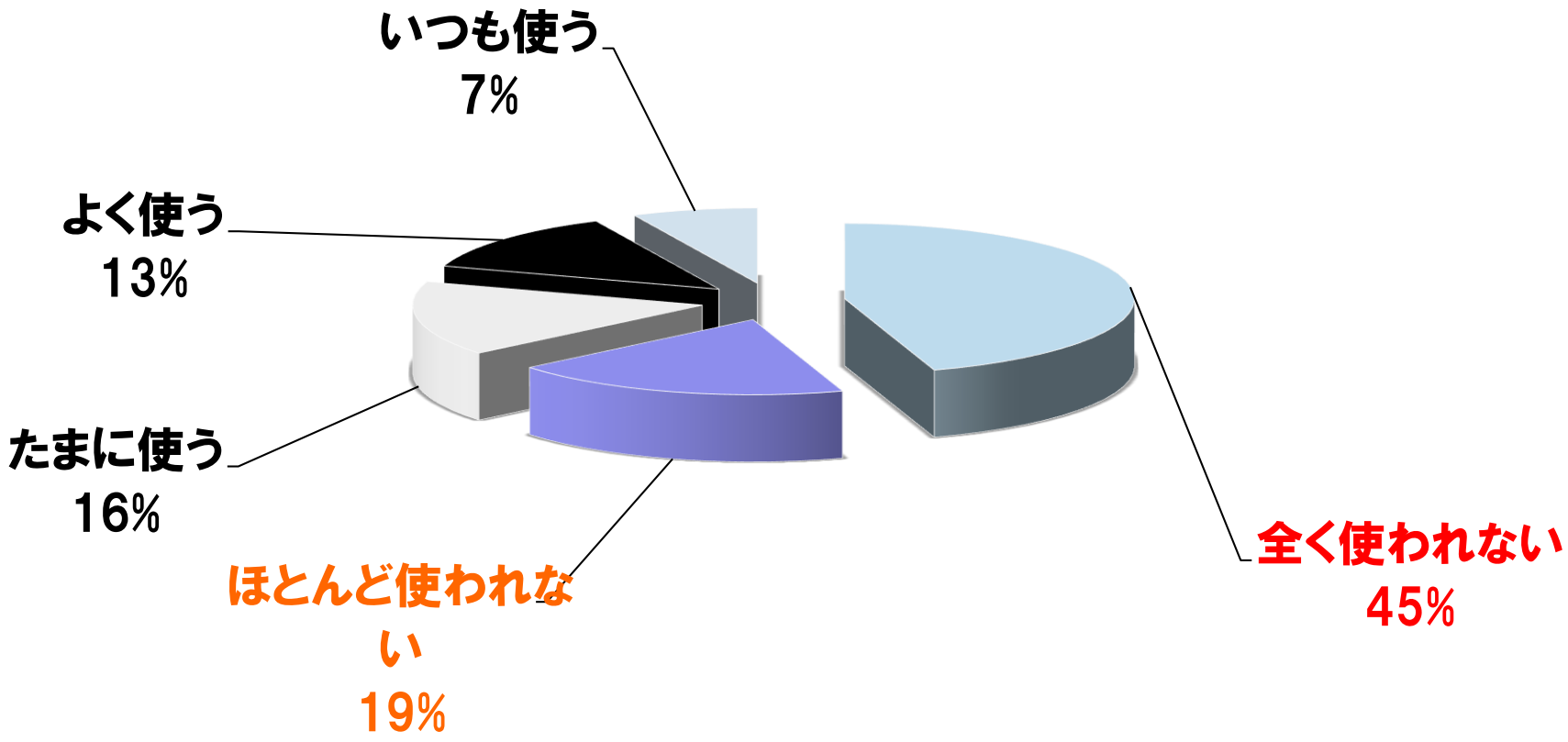


参考データ

システム機能の利用度（要求の劣化）

SEC Software Reliability Enhancement Center

システムの機能の利用度



<出典> *Standish group study report in 2000 chaos report*
（平鍋健児氏のプレゼン資料掲載）

顧客及び経営層の理解

ユーザ・ベンダ共に，
経営層の一層の理解が求められる

顧客・経営層は開発への一層の関与が必要

SEC Software Reliability Enhancement Center

顧客(ユーザ)経営層

ビジネス環境が激しく変化する現状において，ITシステムに関し，従来のように情報システム部門に任せきりでは適切に対応できない．開発形態(*)にも深く関与する必要がある．

(*) アジャイル開発の採用，クラウドコンピューティングの利用，など

＜経営層の責任＞
- 情報システムに関する理解の増進
- 迅速かつ適切な意思決定
- 関係部門との経営上の綿密な調整

開発手法の話に留まらない
企画も一選択肢に

ベンダ経営層

俊敏な開発の実績を武器に受注を狙う海外勢等に対抗するためには，自ら俊敏な開発を実施できる体制作りに取り組むと共に，その結果を顧客に売り込む必要がある．

ベンダ経営層は開発への一層の関与が必要

SEC Software Reliability Enhancement Center

顧客(ユーザ)経営層

ビジネス環境が激しく変化する現状において，ITシステムに関し，従来のように情報システム部門に任せきりでは適切に対応できない．開発形態(*)にも深く関与する必要がある．

(*) アジャイル開発の採用，クラウドコンピューティングの利用，など

＜経営層の責任＞
- 情報システムに関する理解の増進
- 迅速かつ適切な意思決定
- 関係部門との経営上の綿密な調整

アジャイル化は世界の主流

ベンダ経営層

俊敏な開発の実績を武器に受注を狙う海外勢等に対抗するためには，自ら俊敏な開発を実施できる体制作りに取り組むと共に，その結果を顧客に売り込む必要がある．

アジャイル開発人材

アジャイル開発に特徴的なスキル，姿勢がある

アジャイル開発の特徴は，技術者のモチベーションのドライブ要因とマッチしている

アジャイル開発人材の育成の考え方

SEC Software Reliability Enhancement Center

アジャイル開発を実践する活動項目

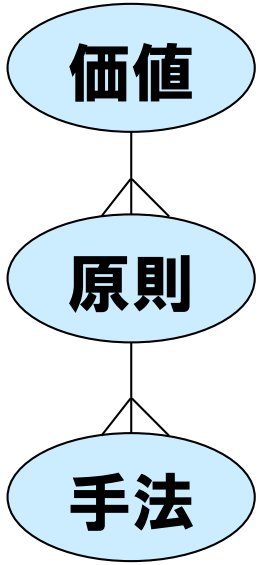
＜アジャイル開発の実際＞

- 一つのプロジェクトで全てのプラクティスを使う訳ではない
- 各プラクティスに厳格な規範はない
- 様々な方法論・数あるプラクティスから，プロジェクトや組織に適したものを取捨選択し，カスタマイズすることが必要

- （平時）一通りのプラクティスを理解する
- （プロジェクト参画時）使用するプラクティスの習得

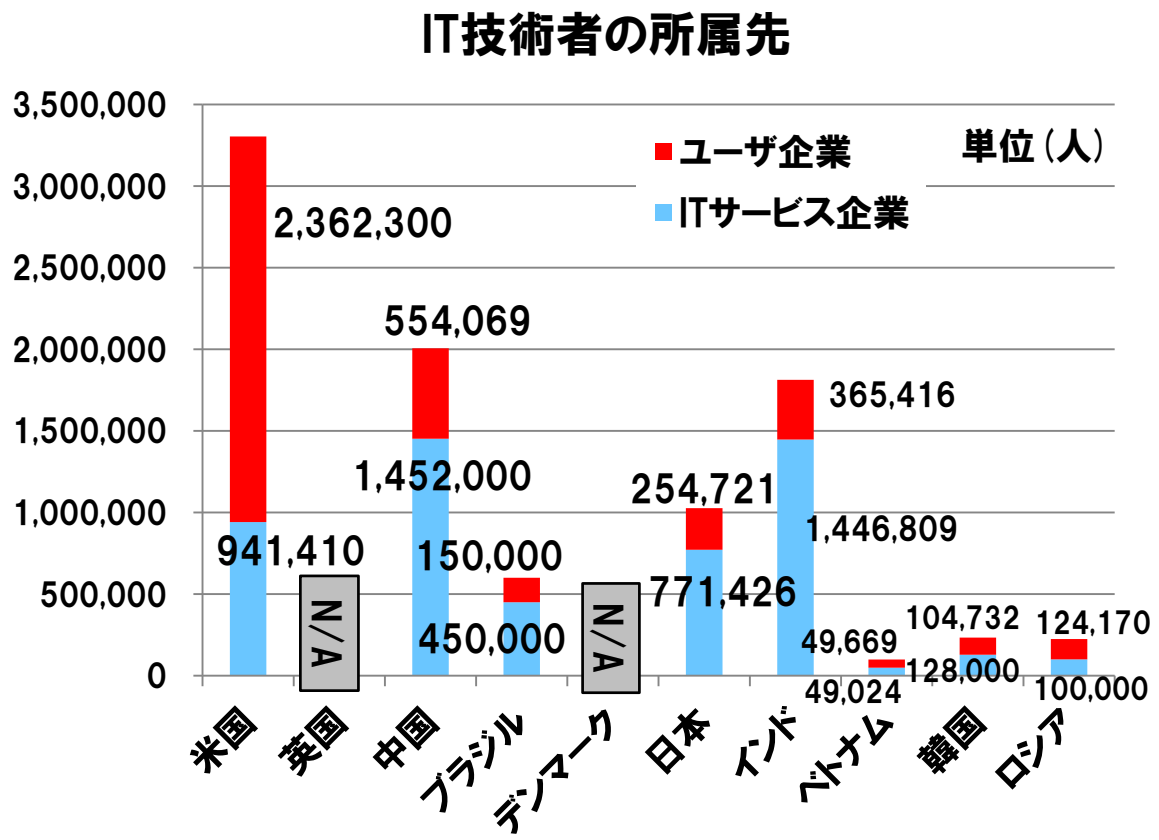
↓

全てを完全に身につけるより，価値に従って行動する習慣を確実に身につけることが重要



参考データ

IT技術者の所属先の国別比較



米国ではユーザ企業にIT技術者が所属する割合が高い
逆に，日本ではITサービス企業に所属する割合が高い

出典：「グローバル化を支えるIT人材確保・育成施策に関する調査」概要報告書，2011年 3月（IPA）

★アジャイル開発はそのままでは生産性が低い

SEC
Software Reliability Enhancement Center

アジャイル開発の生産性を高めるためには，
ツールの活用，特に，
テストの効率化（自動化等）が必須である

日本におけるソフトウェア開発一般の生産性は，諸外国に比べて低いと言われている．※
ALM（Application Lifecycle Management）の導入も検討すべきである．

※ 平成25年度　年次経済財政報告，内閣府
http://www5.cao.go.jp/j-j/wp/wp-je13/13.html

参考データ

各種開発手法の比較例

SEC Software Reliability Enhancement Center

3種開発法の比較(参考値)

		WF	アジャイル	xRAD	アジャイル/WF	xRAD/WF
総費用/JFS	平均	112.19	135.45	40.70	1.21	0.36
	係数	28.20	57.65	6.40		
工数/JFS	平均	1.28	2.15	0.48	1.68	0.37
	係数	0.44	1.60	0.26		
工期/JFS	平均	0.31	0.24	0.10	0.77	0.32
	係数	0.04	0.04	0.03		
データ数		337	51	43		

注) xRADは超高速開発手法のツールの一つ.
平均は各開発法のデータの平均値であり, 係数はJFSの1単位増加に伴う総費用, 工数, 工期の増加を示す. 係数が大きいと, システム開発規模の増加につれて各要因の値の増加がより大きくなることを示す.

アジャイルは, テストを繰り返す等の理由により, 工数大

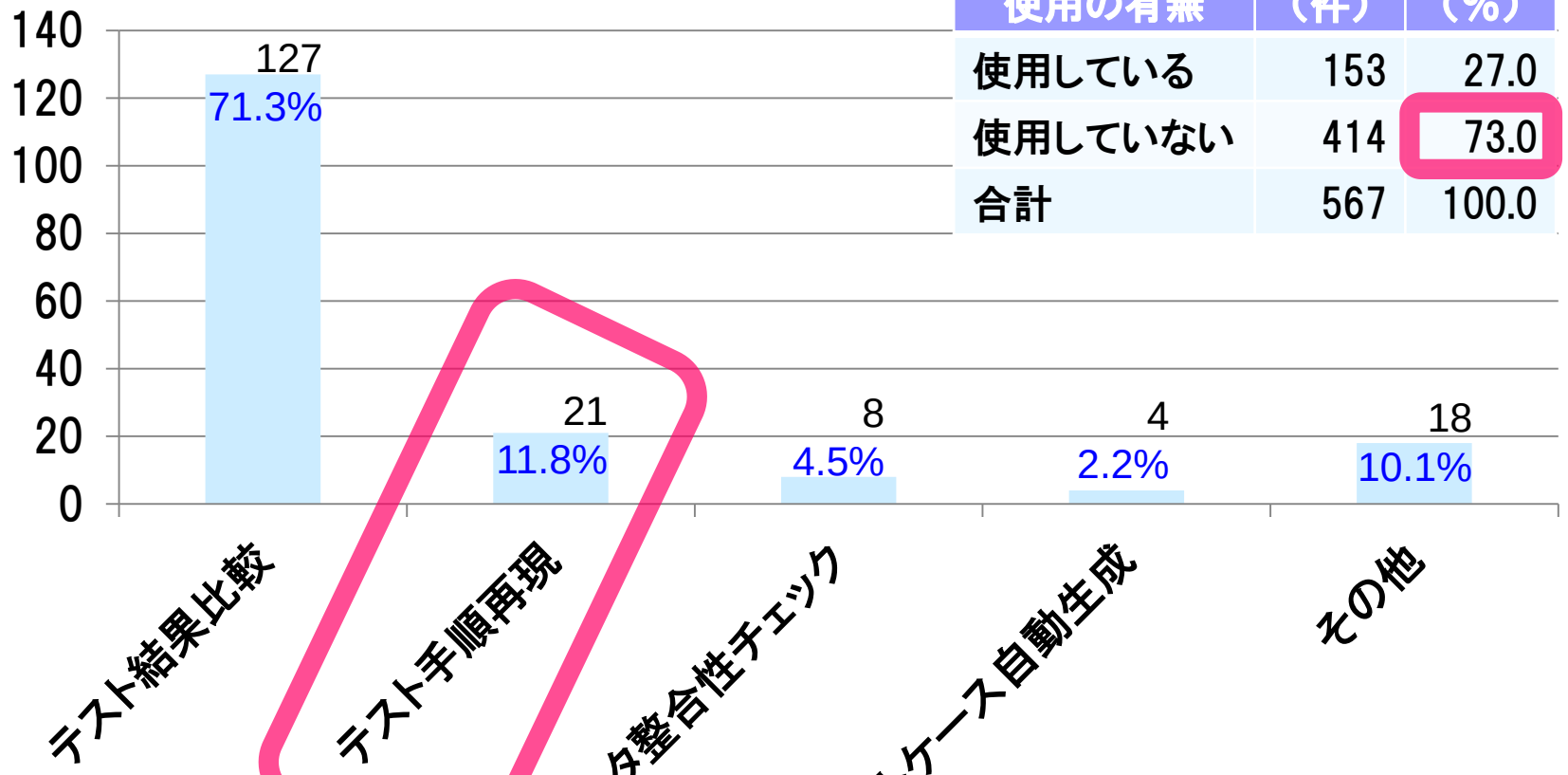
<出典> ソフトウェアメトリックス調査2014, 図表6-258,
一般社団法人 日本情報システム・ユーザー協会 (JUAS).

参考データ

適切なツールの活用（テストツールの活用状況）

SEC Software Reliability Enhancement Center

テストツールの使用の分布



テストツール使用の有無	件数（件）	割合（%）
使用している	153	27.0
使用していない	414	73.0
合計	567	100.0

テストツールの活用により，繰返しテストの効率化を

<出典> ソフトウェアメトリックス調査2014，図表7-60，7-61，
一般社団法人 日本情報システム・ユーザー協会（JUAS）.

★アジャイル開発の契約は運用で解決できる

私見

優先順位の問題.

アジャイル開発が真に必要であると判断し,

リスクをとる覚悟があれば,

契約の問題は,

運用を工夫することにより解決できるはず

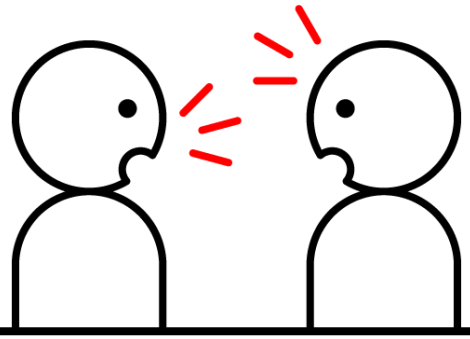
おまけ

SEC
Software Reliability Enhancement Center

ユーザとベンダとの会話のパターン

あなたの現状は，どちらに近いですか？

あなたは，どちらを好まれますか？



ユーザ/ベンダ間の会話例：パターン1(1)

今回の開発の予算は○○円です．

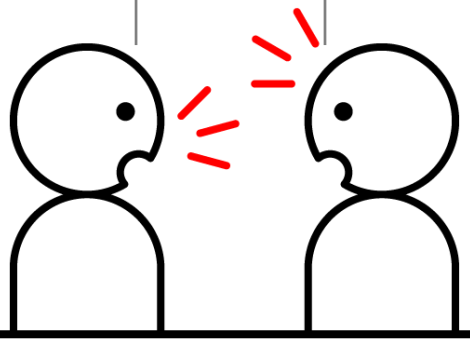
必要かどうか，今ははっきりしないけど，この機能Aは含めたい．
□□にはリリースしたい．

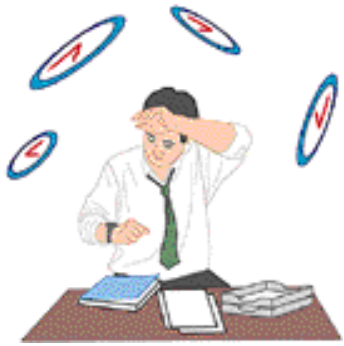
そこは何とかならないか？

ありがとう．それでよろしく頼む．

お客さんの要求を見積もると，コストが少しオーバーするし，納期も厳しい．
機能Aは難しい．

長い付き合いだから，プロジェクトの工夫で何とかしましょう．テスト項目を減らすことにしよう．





ユーザ/ベンダ間の会話例：パターン1(2)

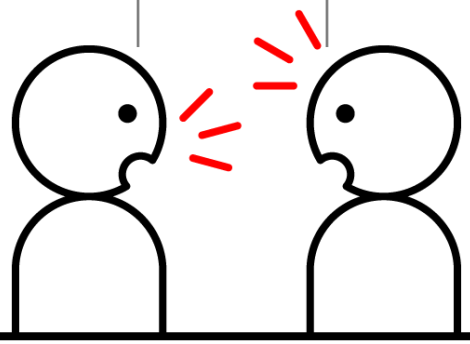
ライバル会社のサービスが明らかになった．対抗するために，機能Bを変更して欲しい．
また，機能Cを追加して欲しい．

今頃言われても困る．
コストも納期も守れない．

そこを何とかしてもらえないか？

仕方ないねぇ．
ドキュメントはリリース後に最小限でいいか？
リリース前のテスト項目もさらに絞れば，何とかなりそうだが．

助かる．それでよろしく頼む．



ユーザ/ベンダ間の会話例：パターン2(1)

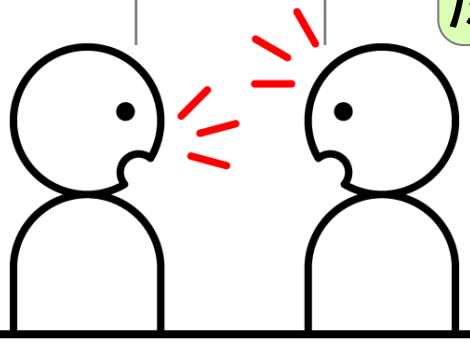
今回の開発の予算は○○円です．

ライバル会社より早く■■までにサービス提供したいので，基本機能ができたら，まず，初期リリースできないか？
その後順次，追加機能をリリースしていきたい．
この機能Aは，今は必要性が不明確なので優先順位は低い．

では，その条件でよろしく頼む．

長い付き合いだから敢えて言うが，基本機能だけでも■■までにリリースすることは難しい．
□□までなら何とかできる．

分かった．要求の内容が変わったら，早めに知らせて欲しい．



ユーザ/ベンダ間の会話例：パターン2(2)

ライバル会社のサービスが明らかになった．対抗するために，既にリリース済の機能Bを変更して欲しい．
また，機能Cを追加して欲しい．

それは構わない．必要であれば，予算の増額を調整する．

助かる．それでよろしく頼む．

分かった．その代わり，当初予算内では，機能Aと機能Dは盛り込めなくなる．

では，当初予定を変更し，4週間後の次回リリース時に，機能Bの変更と機能Cの追加分を含めることとする．