JaSST'23 北陸

品質重視のアジャイル開発
～生き残りの鍵は、技術力と定量化～

2023年　1月20日

株式会社イデソン
誉田　直美

講師紹介

氏　名：誉田　直美（ほんだ　なおみ）
現　職：株式会社イデソン　代表取締役
　　公立はこだて未来大学　客員教授　　博士（工学）

略　歴：
ソフトウェア品質の専門家として、30年以上に渡って大手電機会社で活躍。ウォーターフォールモデル開発およびアジャイル開発の両方に精通し、現場での豊富な経験を保有している。近年では、AIシステムの品質保証にも関わる。単なる知識にとどまらず、現場の事情を考慮した実践的な品質保証が特徴である。
2020年、㈱イデソン設立。ソフトウェア品質に関するコンサルティングを中心に活動している。執筆や講演による啓蒙活動にも力を注いでいる。https://ideson-worx.com/

主な著書・執筆活動：
品質重視のアジャイル開発　～成功率を高めるプラクティス・Doneの定義・開発チーム編成～　（日科技連出版） 2020年9月発行
ソフトウェア品質判定メソッド　～計画・各工程・出荷時の審査と分析評価技法～　（日科技連出版、編著） 2019年8月発行
ソフトウェア品質会計（日科技連出版）2010年発行　<2010年度　日経品質管理文献賞受賞>
ソフトウェア品質知識体系ガイド 第３版–SQuBOK Guide V3- (オーム社、監修) 2020年11月発行。なおV2（ 2014年11月発行）は執筆リーダー、V1（2007年11月発行<2008年度　日経品質管理文献賞受賞>）は著者として執筆に継続的に関与
ソフトウェア開発 オフショアリング完全ガイド（日経BP社　共著）2004年10月発行
見積りの方法（日科技連出版、共著）1993年

受賞：
品質管理学会　品質管理推進貢献賞（2020/11）
プロジェクトマネジメント学会　文献賞（2016/9）
第5回世界ソフトウェア品質国際会議（5WCSQ）最優秀論文賞および最優秀発表賞（2011/11）
第4回世界ソフトウェア品質国際会議（4WCSQ）最優秀論文賞（2008/9）

学会：情報処理学会、品質管理学会、プロジェクトマネジメント学会

委員活動：
日科技連SQiPソフトウェア品質委員会　副委員長
プロジェクトマネジメント学会　上席研究員
筑波大学大学院　非常勤講師（2012年～2016年）　鳥取大学　非常勤講師（2017年）

目次

１．なぜ今、アジャイル開発か
～アジャイル開発とは～

DXは新たなステージへ

■ DX化の重点は、業務の効率化→ビジネスモデルの変革へ

あらゆる利用者層における
デジタル活用の浸透

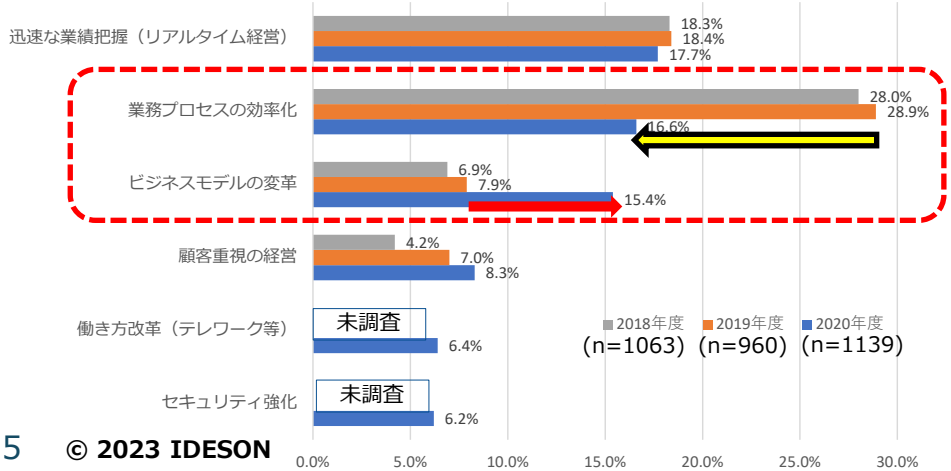
デジタル導入による
企業・自治体等の業務効率化

デジタル活用による
経済再生と社会課題の解決の実現

ビジネスモデルの変革を伴う
新サービスの創出

<IT投資で解決したい中長期的な経営課題1位の経年変化>

※令和3年総務省情報通信白書に基づき、筆者が作成



- 2020年には、「業務プロセスの効率化」が低下、「ビジネスモデルの変革」は大きく伸長
  - コロナ禍が始まった2020年前半は「働き方改革(6月)」が経営課題だったが、後半には「ビジネスモデルの変革(10月)」へ変化
- IT投資は守りから攻めへ転じつつあり、以降、「ビジネスモデルの変革」へのIT投資が増加傾向

（出典）日本情報システム・ユーザー協会(JUAS),「企業IT動向調査報告書2021,p.73 (2020/9-10月調査)」, アンケート対象は東証一部上場企業及びそれに準じる企業

先が見えない時代に不可欠の技法：アジャイル開発

■ 未来がわからない時は、Planで考えるのでなくDoで探る

- 新しいサービスを提供したい
  - ➢誰も解答を持っていない。繰り返し試すしかない
- 産業の壁を越えた新しいビジネスを考えたい
  - ➢知らない産業では想定外の発生がつきもの。実際に試してみないと通用するかどうか判断できない

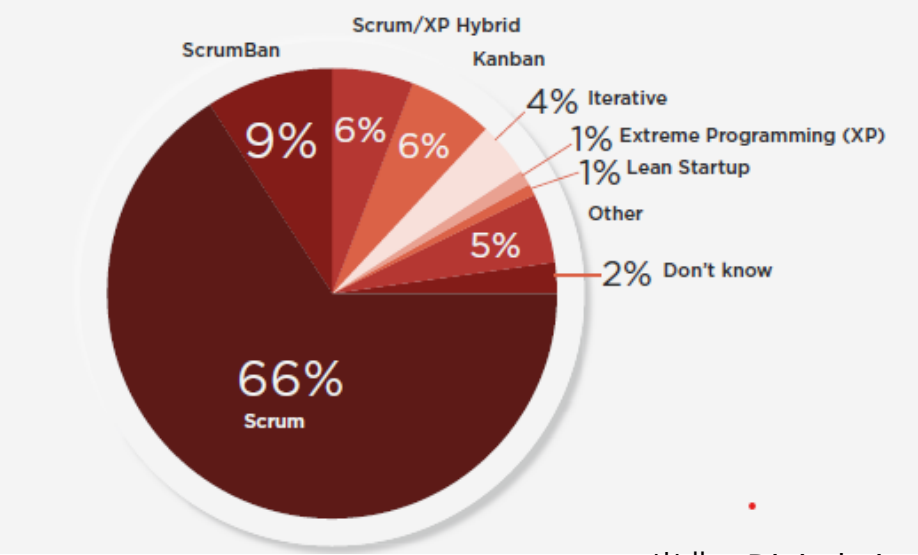
⋮

テスト&ラーンが可能なアジャイル開発が最適

アジャイル開発とは

- 「アジャイルソフトウェア開発宣言」および「アジャイルソフトウェアの12の原則」に則るソフトウェア開発技法

- 世の中でよく使われているアジャイル開発技法



66%がScrumを使用
（ScrumBan, Scrum/XP Hybridを含むと81%）

出典：Digital.ai、15th State of Agile report

アジャイルソフトウェア開発宣言

アジャイルソフトウェア開発宣言

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

この宣言は、この注意書きも含めた形で全文を含めることを条件に自由にコピーしてよい。
http://agilemanifesto.org/iso/ja/manifesto.html

アジャイルソフトウェアの12の原則

アジャイル宣言の背後にある原則

私たちは以下の原則に従う:
顧客満足を最優先し、価値のあるソフトウェアを早く継続的に提供します。

要求の変更はたとえ開発の後期であっても歓迎します。
変化を味方につけることによって、お客様の競争力を引き上げます。

動くソフトウェアを、2-3週間から2-3ヶ月というできるだけ短い時間間隔でリリースします。

ビジネス側の人と開発者は、プロジェクトを通して日々一緒に働かなければなりません。

意欲に満ちた人々を集めてプロジェクトを構成します。
環境と支援を与え仕事が無事終わるまで彼らを信頼します。

情報を伝えるもっとも効率的で効果的な方法はフェイス・トゥ・フェイスで話をすることです。

動くソフトウェアこそが進捗の最も重要な尺度です。

アジャイル・プロセスは持続可能な開発を促進します。
一定のペースを継続的に維持できるようにしなければなりません。

技術的卓越性と優れた設計に対する不断の注意が機敏さを高めます。

シンプルさ(ムダなく作れる量を最大限にすること)が本質です。

最良のアーキテクチャ・要求・設計は、自己組織的なチームから生み出されます。

チームがもっと効率を高めることができるかを定期的に振り返り、
それに基づいて自分たちのやり方を最適に調整します。

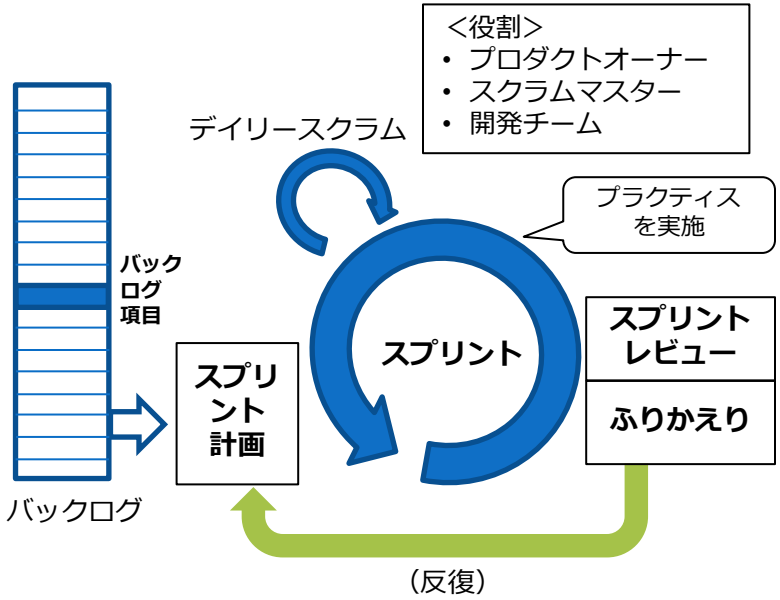
出典:http://agilemanifesto.org/iso/ja/principles.html

2. アジャイル開発の特徴

～ウォーターフォールモデル開発との比較から～

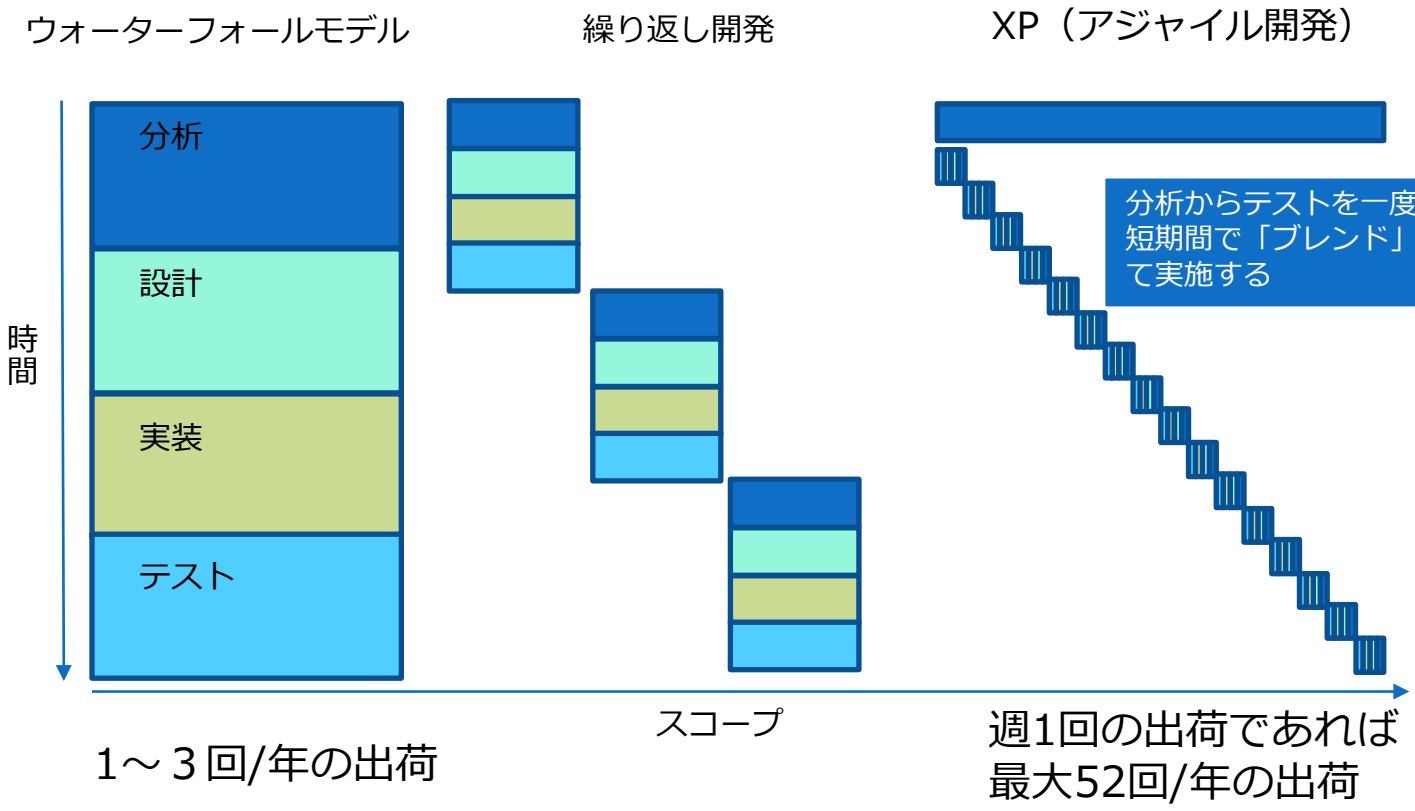
アジャイル開発の概要（スクラム用語での説明）

No	用語	説明
1	バックログ	要件一覧のこと。各要件は優先順位付けし、優先順位の高い順に並べる
2	バックログ項目	要件のこと
3	スプリント	１回の繰り返し開発のこと。スクラムでは、１回のスプリントの期間は１週間～4週間程度と言われている
4	スプリント計画	スプリントの開発計画を立案する場。スプリント開始時に実施する
5	デイリースクラム	開発チームが毎日実施する15分程度の短いミーティング
6	プラクティス	開発習慣
7	Doneの定義	バックログ項目が完了しているかどうかを判定するための基準
8	スプリントレビュー	スプリントの最後に実施する、動くソフトウェアが完成したかを確認する場
9	振り返り	スプリント終了時に実施する開発チームの反省会
10	プロダクトオーナー	開発するプロダクトの責任者
11	スクラムマスター	正しくスクラムを実践するよう支援する役割
12	開発チーム	要件に従って開発するチーム。3～9人の構成が望ましい



※アジャイル開発は、ツールをうまく適用できるよう、開発環境の整備が重要

開発プロセスの違い　～短いウォーターフォールモデルではない～



出典： K. Beck, Embracing Change with Extreme Programming. Computer, 32, 70-77, 1999 に基づき、筆者が追記

開発条件の違い（最も成功する確率の高い条件を比較）

カテゴリ	アジャイル開発	ウォーターフォールモデル開発
適用目的	迅速な価値、変化への対応	予測可能性、安定性、確実性
規模	小規模	大規模
適用環境	混沌、激しい変化	安定、少ない変化
マネジメント	個人間の暗黙知	文書化された形式知
顧客	専任のオンサイト顧客	必要に応じた顧客とのやり取り
要求	優先順位付けされた要件 予想できない変化の受容	正式に承認された要求 予測可能な要求の進化
開発	シンプルな設計 短いインクリメント	大規模な設計 比較的長いインクリメント
開発チーム	レベル１A～３で構成 （レベル１Bは不可、レベル2-3は30%）	レベル１B～３で構成 （レベル１Bは30%許容）

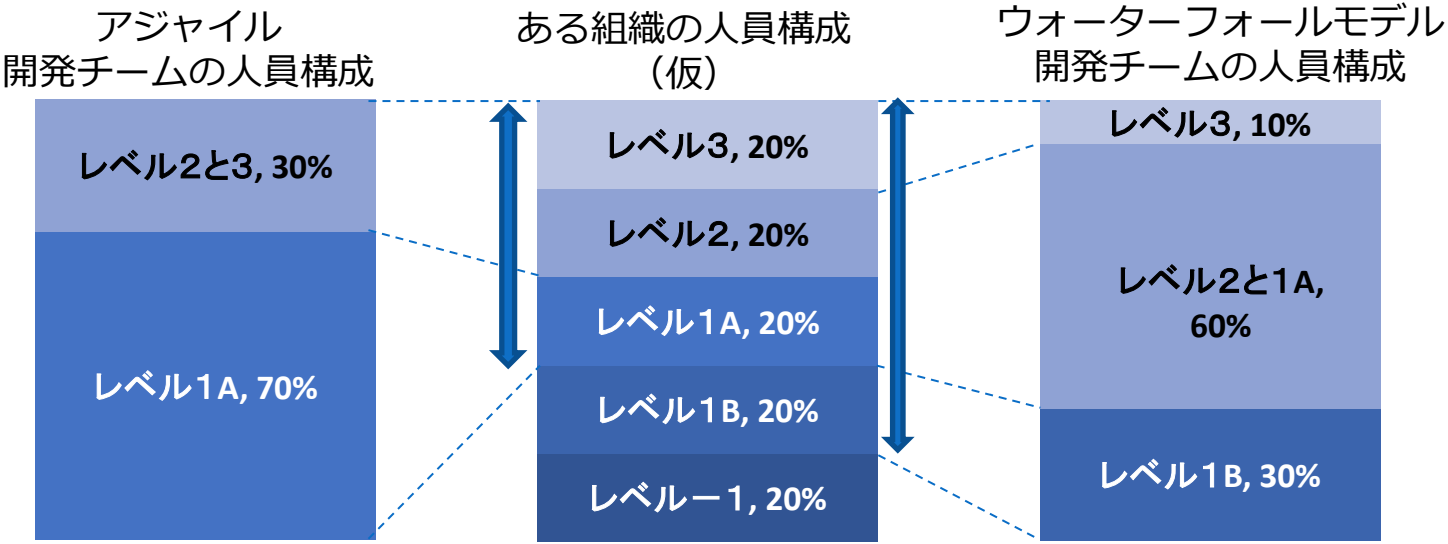
出典：B・ベーム/R・ターナー（著）「アジャイルと規律 ～ソフトウェア開発を成功させる２つの鍵のバランス～」、2004年、日経BP社

開発チーム編成の違い

開発者の技術レベル

レベル		手法の理解と利用の技術レベル
レベル3		適用する条件に適合するように、手法そのものを改訂できる
レベル2		適用する条件に合わせて、手法をカスタマイズできる
レベル1		手法を適用できる
	レベル1A	手法を使って、自由裁量の部分を遂行できる （経験を積めばレベル2になることができる）
	レベル1B	手法を使って、手続き的な部分を遂行できる （経験を積めば、レベル1Aのスキルのいくつかをマスターできる）
レベル−1		手法を使えない、または使わない

出典：アリスター・コックバーン（著）、長瀬嘉秀/永田渉(監訳)、「アジャイルソフトウェア開発」、2002年、ピアソン・エデュケーションおよび前ページの出典から筆者作成



3. アジャイル開発の実情

～アジャイル開発失敗事例を交えて～

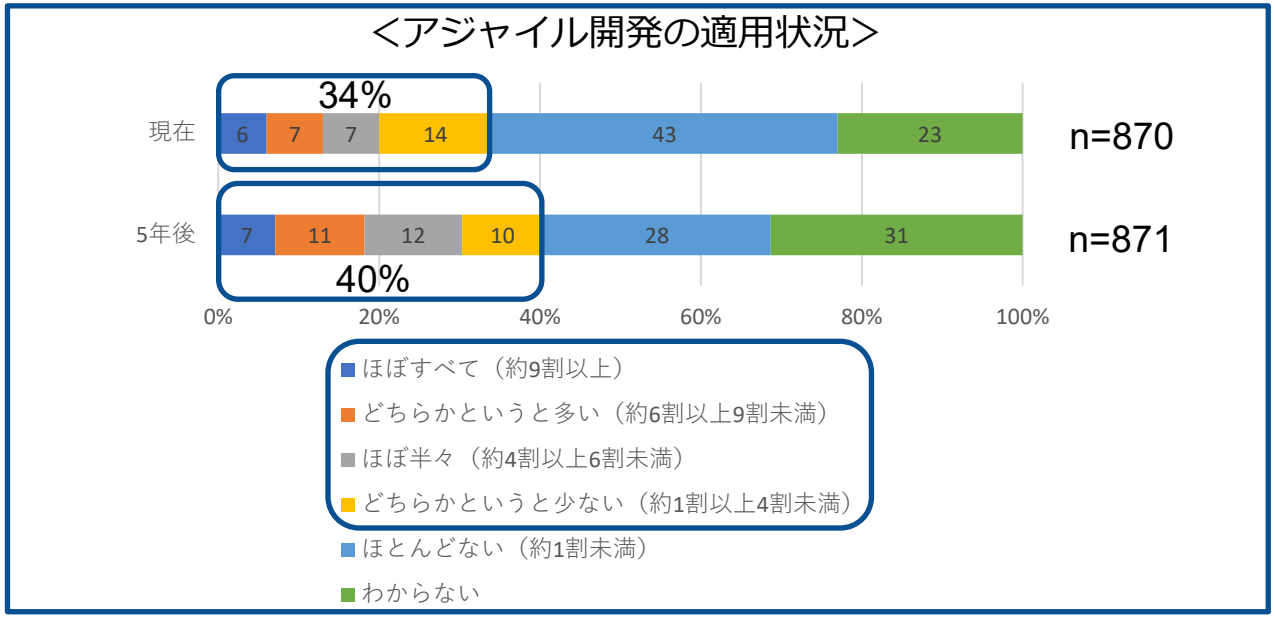
日本におけるアジャイル開発の状況

■ 日本のアジャイル開発は、まだ少ないが着実に増加

- 2021年時点の適用企業は34%⇒5年後は40%

（参考）2018年時点の適用企業は17%

（出典：ガートナー「ITデマンド・リサーチ」2018/5）



出典：「2021年度組込み/IoTに関する動向調査」、2022/5/10，IPA

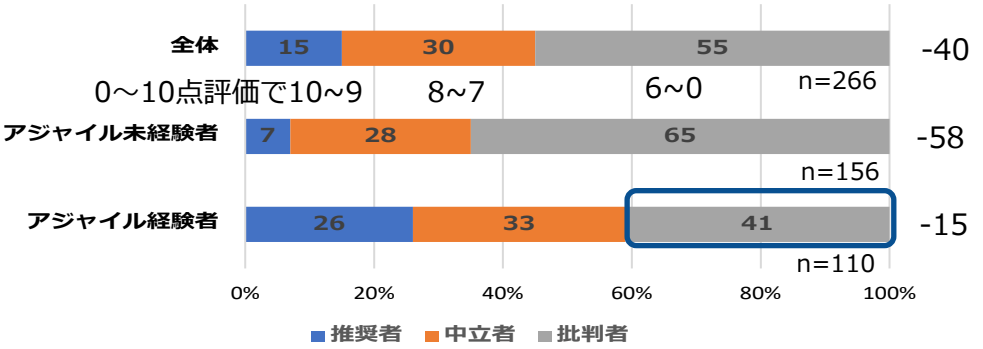
アジャイル経験者の意見：アジャイル実践は難しい

■ アジャイル経験者の41%が批判者（推奨者は26%）
- 顧客ロイヤルティ（NPS)は過去5年間で2番目に低い
  - ➢経験者比率は上がっているため、実践の結果と推測

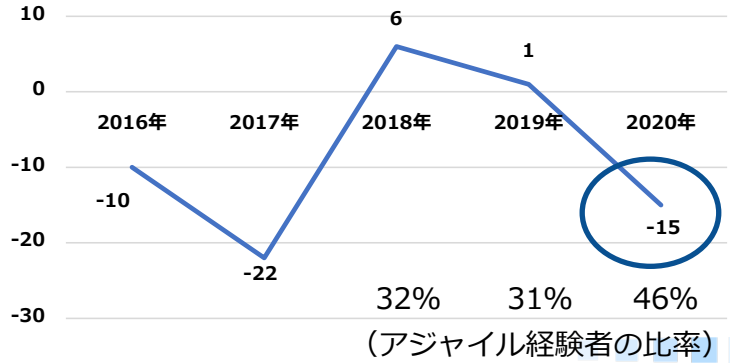
アジャイルの実践に課題

批判者の理由
- 担当のスキルに問題
- ステークホルダーに問題
- メンバーの負荷が高い
- 実現可能性は低い
- 内部統制の枠組みが崩壊
- 正しくないアジャイルが実践される　等

アジャイルを親しい友人や同僚に薦める可能性



アジャイル経験者のNPSの推移



（アジャイル経験者の比率）

※日本企業従業員に対するWebアンケート調査結果（2020/3）

※NPS＝推薦者の割合-批判者の割合

出典：2020年度 アジャイルプロジェクトマネジメント意識調査報告（p.42，58，14），PMI日本支部，https://www.pmi-japan.org/topics/PMI_Japan_Chapter_Agile_Survey_2020.pdf

アジャイル開発失敗事例　～初心者が陥る失敗～

No.	失敗事例	お薦めする解決方法
1	アジャイルっぽい開発 •プロトタイプ（試作品）を本物にしてしまう。正常系の主要機能は動くが、異常系や性能などは設計されておらず、すぐに不具合に遭遇する	•プロトタイピングと製品開発を分ける •異常系や性能などもバックログとして挙げる
2	アジャイルっぽい開発 •仕様書を書かない	•作成すべき仕様書を決め、仕様書を作成する
3	未熟な技術者の開発チームによる開発 •すぐにソフトウェアができたつもりになるが、実際は機能不足・低品質で負債の山になる	•開発チームを一定の技術レベルの技術者で構成する
4	バックログが大きすぎて、1回の開発スプリントで動くソフトウェアが出来上がらない •スプリント1回目（設計まで）⇒スプリント2回目（コーディング）⇒スプリント3回目（テスト）と、ウォーターフォールモデルと同じになってしまった	•バックログの大きさを1回のスプリントで作成できる大きさに区切る（区切れるかどうかはスキルに依存）
5	ある程度スプリントを繰り返したところで、デグレードが頻発し、開発やり直し	•テスト自動化し、毎回テストする

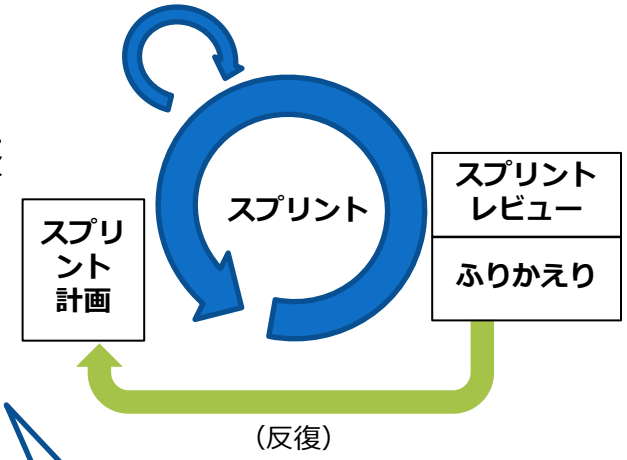
4. アジャイル開発の難しさとは

世の中のアジャイル開発への取り組み：軸足の異なる2つのアプローチ

■ アジャイルをうまく回す

- 例：スクラム研修
  1. 認定プロダクトオーナー
  2. 認定スクラムマスター
  3. 認定スクラムデベロッパー
  4. スクラムデベロッパーテクニカル研修

※1と2はマネジメント系に力点がある



世の中の視点は、「アジャイルをうまく回す」方法

その理由は、SW開発以外へのアジャイル適用が進んでいるため

■ アジャイル開発で高い品質・生産性を達成する

アジャイルをうまく回しても、SW開発の品質・生産性を確保できるわけではない（ソフトウェア工学からの取組み必須）

アジャイル開発とは（本講演の立ち位置）

■ アジャイル開発とは

「所定の品質を確保したソフトウェアを、短期間で繰り返しリリースする手法」

■ アジャイル開発が解決しようとする課題

「要件がなかなか決まらない、決まったとしてもすぐに変更する」

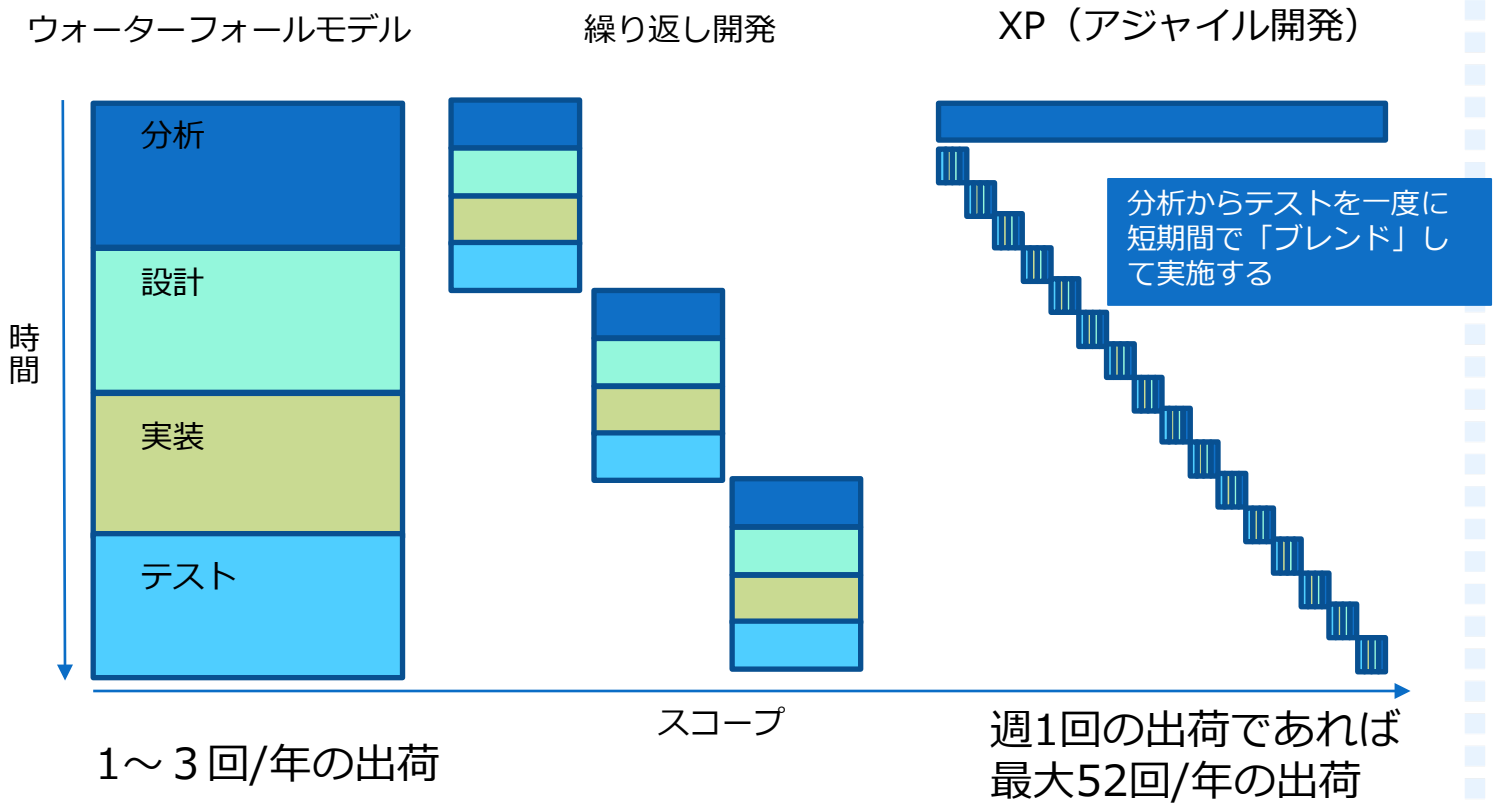
- 従来のウォーターフォールモデル開発は、開発初期に要件が確定することが前提
- アジャイル開発は、要件は変化するものとして、変化する要件へ対応できるよう考案された開発技法
  - ➢バックログは、常に最新状態に見直し、優先順位付けしていることが前提
  - ➢ただし、開発中の要件は、そのスプリント内では変更しない

※「ゼロから始めないアジャイル開発」をお薦め

- アジャイル開発宣言から20年以上の業界経験を取り入れる
- 従来からのソフトウェア工学の知見を活かす

「短期間」を攻略するには自動化・効率化が必須

（短いウォーターフォールモデル開発ではない）



出典： K. Beck, Embracing Change with Extreme Programming. Computer, 32, 70-77, 1999 に基づき、筆者が追記

問題解決法として「自己組織化」を重視（しすぎ）

■ 自己組織化した開発チームは、常に改善が回る

■ しかし…

- 自己組織化するまでに時間がかかる
  - ➢多重下請構造に慣れた技術者が、自律した技術者に変わるまで、時間がかかる（1~2年か）
    - ✓経営陣は、それまで待てるだろうか？

主要なメトリクスが相対値（客観性を担保しにくい）

■ アジャイル開発の主要なメトリクスは、開発チーム内でしか通用しない相対値

開発チーム内では効果的に使える
開発チーム外との比較や基準値の横展開ができない

■ アジャイル開発でよく使用されるメトリクス

➢ストーリーポイント

その要件を実現するのに必要な作業量の相対値

- 工数（人時）の見積もりより、簡単で有効
- 工数（人時）の見積もりのように正確性を要求されない

➢ベロシティ（速度）

その要件を実現するのに必要な作業量の相対値

- ベロシティを継続的に追跡することにより、その開発チームの開発できる量（開発できる速度）がわかる

5. アジャイル開発の品質確保のポイント

～鍵は技術力と定量化～

品質重視のアジャイル開発は、全体的な仕組みで実現する

初めから品質を作り込む

<開発チーム編成>

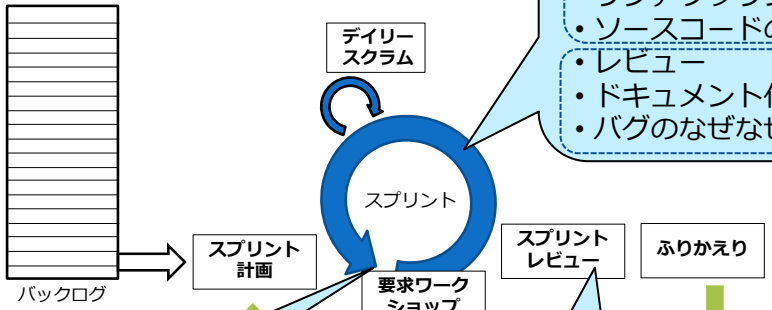
立場	要件
技術リーダー	自ソフトウェア開発組織の上位20%に入る技術者。 コミュニケーション力があるほうが望ましい
開発チーム員	自ソフトウェア開発組織の上位80%に入る技術者

技術系プラクティスをリードできる技術リーダーを設定

アジャイル開発宣言から20年の知見に基づく

<技術系プラクティス>
- ペアワーク
- テスト自動化と継続的インテグレーション
- リファクタリング
- ソースコードの共同所有
- レビュー
- ドキュメント化
- バグのなぜなぜ分析と水平展開

日本の品質レベルを支えてきた技術に基づく



ウォーターフォールモデル開発
他のアジャイル開発チーム

※顧客の納得性を高める

<役割>
- プロダクトオーナー
- スクラムマスター
- 開発チーム
- 品質技術者

品質技術者の追加を推奨

<推奨するメトリクス>
- Done判定後のバグ数
- テスト項目数
- 工数

客観的メトリクスによる比較

Doneの定義⇒判定

品質が作り込まれていることを常に確認する

<プロセス品質> 基準値達成を確認しながら開発を進める

品質ダッシュボード

自動計測・自動判定

<プロダクト品質> 実際に良いものができていることを確認する

成果物の評価

設計仕様書
テスト仕様書
動くソフトウェア

※品質技術者による評価を推奨

推奨プラクティス ～3種類のプラクティスを意識する～

No	マネジメント系プラクティス
1	スプリント計画
2	デイリースクラム
3	要求ワークショップ
4	スプリントレビュー
5	振り返り（KPT)
6	タスクボード
7	バーンダウンチャート

No.1～5はスクラムによるものである

アジャイル開発宣言から20年の知見より

No	技術系プラクティス
1	ペアワーク
2	テスト自動化と継続的インテグレーション
3	リファクタリング
4	ソースコードの共同所有
5	レビュー*1
6	ドキュメント化*2
7	バグのなぜなぜ分析と水平展開*3

アジャイル開発宣言から20年の知見より

日本の品質レベルを支えてきた技術より（アジャイル開発で検証済）

No.1～4はXPによるものである（No.1はXPではペアプログラミングと表記）

*1ソフトウェア工学で説明されるフェイストゥーフェイスのレビューである

*2仕様書作成を明示するためのプラクティスである

*3バグはプロセス課題を示し、水平展開による残存バグ摘出が必須である

品質を決めるのは、技術系プラクティス

※テスト系プラクティスとバグにこだわる日本の技術は、テストエンジニアの領域

（参考）日本の品質レベルを支えてきた技術を利用する

<パフォーマンスデータの比較>

	インド	日本	米国	欧州他	全体
プロジェクト数	24	27	31	22	104
新規コード行数/人月*	209	469	270	436	374
出荷後12ヶ月バグ数/KL*	0.263	0.020	0.400	0.225	0.150
参考：100KLのソフトウェアの出荷後12カ月のバグ数	26件	2件	40件	23件	15件

*：メジアン（中央値）を表示

日本は米国の1/20

出典： M. Cusumano, “Software development worldwide: the state of the practice”, IEEE Software, 2003（最後の行は筆者が追記）

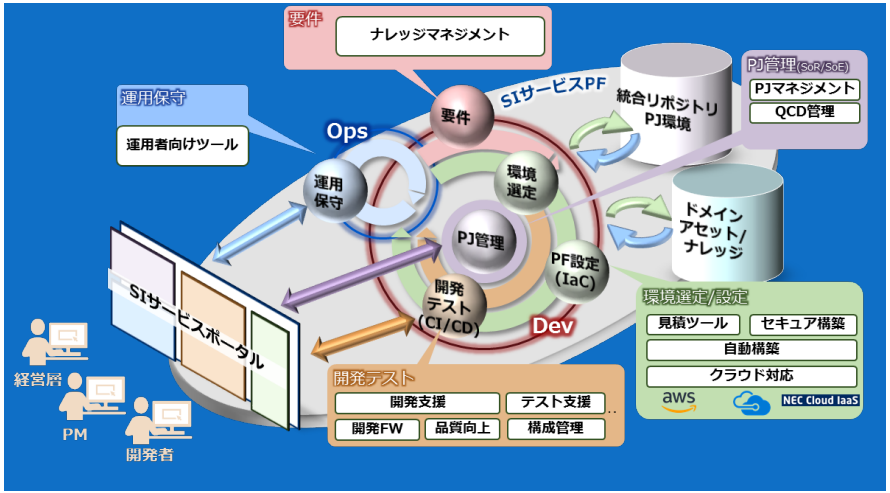
■ 日本の品質レベルを支えてきた技術とは

- バグにこだわる考え方と取り組み
  - ➢レビュー
  - ➢バグのなぜなぜ分析と水平展開
  - ➢品質ゲート　　など

開発環境整備による自動化・効率化

マネジメント系ツール

No	ツールの種類	説明
1	タスクボード	タスクをTODO(未着手)/DOING(着手中)/DONE(完了)の3段階の状態で管理
2	バーンダウンチャート	スプリントの作業進捗をグラフ化で可視化したもの。タスクボードと連動して、バーンダウンチャートを自動的に作成可能
3	バグ管理	バグや問題点の管理。チケット管理ツールで、タスクボード、バーンダウンチャート、バグ管理を一つのツールで実現することもある
4	KPT	振り返りの手法であるKPTをツール化



技術系ツール

No	ツールの種類	説明
1	構成管理	成果物（ソフトウェア、テストプログラム、ドキュメント類等）の構成管理
2	テスト自動化	xUnitなどのユニットテストフレームワーク SeleniumなどのUI自動操作ツール Jenkinsなどのテスト自動化ツール　など
3	コードメトリクスの計測	主なコードメトリクスは、ソースコード行数、プログラムのネストの段数、コメント行数、サイクロマチック数などの複雑度、条件分岐数　など
4	コード静的解析	ソフトウェアを実行せずにソースコードを解析して問題点を指摘する。利用言語に合わせてツールを選択する
5	テストカバレッジ計測	カバレッジの計測方法には、C0（命令網羅）,C1（分岐網羅）,C2（条件網羅）が知られている。
6	OSSライセンス違反検出	OSSの著作権者が定めたOSSライセンスの違反検出。Black Duckが有名
7	セキュリティ脆弱性検出	設計・コーディング・テストの各場面でセキュリティを意識して開発する必要がある。Webアプリケーション漸弱性検査ツール、ネットワーク脆弱性検査ツール、などがある

利用を推奨するツール

ツールを集約して提供する
クラウド型開発環境（例）

推奨する開発チーム編成　～一定の技術力を確保～

立場	要件
技術リーダー	自ソフトウェア開発組織の上位20%に入る技術者。 コミュニケーション力があるほうが望ましい
開発チーム員	自ソフトウェア開発組織の上位80%に入る技術者

■ 優秀な技術リーダーの設定
- 技術系プラクティスの適切な実施は、技術リーダーの実力に大きく左右される

■ 開発チーム員は一定レベルをアサイン
- 技術レベル１A以上を推奨
  - ➢１A:適用する技法を使って、
  - ➢自由裁量部分を遂行できる

開発者の技術レベル

レベル		手法の理解と利用の技術レベル
レベル3		適用する条件に適合するように、手法そのものを改訂できる
レベル2		適用する条件に合わせて、手法をカスタマイズできる
レベル1		手法を適用できる
	レベル1A	手法を使って、自由裁量の部分を遂行できる （経験を積めばレベル2になることができる）
	レベル1B	手法を使って、手続き的な部分を遂行できる （経験を積めば、レベル1Aのスキルのいくつかをマスターできる）
レベル-1		手法を使えない、または使わない

出典：アリスター・コックバーン（著）、長瀬嘉秀/永田渉(監訳)、「アジャイルソフトウェア開発」、2002年、ピアソン・エデュケーション
およびB・ベーム/R・ターナー（著）「アジャイルと規律 ～ソフトウェア開発を成功させる２つの鍵のバランス～」、2004年、日経BP社
から筆者作成

Doneの定義へ、出荷OKの基準をあらかじめ明確化する

対象	カテゴリ	No	審査観点	審査基準
バックログ項目	①ツールによる確認	1	静的解析	ソースコード静的解析による高以上の指摘に対する修正　100%
		2	コードメトリクス	実行行数200行以下　100%
		3		ネスト4以下　100%
		4	カバレッジ	単体テストカバレッジ　100% ※何らかの理由により実行していないコードは、レビューにより妥当性を確認済である
		5	セキュリティ	セキュリティ脆弱性検査による指摘に対する修正　100%
		6	OSS	OSSライセンス違反検査による指摘に対する修正　100%
	②テスト結果	7	テスト実施	新規テスト実施率　100% ※非機能テストなど、機能以外のテストを含む
		8		リグレッションテスト実施率　100%
		9	テスト自動化	単体テスト自動化率　100% ※他のテストもできるだけ自動化する
		10	実行結果の記録	テストの実行結果の記録　100%
	③成果物の評価	11	設計仕様書	設計仕様書の成果物評価による指摘に対する修正　100%
		12	テスト仕様書	テスト仕様書の成果物評価による指摘に対する修正　100%
		13	最終成果物	ソフトウェア評価による指摘に対する修正　100% ※ソフトウェア評価対象には、マニュアル、インストーラーなど顧客に提供するものすべてが含まれる
	④成果物の登録	14	設計仕様書	設計仕様書一式の構成管理への登録　100% ※技術調査メモなどの必要情報を含む
		15	ソースコード	ソースコード一式の構成管理への登録　100%
		16	テスト仕様書	テスト仕様書一式の構成管理への登録　100%
		17		テストコード一式の構成管理への登録　100%
スプリント	⑤バグ対応	18	未解決バグ	未解決バグ　0件
		19	水平展開	バグ分析と水平展開実施率　100% ※対象：当該スプリントで摘出された全バグ
	⑥アクションアイテム対応	20	アクションアイテム対応	残存アクションアイテム　0件 ※対象：当該スプリントで対応予定のアクションアイテム

プロセス品質

プロダクト品質

Done判定後のバグが対象

ツールで確認できるものは**自動計測 自動判定**

実施すべきテストを確実に実施し、テスト自動化していることを確認

品質技術者が、実際に仕様書の中身を見たり、SWを動かす

実際に登録された成果物を確認

残存バグがないことを確実にする

バグが発生すると、ここが最も重い項目となる

AIは、バックログに追加し、確実に対応できるようにする

定量化：比較を可能にするメトリクスの収集

<手動計測を推奨するメトリクス>

カテゴリ	単位	データ項目	説明
ストーリーポイント	ストーリーポイント（SP）	ベロシティ	今回のスプリントで完了したバックログ項目のストーリーポイントの合計値
		未完了ストーリーポイント	今回のスプリントで完了しなかったバックログ項目のストーリーポイントの合計値
Done判定後のバグ	件数	Done判定後のバグ	Done判定を受けた後に摘出したバグ
		うち新規バグ	Done判定を受けた機能から摘出されたバグ
		うちデグレードバグ	ある機能の開発の影響を受けて既存機能が動作しなくなったバグ
		うち既存バグ	既存機能から摘出したバグ
工数	人H	工数	開発チームが、アジャイル開発に費やした工数
		うちペアワーク	ペアワークに費やした工数
		うちレビュー	レビューに費やした工数。レビュー記録票の合計工数を基本とする
テスト項目数	項目	新規テスト項目	今回のスプリントで開発した機能に対するテスト項目
		リグレッションテスト項目	既存機能に対するテスト項目

※テスト項目数は、自動実行と手動実行が混在する場合が多いため、本一覧へ入れている。

※アジャイル開発で収集するメトリクスは、上記以外にDoneの定義で自動収集する項目①~⑥がある

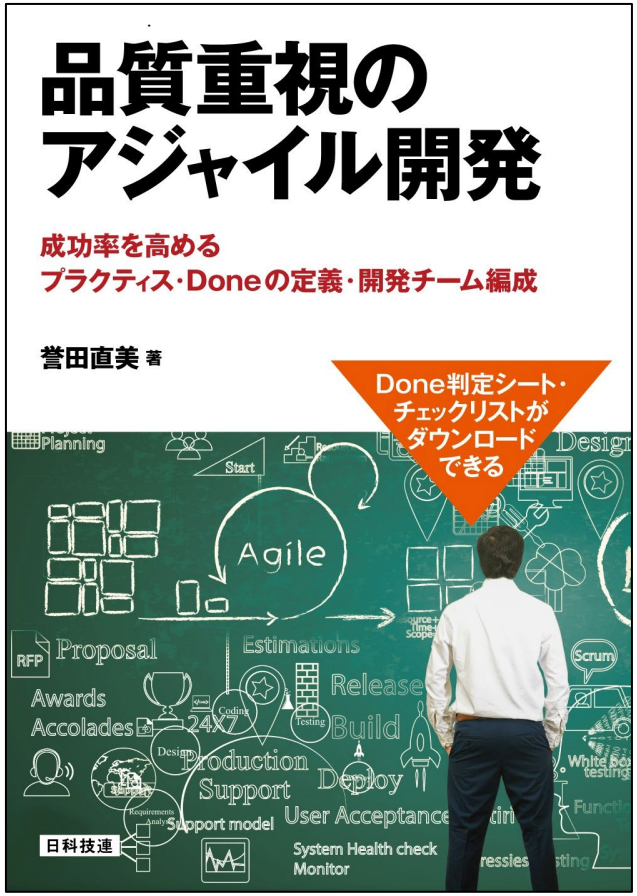
6.まとめ

まとめ：品質重視のアジャイル開発のポイント

1. 品質重視のアジャイル開発は、全体的な仕組みで実現
   - 自己組織化だけに頼らず、開発チーム編成を重視
2. ソフトウェア開発の品質・生産性確保には、技術力と定量化への取り組みが必須
   - 技術系プラクティスを確実に実施
   - 自動化・効率化を支える開発環境の整備
   - 一定レベルの開発チーム編成
   - 合格基準は、あらかじめDoneの定義へ入れて客観的に評価
   - 比較を可能にするメトリクスを収集評価

誰でも開発できる時代は終わった。鍵は、技術力と定量化！

書籍のご紹介　「品質重視のアジャイル開発」



品質重視のアジャイル開発
～成功率を高めるプラクティス・Doneの定義・開発チーム編成～

誉田　直美[著]
2020年9月26日発行
日科技連出版社　定価 2700円（税抜き）
ISBN-10：481719717X　**ISBN-13：**978-4817197177

[目次]

https://ideson-worx.com