IPA Software Engineering Center
Information-technology Promotion Agency, Japan

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

～国内の中規模及び大規模開発プロジェクトへの適用事例調査～

目次

背景

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

アジャイルソフトウェア開発宣言 [※]

①プロセスやツールよりも，個人と対話を
②包括的なドキュメントよりも，動くソフトウェアを
③契約交渉よりも，顧客との協調を
④計画に従うことよりも，変化への対応を

→ 協調とコミュニケーションを重視して動くソフトウェアを成長させながら変化に対応する

アジャイル開発のスイートスポット

システム規模	0. . 12. . 300（開発メンバー数）
深刻度	シンプル、経済被害、. . . . 人身事故
システムの成熟度	新規開発、レガシー保守
要件の変化率	低、中、高
ビジネスモデル	自社開発、オープンソース、. . .
アーキテクチャ	安定、変化した、新しい
チームの分散	一か所、. . . 、オフショア、外部委託
統制	単純なルール、. . . . 、SOX、. . .

(Philippe Kruchten, 2009, “Is Agility a New Passing Fad?”)

→ スウィートスポットは大規模開発ではない

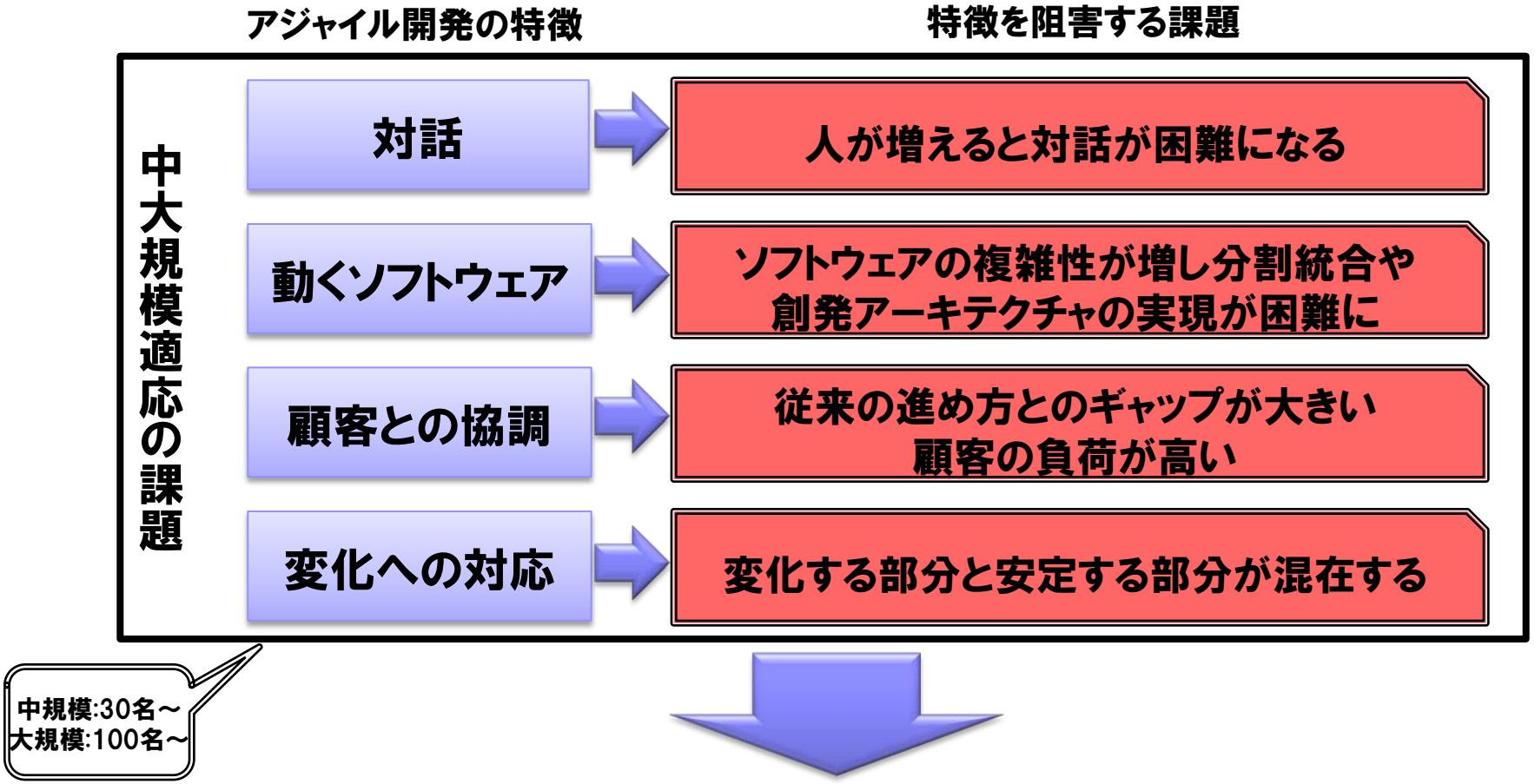
→ 海外では大規模事例も紹介されつつある

平成21年度事例調査
中大規模（30名～）の事例はなかった

→ 近年、国内での大規模事例が増えてきた。

※:http://agilemanifesto.org/iso/ja/

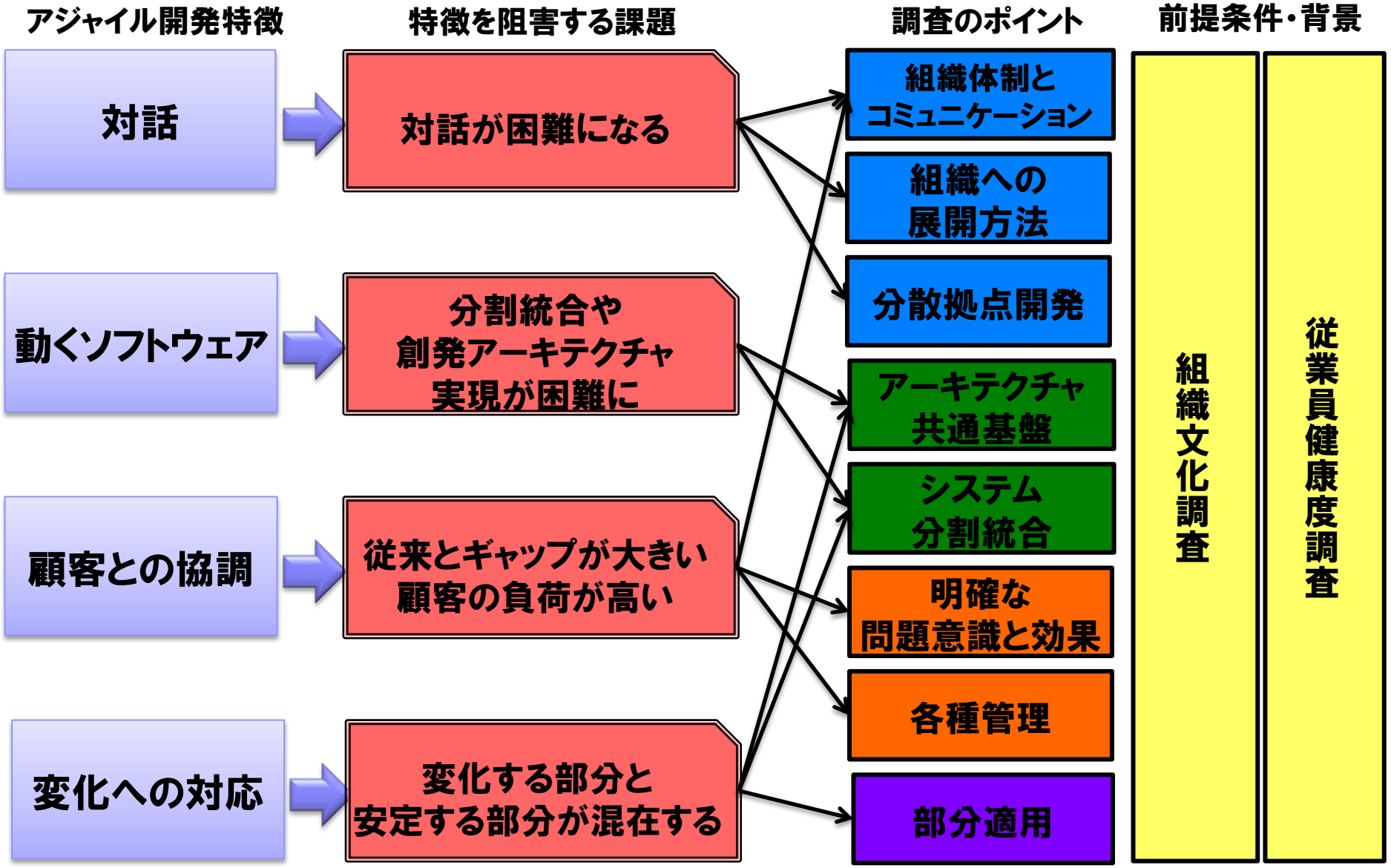
目的



国内において中大規模開発における事例を調査し成功の秘訣を収集する

大規模における調査課題とポイント

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



調査ポイントと想定結果

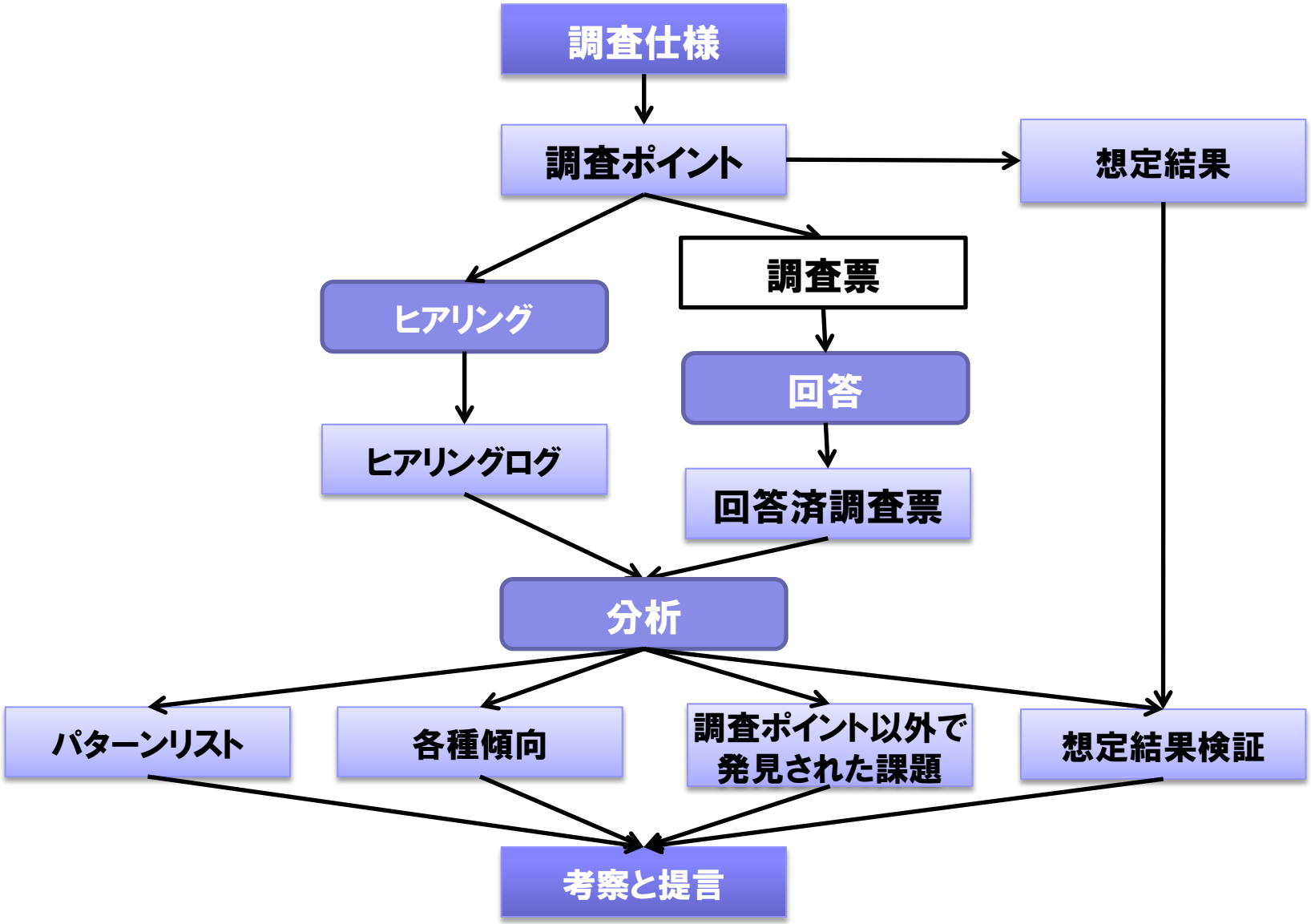
SEC
Software Engineering
for Mo·No·Zu·Ku·Ri

背景・問題の明確化

組織体制とコミュニケーション	チームはレイヤ毎ではなく機能毎に構成されている。 段階的朝会などでチーム間のコミュニケーションを効率化している。
組織への展開方法	試行チームから始め部分的に導入し段階的に全体へ展開している。 強いリーダーシップを持つ人物がいる。
分散拠点開発	同一拠点で作業して、次の段階で分散してチームを作る。IP電話会議システム、チャットなどコミュニケーションツールを駆使している。
アーキテクチャ共通基盤	アーキテクチャ自体も非ウォーターフォール型で段階的に構築されている。アーキテクチャをサポートする専門チームが構成されている。
システム分割統合	サブシステムを疎結合な機能で分割している。早期からサブシステム間で継続的インテグレーションを実現している。
明確な問題意識と効果	経営、顧客側が明確な問題意識を持ち導入している。 期待とその結果の効果測定が明かである。
各種管理	顧客とは準委任契約でスコープ管理を行っている。 QAテストフェーズを設けて品質管理を行っている。
部分適用	変化の必要な部分にのみ適用している。 同期ポイントを反復のタイミングで合わせている。

組織文化調査	従業員健康度調査
組織文化が変われば文化的問題が発生。 組織文化が似ていれば問題は発生せず。	ウォーターフォール型よりも非ウォーターフォール型の方が精神健康度が高い。

調査方法



調査対象

中規模および大規模なプロジェクト

本調査では、プロジェクトに参加した人数で区分し調査する。

- 中規模（30 名前後～100 名未満）
- 大規模（100 名以上）

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。

部分適用プロジェクト

全工程を非ウォーターフォール型開発で実施したプロジェクトに限定せず、他の開発手法と組み合わせた「部分適用」の事例についても収集する。

分散拠点開発プロジェクト

異なる企業間で構成される合同プロジェクトやオフショア開発を採用した等の理由により、「開発拠点が分散配置」される開発プロジェクトの事例についても収集する。

事例一覧

（各事例の概要は【別紙】を参照のこと）

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

調査先	規模	部分適用	採用手法［※1］	システム種別	契約関係［※2］
A社	大規模		独自	B2Cサービス（SNS）	自社開発
B社	大規模		スクラム	B2Cサービス（ソーシャルゲーム）	自社開発
C社	大規模	○	スクラム	ゲームソフト	受託開発（契約形態:非公開）
D社	大規模	○	スクラム+独自	基幹システム	受託開発（準委任）
E社	中規模		スクラム	B2Cサービス（会員サービス）	自社開発
F社-1	中規模		スクラム+XP	B2Cサービス（医療・健康）	自社開発 + オフショア（準委任）
F社-2	中規模		スクラム+XP	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）
G社	中規模		XP	B2Cサービス（会員サービス）	受託開発（請負）
H社	中規模	○	XP	B2Cサービス（ECサイト）	受託開発（請負）
I社	中規模	○	XP	B2Cサービス（会員サービス）	受託開発（準委任）
J社【参考】	小・中規模組織展開		独自	B2Cサービス（会員サービス）	自社開発
K社【参考】	小規模組織展開		スクラム	B2Cサービス（SNS）	自社開発

大規模（100名以上）:**4件**

中規模（30～99名）:**6件**

部分適用事例:**4件**

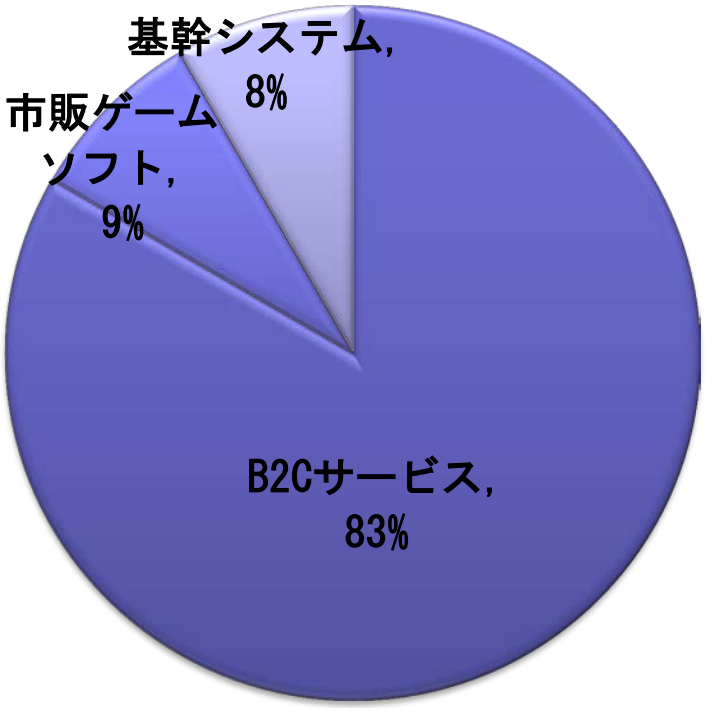
※1:スクラム，XP → 両手法をベースにカスタマイズした事例も含む
スクラム+XP → 両手法を組み合わせて実践している事例
独自 → 特に手法を決めない、自分達で定義

※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

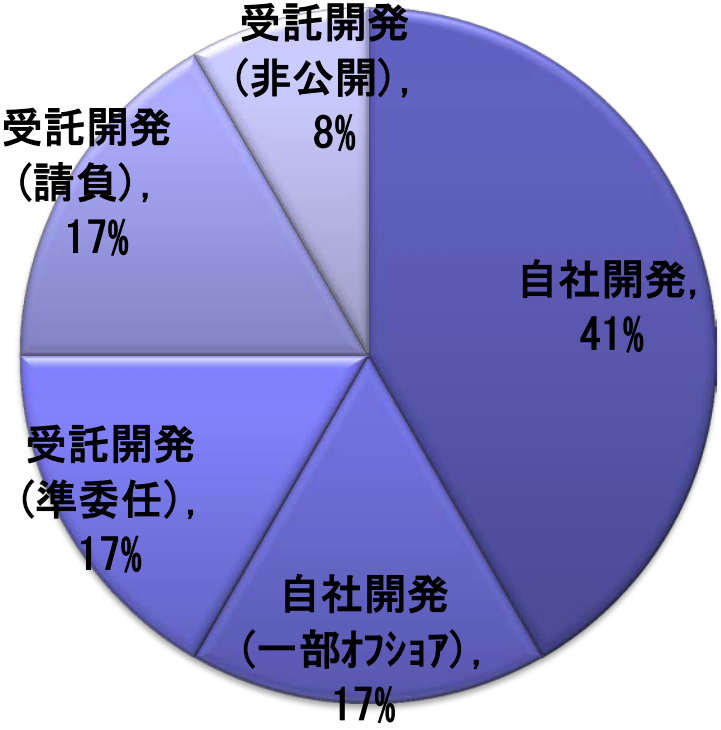
適用分野

今回、調査対象としたプロジェクトの大半（80%以上）がインターネット上のB2Cサービス開発に非ウォーターフォール型を適用している。また、半数以上が自社組織内に開発部隊を抱える形態（事業者と開発者のあいだに契約関係がない形態）を採用していた。

システム種別



契約関係



非ウォーターフォール型開発の採用理由

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

ウォーターフォール型開発で
プロジェクトが失敗した

理由 No.1
要件が固められない

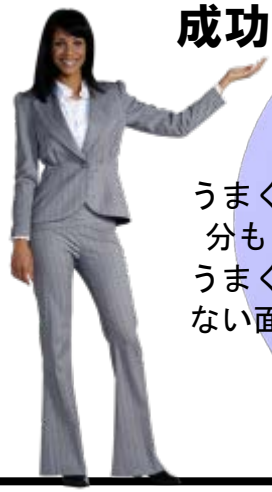
実際のサービスや製品を見ないと
ビジネス価値を評価できない

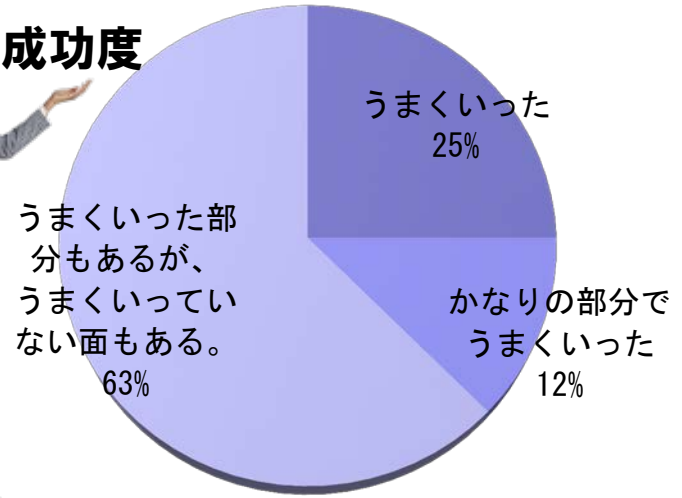
理由 No.2
管理コストを下げたい

理由 No.3
市場や社会の
変化に対応したい

非ウォーターフォール型開発の選択

効果は…
1. 動くモノを見ながら開発できた
2. 管理コスト削減
3. 迅速なサービスの提供
でした





想定結果と調査結果の比較

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

組織体制とコミュニケーション

- チームには企画、開発、テスト毎の人々が集められている(職能横断チーム)。
- 効果的な情報共有を実施している。
  - 見える化
  - 頻繁なチーム間の情報共有
  - オープンなスペースに全員同席
- 徐々にチーム人数を増やしている。

- 同一拠点での開発では半数以上が同一フロアに人を集めている、あるいは段階的に近づける
- 壁やホワイトボードを大きくとりタスクボードなどを貼り全員で共有した(半数以上)
- 7割以上が職能横断チーム
- チーム間の情報共有を実施
  - 段階的朝会を実施
  - チームメンバーのローテーション
- 半数近くの事例では徐々に人数を増やしていっている

組織への展開方法

- プロジェクト全体で一斉に非ウォーターフォール型開発を適用するのではなく、試行チームで評価した上で、適用チームを徐々に拡大している。
- 同一プロジェクトの場合でも、変化の度合いが高い部分から中心に適用し、広げている。
- 組織文化を変遷させるためのリーダーシップが存在する。

- 4割以上の事例でプロジェクトへ段階的導入を選択
  - 導入障壁が低い所
  - 必要な所
- 4割の事例で認定スクラムマスターの資格を取得し活用
- 2割の事例で外部の人間(コーチ、コンサル)を利用して導入
- ほぼすべて事例においてリーダーシップを発揮している人物がいる

想定結果と調査結果の比較

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

分散拠点開発

- 同一拠点で作業して、次の段階で分散してチームを作る。
- IP電話会議システム、チャットなどコミュニケーションツールを駆使している。

- 分散事例の8割において、最初は同一拠点で始め、一緒に作業した後に拠点を分散した。
- 分散拠点開発のうち4割の事例ではテレビ会議システムを使っていた。
- IP電話会議システム、チャットシステムといったツールは不可欠

アーキテクチャ共通基盤

- アプリケーション部分とは異なるサイクルで、共通基盤、アーキテクチャも漸進的に設計開発されている
- アーキテクチャ、共通基盤を担当するチームが存在する。

- 半数以上の事例が既存の自社フレームワークを再利用している、あるいは、標準的なオープンソースフレームワークを利用している。
- 新規で作り込む場合は最初にトップエンジニアが設計構築している。
- サービス成長過程でシステムアーキテクチャが大きく変更されるケースあり。
- アーキテクチャ、共通基盤チームが存在して開発チームの支援を行っている

想定結果と調査結果の比較

システム分割と統合

- サブシステムを疎結合な機能で分割している。
- 早期からサブシステム間で継続的インテグレーションを実現している。
- チームはそのサブシステムが持つ機能やサービス毎に分かれている。(フィーチャーチーム)

- 半数近くがフィーチャーチームであった。
- 2割強がシステム結合部分を優先的に開発し早期にインテグレーションを開始した
- CI(継続的インテグレーション)を実施しているチームは7割(試行中チームを含めて9割)。
- ソフトウェア開発チームと画像などデータ作成チームを疎結合にして別々に作業できるように構成した

明確な問題意識と効果

- 経営者や顧客が問題意識、考慮すべき制約、期待を明確にした上で適用判断をしている。
- 期待に対しての、効果測定が、きちんとなされている。
- 期待が明確である場合は、成功度合いが高い。

- 問題意識は開発側が持つケースが多かった
  - 受託の場合顧客に提案→受理
  - サービス系はビジネス特性上不可欠
- 期待は非常に明確だった。管理コスト低減、変化への適応で全体の8割
- 期待に対しての8割以上が期待通り
- プロジェクトの成功度(五段階)[※] 平均 3.7 (5を上げた事例は3つ)

※「【別紙】プロジェクトの成功度の基準」参照

想定結果と調査結果の比較

各種管理

- 自社開発もしくは外部ベンダに委託する場合は契約を準委任契約にしてスコープ管理を容易にできるようにしている。
- 顧客の負荷軽減のため顧客プロキシ[※1]を配置している。
- テストフェーズを設けてQAテスト[※2]を実施している。

- 半数以上が自社サービス開発であった
- 受託開発の場合は準委任契約は一例だけであり、他は請負契約であった。
- 顧客役の高負荷が課題になっている例が複数あった
  - ●顧客支援チームで軽減している事例もあり
- 大半の事例が反復とは別にテスト工程を設けていた

部分適用

- 変化の必要な部分にのみ適用している。
- 同期ポイントを反復のタイミングで合わせている。

- 非ウォーターフォールを適用しないチームの条件
  - ●変化がそれほどないドメイン
    （データ作成、変更のない業務、メールエンジン）
- 非適用チームと適用チームが疎結合になるように
- 非適用チームにおいても、壁を使っての工程の見える化(カンバン)で情報共有を計る(1事例)
- 規模が大きくなったり要求品質が高くなったりした場合は非ウォーターフォールからウォーターフォールへチューニングする(1事例)

※1:顧客の代表者、もしくは、プロジェクトに参加してもらえない顧客の代理人
※2:ソフトウェア品質の測定・確保・保証を目的として行われるソフトウェアテスト

想定結果と調査結果の比較

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

組織文化の移行

- 非ウォーターフォール型の適用と展開と共に、組織文化の移行が発生している場合は問題が起きている。
- 組織文化が非ウォーターフォール型に近い場合は、特に問題は起きていない。

- 自社サービス開発企業は組織文化と非ウォーターフォール型の文化の親和性が高く問題が発生していない
- 自社サービス開発以外の事例の8割が非ウォーターフォールに疲れる・合わない人が存在する
  - 事例毎に1～2割の人が適合しない
  - 合わない人は外す/抜ける選択をしている

従業員健康度の比較

- ウォーターフォール型よりも、非ウォーターフォール型でのプロジェクト従事者の方がストレスが低く、生きがいも高い。

- 非ウォーターフォール型開発の従事者は、ストレス度が低く、生きがい度が高かった
  - ただし合わない人が異動されているケースはある

中大規模事例での工夫一覧 （各工夫については【別紙】を参照のこと）

中大規模特有の工夫

■組織体制
- チーム間ローテーション

■コミュニケーション
- 段階的朝会
- チーム跨ぎのふりかえり

■展開
- 漸進的な人数増加
- 漸進的な展開
- 社内勉強会

■分散拠点開発
- 同一拠点から分散へ
- TV会議

■アーキテクチャ
- 組織の共通基盤アーキテクチャの利用
- アーキテクチャについての教育

■システム分割/インテグレーション
- 同じリズム

■品質
- 第三者テスト

■部分適用
- 必要な部分のみ適用
- 疎結合なチーム
- 工程の見える化

小規模とは逆のアプローチをとる工夫

■アーキテクチャ
- 最初のアーキテクチャ構築
- アーキテクチャ専門チーム
- 運用保守チーム

■品質
- テスト・フェーズ

小規模と同様だが特に注意して実施する工夫

■コミュニケーション
- 完全透明性

■展開
- パイロット導入
- 認定研修・コンサルタントの利用

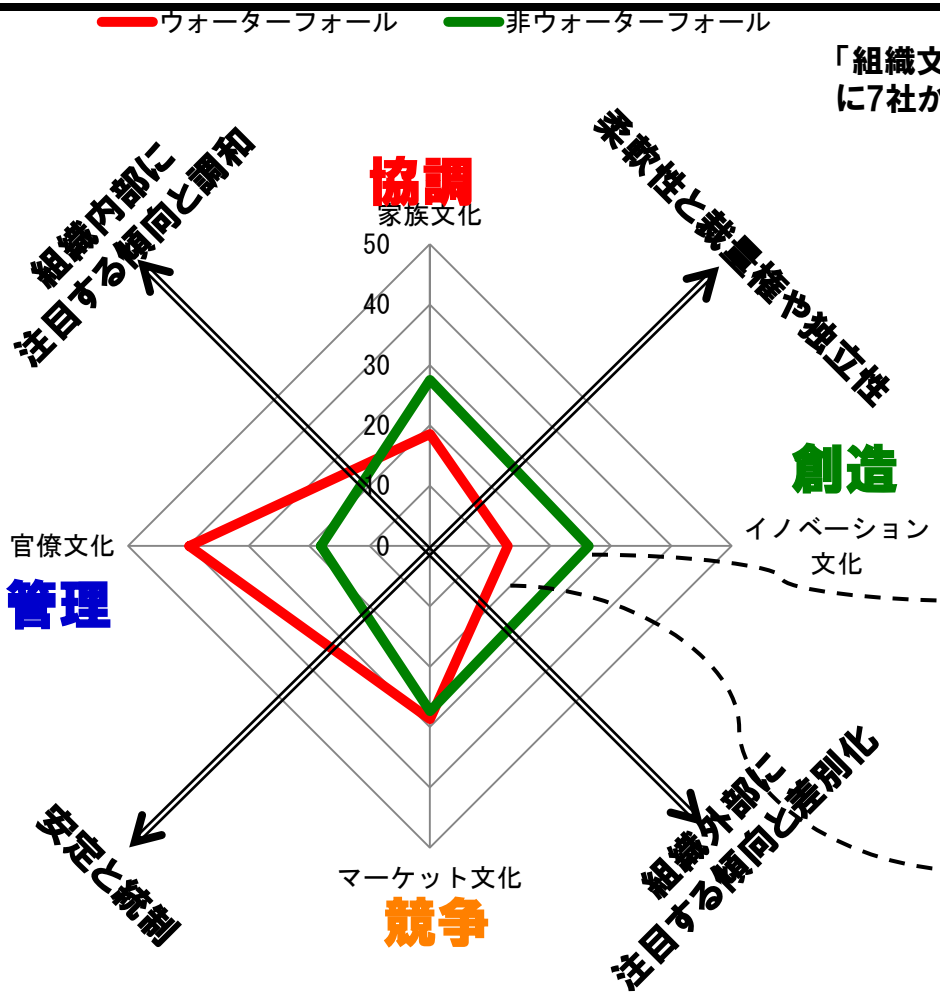
■分散拠点開発
- チケットシステム
- リアルタイムチャット

■アーキテクチャ
- アーキテクチャの改善

■システム分割/インテグレーション
- 疎結合で分割
- 早期からのインテグレーション
- 継続的インテグレーション

■品質
- 重視するビジネス価値
- ビジネス価値の変化
- タイムボックス優先の品質
- 自動単体テスト

組織文化の傾向



「組織文化診断ツール」(OCAI http://www.ocai-online.com/) を元に7社からの個別アンケートを回収集計した。

非ウォーターフォール型を適用することによる組織文化の変化を調査し主に2つの変化の傾向が見受けられた。

非ウォーターフォール型
「より協調、創造の傾向が高く
管理傾向は低い」

ウォーターフォール型
「より管理傾向が高く
協調、創造の傾向が低い」

管理から協調、創造へ

有効データ (7社から回答: 非ウォーターフォール:81件、ウォータフォール:53件)

従事者の健康度

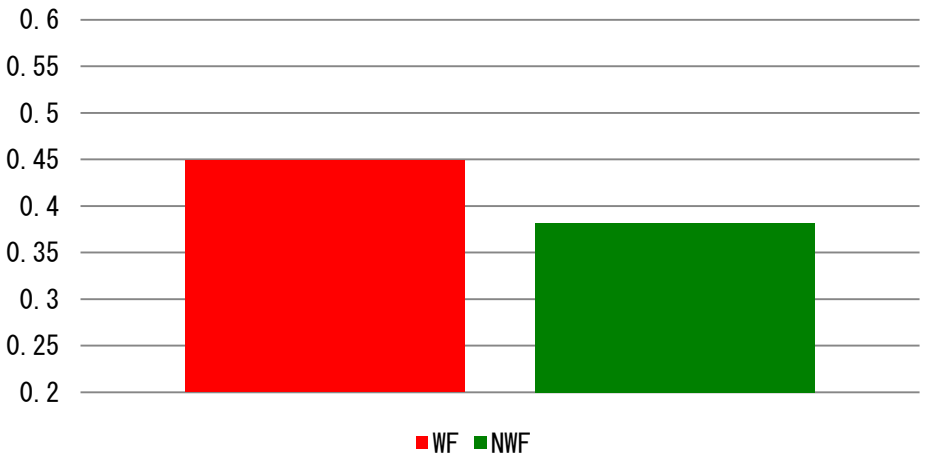
SEC
Software Engineering
for Mo·No·Zu·Ku·Ri

いきいきしているか（健康度）をストレスが少なさと生きがいで評価した

ストレスが少ない
心身と社会的な緊張状態

生きがいがある
生活の質を感じるか

ストレス度　p ＜ 0.05



VAR	Sample Size	Mean	Variance
Waterfall (WF)	54	0.44856	0.02697
Non-Waterfall (NWF)	81	0.38203	0.02434

生きがい(QoL)度　p ＜ 0.05

VAR	Sample Size	Mean	Variance
Waterfall (WF)	54	0.49228	0.0365
Non-Waterfall (NWF)	81	0.64609	0.0336

ウォーターフォール型開発(WF)および非ウォーターフォール型開発（NWF)それぞれにおいて
ストレスおよび生きがい(QoL: Quality of Life)の度合いを質問紙法を用いて調査した
調査対象のうち、6組織 81名（NWF n=81,WF n=54）から有効回答を得た

ウォーターフォール型開発に比べ、いきいきしている

調査ポイント以外で発見された課題

企画

全体計画の把握困難
非ウォーターフォール型開発では、要求の変化や開発状況に応じて着手する順番や範囲を決めるためプロジェクト開始時にプロジェクト全体の把握が困難な事例があった

新手法へ期待過大
非ウォーターフォール型開発の推進者や開発現場によっては、手法の長所のみを理解し、欠点や潜在リスクを考えずに導入し、開発中に問題が顕在化する事例があった

導入

手法移行の問題
ビジネス企画者および開発者側双方に非ウォーターフォール型開発のスキルや経験が不足し、導入以前の開発手法から移行困難もしくは導入に逆行する圧力があった

ビジネス企画側にボトルネック発生
スクラム導入の結果、ビジネス企画者で決定待ちなどのボトルネックが発生した事例が多くあり、開発者が信頼しなくなる事例もあった

開発

反復中の品質悪化のしわよせ
反復漸進的にシステムを構築していくが、そこで発生した障害を放置などによって品質が悪化し、後半に対応に追われてしまった。

他組織とのリズム不調和発生
セキュリティ監査や外部テスト業者、発注者の外部組織や部分適用のウォーターフォール型で進めているチームといった関係組織との間で反復リズムと適合せずに問題が発生している事例があった

管理工数増加や開発速度低下
市場の状況や利用者から要望をダイナミックに対応する事例については、管理工数増加や開発速度低下が心配された事例があった

中大規模受託開発の現状と適用ポイント

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

契約

【現状】準委任と請負の事例が半数ずつであった。
【適用ポイント】中大規模開発では小規模開発よりも大きなスコープ調整が行われることがあり、準委任契約のほうが適している。請負契約の事例では、現場レベルで作業内容を柔軟に入れ替えることができる約束をしていることが多い。

顧客の理解

【現状】いずれも開発ベンダが起点となって非ウォーターフォール型開発を進めているが、発注者に非ウォーターフォール型開発を適用することを宣言している事例と発注者には言わず開発ベンダ内部のみで適用している事例の2パターンがあった。
【適用ポイント】顧客に非ウォーターフォール型開発への理解があればあるほど、開発ベンダが顧客への理解を求めたり、発注者のやり方に合わせるための非本質的な作業が省けるため、オーバーヘッドが削減され、開発が効率化できる。

人材育成

【現状】自社開発では、ほとんどの事例で開発者に認定研修を受講させていたり、外部のコーチを雇ったりしているが、受託開発ではこのような事例はほとんどなかった。
【適用ポイント】発注者と開発ベンダの関係は、プロジェクト期間中の一時的なものであり、発注者が開発ベンダの開発者を育成するという観点は存在しない。中大規模開発に非ウォーターフォール型開発を適用する前提として発注者と開発ベンダの長期的なパートナーシップが必要である。

まとめ

中・大規模の非ウォーターフォール型開発において、成功に寄与する要因を以下にまとめる。

- リーダシップを発揮するキーマンが重要である
- 教育と経験が重要である
  - 教育コースを受講する、社内で勉強会を実施する
- コミュニケーションとリスクに配慮し、段階的に導入する
  - リスクが限定されたプロジェクトで試行し評価する
  - 原則としてチーム全員が同一拠点・同一フロアに集まる
  - チームメンバーをローテーションする
  - 分散拠点開発の場合は、広帯域のコミュニケーションが行えるようツールを整備する
- システム分割の際は、疎結合にし、頻繁にインテグレーションする
- アーキテクチャは事前に設計する
- 文化と健康にも配慮する
  - 管理よりも協調、競争よりも創造を大切にする企業文化を醸成する

中大規模の非ウォーターフォール型開発における課題と目指すべきゴール

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

	課題	提言
法・環境	事業者と開発者のあいだに契約やセキュリティ基準の壁や物理的な距離による壁がある	■ 複数企業、多国籍、多重構造中大規模に適した契約テンプレート、コンプライアンスやセキュリティ基準、監査基準の整備と提示する
プロジェクト	事業者のソフトウェア開発への理解とコミットメントが乏しいため、中大規模開発に最適な開発手法の選択やプロジェクトの進め方できていない	■ 参加型意思決定手法の紹介・導入支援 ● 事業者と開発者の関わり方やプライオリティ付けの手法を学び、実践できるようにする ● 大規模開発であっても、一気に大きくせずに漸進的な展開方法を提示する
人材	非ウォーターフォール型開発ができるスキルを持った開発者を一定人数以上、集めることが難しい	■ 非ウォーターフォール型開発実践者の育成と人材確保 ● 認定研修・コンサルタント活用・社内外の勉強会への助成と人材プールを構築する
組織	中大規模開発では組織を含めた改革が必要とされることがあるが、成功へ導く組織構造と、変革のプロセスが明確ではない	■ 「開発手法を変える」ではなく「組織文化を変える」視点へ移行する ● 組織評価とデザイン手法を立ち上げる
リファレンス	中大規模開発に関して、日本の産業構造や文化にあった開発手法がまとめられていない	■ 非ウォーターフォール型開発リファレンス作成 ● 中大規模開発の幅広く事例を収集し、工夫や指針の提示が必要 ● ビジネス、文化や組織構造の調査と設計手法の策定

目指すべきゴール

- 日本のソフトウェア産業の実態に適したソフトウェアの作り方を提案する
- 日本のソフトウェア産業の競争力を高める
- エンジニアが生き生きと働ける環境を作る

【別紙1】事例概要（1）

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

A社	大規模（103名）	インターネット上のB2Cサービス開発（SNS）

- サービスを小さく作って大きく育てる自然な成長を伴う大規模適用
- アジャイル/スクラムをあまり意識していない
- 人数の多いデザイナー/イラストレーターと、開発の間を疎結合にしておくことで人数増に対応

B社	大規模（100名）	インターネット上のB2Cサービス開発（ソーシャルゲーム）

- 段階的朝会により複数チームが連携
- ビジネスのミッション（ビジネスタスク）によってチームを分割
- 企画・開発・デザイナが一体となりチームを構成、スクラムで定義されているプロダクトオーナー・スクラムマスター・開発者というロールを超えて活動

C社	大規模（100名）	ゲームソフト開発

- 開発者が現場からボトムアップで徐々に導入
- 開発者・企画者・サウンド担当者・デザイナの一部がスクラムを実施（**部分適用**）
- 発注者にはスクラムであることは言わずに内部的に実施、発注者との連携（進捗報告等）は窓口となる担当者を設置することにより解消

【別紙1】事例概要（2）

D社	大規模 （100名以上）	基幹システム再構築

- 要求が変わりやすいフロントの部分を中心にアジャイルを適用、要件が固まっているバックエンドの部分はウォーターフォール型開発（**部分適用**）

E社	中規模（40名）	インターネット上のB2Cサービス開発（会員サービス）

- 開発部門の事業責任者が開発全体を取り仕切ってスクラムを導入、チーム全員の進捗をチェックするなど強力なリーダーシップを発揮
- 職能横断チーム[※1]ではなく、職能別チーム[※2]となっており、五月雨式にタスクを流している

F社-1	中規模（40名）	インターネット上のB2Cサービス開発（医学・健康）

- 会社のトップ（執行役員）がトップダウンでスクラム導入を推進
- 手はじめにIT部門と企画部門を統合する社内改革を実施、その後、2つのプロジェクトで成功事例を作って、全社展開（この事例はそのうちの1つ）
- 中国へのオフショアを積極活用（**分散拠点開発**）

※1:企画、開発、デザイナのような異なる職能をもった人々を集めて構成されたチーム
※2:単一の職能を持った人々を集めて構成されたチーム

【別紙1】事例概要（3）

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

F社－2	中規模（50名）	インターネット上のB2Cサービス開発（エンターテイメント）

- F社－1の成功をうけて全社展開が決まったことに伴い、F社－1の事例を後追いする形で導入
- 中国へのオフショアを積極活用（**分散拠点開発**）

G社	中規模（50名）	インターネット上のB2Cサービス開発（実証実験）

- 顧客と開発との間の透明性を最大限に実現し信頼関係を構築
- チーム間でメンバーローテーションを実施して知識を伝播
- 最新技術、ツールを使いこなしての高速開発のため脱落者も出る
- 国内数拠点での**分散拠点開発**

H社	中規模（50名）	インターネット上のB2Cサービス開発（ECサイト）

- 特定のサブシステムのみアジャイル開発を適用、他はウォーターフォールで開発したという部分適用事例
- 開発会社がアジャイル開発を提案
- 国内数拠点での**分散拠点開発**

【別紙1】事例概要(4)

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

I社	中規模（50名）	インターネット上のB2Cサービス再構築（会員サービス）

- **部分適用**、国内数拠点での**分散拠点開発**
- スコープの見直しに柔軟に対応するためにアジャイルを適用
- 一気に人数を増やすのではなく、チームを徐々に拡大
- 顧客プロキシを設けることで顧客側のボトルネックを解消

J社	組織展開	インターネット上のB2Cサービス開発（会員サービス）

- アジャイル開発の要素を取り入れた標準プロセスをつくって全社に展開
- 規模が30～40人になると、ウォーターフォール型開発の要素を取り入れたカスタマイズ（仕様変更のコントロールのやり方を変える、テスト項目やドキュメントのボリュームを増やす）を行って対応

K社	組織展開	インターネット上のB2Cサービス開発（SNS）

- アジャイル導入を推進する横断組織が存在、コーチ役を果たす
- この企業の文化・風土がアジャイルなやり方との親和性が高い
- Ad-Hocなやり方（できたものから、都度リリースしていくようなやり方）からスクラムに移行

【別紙2】 中大規模事例でよく行われた工夫

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

- 各種事例の調査の結果、よく行われている工夫を調査ポイント毎にまとめた。
- 上から時系列で適応するタイミングを表現する
- [大] は中大規模特有の工夫
- [逆] は小規模とは逆のアプローチをとる工夫
- [注] は小規模と同様だが特に注意して実施する工夫
- [同] は小規模と同様に実施する工夫

【別紙2】組織体制についての工夫

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

契約

準委任契約 [同]

組織間で契約を行う場合には、顧客との間、開発パートナーの間共に準委任契約を行う。その結果変化に適応する開発を実施しやすくなる。

体制

職能横断チーム [同]

専門的役割別でチームを構成するのでなく、顧客役を含めたある機能を実現するために必要な様々な職能を持つ人間がチームを組む。その結果チーム感が増す。

役割を超えた支援 [同]

ビジネス企画者などボトルネックになりがちな作業を支援するための役割を設けて支援する。その結果、ボトルネックが解消し、バックログの提供、受入レビューなど全体が円滑に進捗できる。

変更

チームメンバーローテーション [大]

チーム間の知識伝播を促進するために、メンバーのローテーションを行う。一時的に速度は落ちるが結果として各チームの知識を効率よく伝播でき多能工化が高まる。

不適合メンバーの入れ替え [同]

チームメンバーの中には非ウォーターフォール型開発に合わない人がいる。その場合は無理に残すのではなく、メンバーを入れ替えて互いにストレスなく進める。

【別紙2】コミュニケーションについての工夫

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

ファシリティ

全員同席 [同]

拠点内において、顧客役、開発に関わらず、物理的にできるだけ近くに集めてコミュニケーションをとりやすくする。複数チームも含む。その結果、リッチなコミュニケーションが実現できる。

見える化 [同]

壁やホワイトボードに作業状況などを一目でわかるように貼り出し共有する。積極的にアナログツールを使う。その結果チーム内で情報や認識を共有できる。

共有

段階的朝会 [大]

複数チーム間は朝会を段階的に実施する。チーム→全体（→チーム）代表者だけで集まってもよいがスペースがある場合が全員集ってもよい。その結果全員が今の状況を共有できる。

完全透明性 [注]

ソースコード、進捗状況、開発者の日記、テスト結果などすべての情報を顧客役も含めて共有する。隠し事はしない。その結果誰もが状況を把握でき、互いの信頼関係を構築できる。

チーム跨ぎのふりかえり [大]

個々のチームのふりかえりの結果をチーム間で発表したり共有する。各チームでの知識や改善が伝播でき、そこから共通的なルールを導くこともできる。

【別紙2】展開についての工夫

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

準備

パイロット導入 [注]
最初から大規模には展開できないので、まず小さなチームで実際に試してみて評価する。結果が出たら実際の導入展開に進む。

認定研修・コンサルタントの利用 [注]
我流で実践すると、どうしていいかわからないことが多い。認定研修への参加、外部の
やコーチを招聘する。その結果、どのように振る舞うかを学ぶことができる。

導入

漸進的な人数増加 [大]
一度に人を増やさず、反復リズムと同じタイミングで徐々に増やしていく。少しづつチームメンバーと知識を いくことで無理ない拡大を実現することができる。

展開

漸進的な展開 [大]
一度にすべてのチームに広げるのではなく、まずは導入障壁の低い所、最も必要な所から導入し、少しづつ展開していく。ふりかえりで次はどこに広げていけばいいかを考える。

社内勉強会 [大]
自分達で勉強会を開催し、まだ非ウォーターフォール型開発を知らない人に伝える。何故必要なのかのロジックや、ワークショップでの体感で伝えることで納得感を持ってもらう。

【別紙2】分散拠点開発についての工夫

開始

ツール

同一拠点から分散へ［大］
最初は同一拠点に集ってチームで作業をする。その後拠点を分散して、それぞれでチームを組む。同じ拠点で仕事をして信頼関係が生まれコミュニケーションがとりやすい。

チケットシステム［注］
同一拠点で実現できる見える化は利用できない。そのためチケットシステムは必要不可欠である。特に残りの要件を管理するために利用される。（※1）

TV会議［大］
常時クリアな映像と音声のTV会議システムを用いることで、拠点が別であっても、対話コミュニケーションが実現できる。

リアルタイムチャット［注］
気軽にコミュニケーションをとるためにチャットは必須となる。IP電話会議システム、チャットシステムを用いてリアルタイムにメッセージ交換を可能にしておく。

※1. Redmine, JIRA, Pivotal Trackerなどが好まれている

【別紙2】アーキテクチャについての工夫

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

構築

最初のアーキテクチャ構築 [逆]
中大規模では、プロジェクト前半にアーキテクチャを構築する事例が多く、
業務にあわせたアーキテクチャを選択・構築している

アーキテクチャ専門チーム [逆]
アーキテクチャ専門チームを編成し、アーキテクチャの構築を行う。業務アプリケーションに合わせたアーキテクチャの提案も行う。

利用

組織の共通基盤アーキテクチャの利用 [大]
構築されたアーキテクチャを、組織の共通基盤とし、再利用できるようにしている。その結果、新規でプロジェクトとして構築する手間が省け、かつこなれているため

アーキテクチャについての教育 [大]
業務アプリケーションの開発チームや運用保守チームに対して
アーキテクチャについての教育や支援を提供する。

保守

運用保守チーム [逆]
アーキテクチャの運用保守チームにより、運用監視や保守業務を行う。
アプリケーション開発やビジネスオーナーと密に一体になる事例もあった

アーキテクチャの改善 [注]
利用しているアーキテクチャが、要件を満たさない場合は、プロジェクト途中であってもアーキテクチャの改善を少しづつ行っていく。複数のチームで協力して対応していく。

システムの基本構造であり、コンポーネント群、コンポーネント間の相互関係と環境との関係、設計と改良を律する原理・原則により構成される

【別紙2】システム分割/インテグレーションについての工夫

設計

疎結合で分割 [注]

システムを疎結合で分割を行う。単純に疎なサブシステムというだけでなくアプリ部分とプラットフォーム部分（PaaS）のような分割方法もある。

開発

フィーチャーチーム [同]

チームはレイヤー（UI、ビジネスロジック、DB）や工程（要件定義、設計、開発、テスト）で分割するのではなく、ある機能やサービス単位でチームを作る。その結果動くソフトウェアを作りやすくなる。

早期からのインテグレーション [注]

システムインテグレーションが必要な要件を洗い出して優先的に実施していく。その結果早期からシステムインテグレーションが可能になりビックバンを防ぐことができる。

同じリズム [大]

複数チームと連係する際には各チームの反復のリズムを合わせておく。その結果互いのリズムを崩さずに自然にインテグレーション可能なスケジュールを組むことができる。

結合

継続的インテグレーション [注]

単体自動テストを常に動作させておき、不具合を早期対応できるようにしておく。その結果システムインテグレーション時の問題を減らし、インテグレーションも常時実行できる。

【別紙2】品質についての工夫

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

非ウォーターフォール型開発を実施する上で、品質、品質管理と品質保証についての工夫を紹介する

企画

重視するビジネス価値/ビジネス価値の変化 [注]
置かれている環境で何が重要なビジネス価値であるかを認識し、更にその変化に追従できるようにしておくことで、最適なQCD+スコープのバランスをとることができる。

タイムボックス優先の品質 [注]
ビジネス価値を設定した上で、品質の位置付けを決め、タイムボックス内での完成を優先する。その結果、システムにおける適切な品質によって期間内に動作するソフトウェアが受入可能になる。

開発

ピア・レビューの実施 [同]
仕様書やプログラムに対してピア・レビューを実施し、品質を向上させる

自動単体テスト [注]
単体テストはテストツールを用いて自動化しておく。そのため、継続的インテグレーションのタイミングでリグレッションテストが実行できる。

テスト

テスト・フェーズ [逆]
反復とは別に、プロジェクト後半でテストフェーズを設けて実施する。

第三者テスト [大]
セキュリティや法務などの専門家によるチェックや、テスト専門業者によるテストによって、チームでは対応できない部分に対して品質保証する。

【別紙2】部分適用についての工夫

判断

必要な部分のみ適用 [大]
多くのサブシステムの中で変化が激しい、要件が固められない、といった必要な部分に絞って適用する。逆に安定していて変化しない部分には無理に適用しない。

開発

疎結合なチーム [大]
非ウォーターフォール型開発を適用するチームと、非適用チームの関係はできるだけ疎結合にしておき、関連せずに並行して進むことができるようにしておく。

工程の見える化 [大]
ウォーターフォールチームであっても、今の作業が具体的にどの作業工程であるかを壁に貼り出して見えるようにすることで、今の状況を全員で把握できるようになる。

【別紙3】プロジェクトの成功度の基準

プロジェクトの成功度については、下記の5段階の自己評価で調査先から回答を得た。

- 5: うまくいった
- 4: かなりの部分でうまくいった
- 3: うまくいった部分もあるが、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