

7.アジャイル開発講座 概要

ねらい	従来のウォーターフォール型開発経験者がアジャイル開発や DevOps の新世界に移行できる様に基本的な考え方（思想）を含めた正しい理解を得る知識研修であり、パラダイムシフトを起こせる気付きを作る。				
開催日程	12 時間（e-learning 2 時間含む）				
受講条件	IT 技術者としての経験が 3 年以上、ICT の基礎知識を持っていること				
学習目標	・講義とワークショップを組み合わせて自身で考える場を提供することにより、アジャイル思考を身につける。				
	・最新のソフトウェア開発 & 運用環境の動向を理解し、実務でアドバイスできる。				
	時間	講義	演習	学習概要	学習詳細
カリキュラム概要	1:10	0:40	0:30	アジャイル開発の概要	<ul style="list-style-type: none"> ・アジャイル開発の特徴 ・アジャイル開発に必要なスキル ・アジャイル宣言と 1 2 のアジャイル原則 ・アジャイルの価値観 ・パラダイム・シフト ・《WS》ソフトウェアを作る意味、ユーザーの要望（グループ・ディスカッション）
	0:20	0:10	0:00	スクラム	・スクラムの価値観（働き方）
	1:00	0:20	1:00	スクラムのプロセスと役割	<ul style="list-style-type: none"> ・スクラムのプロセス ・プロジェクト管理 ・スクラムの役割（登場人物） ・《WS》ユーザーストーリー（演習）
	0:30	0:20	0:00	XP 概説	<ul style="list-style-type: none"> ・XP の価値感 ・手法の紹介
	2:00	0:30	1:30	アジャイルな計画	・アジャイル開発で用いる計画手法紹介

					・2レベル・プランニング（プランニング・オニオン）
					・《WS》タスク分解（演習）
	1:30	0:30	1:00	品質管理	・アジャイル開発での品質
					・《WS》振り返り（KPT）（演習）
	1:40	0:40	1:00	大規模プロジェクト	・スクラム・オブ・スクラムの紹介
					・他の大規模プロジェクトの手法紹介
					DAD, SAFe
	1:30	0:30	1:00	まとめ	・《WS》アジャイルをどう活用するか？（グループ・ディスカッション）
	合計時間	10:00	3:40	6:20	

時間	学習項目	学習項目の狙い	詳細内容
1:10	第1章 アジャイル開発の 概要	目的：アジャイル開発の基礎 について学ぶ ゴール：アジャイル開発の大ま かな流れと根本にある考え方を 理解する	第1章【講義】
			(1)アジャイル開発の特徴
			・アジャイル開発がどのような特徴を持った開発手法なのか理解を 促す。反復、一人多役、スコープで調整、常時リリース、適応型 開発プロセスという用語の意味を解説しながらアジャイル開発の流 れを説明する。
			(2)アジャイル開発に必要なスキル
			(3)アジャイル宣言とは
			・アジャイル開発の様々な手法を考えた第一人者たちが集まり、ア ジャイル開発を定義したものであると説明する。
			(4)12の原則
			・原則の各文章を紹介する。
			【演習】アジャイルの価値
			(1)12のアジャイル原則を理解し、『アジャイル開発に移行する ために、現状から変えなければならない事』を上げてください。
			(2)チームで30分間討論して、重要度（優先順位）の高い順 に表記してください。
			(3)発表してください。
			【講義】
			アジャイルの価値観
			(1)タイムボックス
			・開発における各作業を時間分けして管理すること、短時間で 集中して行動することを重要視することを説明する。 ・タイムボックスを超えた場合には必ず超えた理由を振り返り改善 するも説明する。
			(2)チーム作業
			・優先順位の設定、進捗管理をマネージャーに一任するのではな く、チーム内で共有しながら進めることを説明する。
(3)ユーザー第一の考え方			
・短い期間でリリースを繰り返すため、ユーザーは一部が出来上が るたび動くソフトウェアを確認できることを説明する。			

			<p>・ユーザー自身の自発的な参加を促す（プル型プロセス）についても説明する。</p> <p>(4)一人多役</p> <p>・分析、設計、実装、テストに担当を専任するのではなく、全てをこなせるエンジニアが必要であると説明する。</p> <p>(5)チームの自己組織化</p> <p>・チームを自律した自己組織化することで外部からの阻害要因を排除すること、チーム内で同じ目標を共有し開発へのモチベーション向上に寄与することを説明する。</p> <p>パラダイム・シフト</p> <p>(1)正解主義から適応主義へ変換</p> <p>・アジャイル開発は正解を導いてからその正解に向かうのではなく、常に変化が状況に応じて変化がありえることを受け入れる考え方が重要だと説明する。また、より良い成果のために変化し適応していくのが適応主義であると説明する。</p> <p>第1章クイズ</p> <p>(1)アジャイル開発でのドキュメント作成について適切なものを選びなさい。</p> <p>(2)アジャイルソフトウェア開発の原則について適切なものを選びなさい。</p>
0:20	<p>第2章 スクラム (軽量なプロジェクトの管理)</p>	<p>目的：スクラムの基本構造、進め方と手法の役割について学ぶ</p> <p>ゴール：スクラムの有用な点を理解しパラダイム・シフトについて考える</p>	<p>第2章【講義】</p> <p>(1)スクラムの価値観（働き方）</p> <p>(2)アジャイル開発の仕事の基本構造</p> <p>(3)要求の変質</p>
1:00	<p>第3章 スクラムのプロセスと役割</p>		<p>第3章【講義】</p> <p>(1)スクラムの特徴</p> <p>・軽量であることや反復して行うリリース、チームが自己裁量権を持つ等、プロセスを説明する上で前提となる情報を説明する。</p> <p>(2)スクラムの各工程</p> <p>・図を確認しながら、各工程についてそれぞれの役割とつながりを説明する。</p> <p>(3)プロダクト・バックログ</p>

		<p>・開発要求のリストでありプロジェクトには 1 つだけ。プロダクト・オーナーが優先順位の責任者であることや、従来の要件定義と違い動的であることなどの特徴を説明する。</p>
		(4)ユーザーストーリー
		<p>・ユーザーからの要求を具体化するための手法の 1 つであると説明。どんな特徴がありどんな方法で進めていくのか、具体的な手法を提示しながら解説する。第 3 章の講義終了後、演習としてユーザーストーリーを作成する。</p>
		(5)スプリント・プランニング
		<p>・開発チームとプロダクト・オーナー間の会議であり、どのような内容を話し合うのかや、プロダクトへの追加方法を解説する。</p>
		(6)タスクへの分割（タスクの定義）
		<p>・スプリント目標達成へのタスクリスト。</p>
		<p>・プロダクト・バックログの要求リストごとに、実装に必要な作業を詳細に分割したものをタスクということを説明する。</p>
		(7)スプリントレビュー
		<p>・スプリントで構築したプロダクトを関係者に提示すること。開催の注意点を説明する。</p>
		(8)レトロスペクティブ（振り返り）
		<p>・評価とプロセスの改善が目的。開催ルールを説明する。kpt 法が用いられることが多い。</p>
		(9)プロジェクト管理
		<p>・見える化が原則。コミュニケーションも重要視されている。チーム規約の例も提示して説明する。</p>
		(10)スタンドアップ・ミーティング
		<p>・短い時間で行われるため、簡潔にポイントを絞ることが重要と説明する。</p>
		(11)DoD（完了基準）
		<p>・品質管理の基本であり、チーム内で明確にしておく。</p>
		(12)バーンダウン・チャートとベロシティ
		<p>・チームが進む速度を表し、1 回のスプリントで開発できる量と説明する。</p>
		(13)コミットメント
		<p>・スプリントの終わりに、働くプログラムを提供することでコミットメントすることを説明する。</p>

			スクラムの役割（登場人物）
			(1)プロダクト・オーナー
			・役割と責任について説明する。優先順位の最終決定等の責任を負う。
			(2)スクラム・マスター
			・役割と責任について説明する。チーム支援が主な役割。
			(3)開発チーム
			・役割と責任について説明する。チーム全体として責任を持って開発を行う。
			(4)スクラムの環境
			・写真を交え、オープンでコミュニケーションの取りやすい環境について説明する。
			第3章クイズ
【演習】ユーザーストーリーの作成			
0:30	第4章 XP概説	目的：XPの手法や進め方を学ぶ ゴール：アジャイル開発の手法により違いについて理解を深める	第4章【講義】
			概要
			(1)XPの概要説明
			(2)価値観を構成する要素について説明する。
			手法の紹介
			(1)テスト駆動開発（TDD）
			・TDDの流れを図とともに解説。
			(2)リファクタリング
			・出来上がったソースコードに対して作業レベル、使用レベルに応じて調整を行っていく。注意点についても説明する。リファクタリング基本方針についても説明する。
			(3)ペア・プログラミング
・ペアを高頻度で変更することによる緊張感の保持や、1人で考え込む時間を減らすことが目的と説明する。			
(5)テストの自動化			
・テストの自動化につながるのは、イテレーションでテスト・カバレッジを上げていくことであり、その方法論について解説する。			
(4)10分間ビルド			
・自動的にシステム全体をビルドしていくという手法で時間を短縮し、全てのテストを10分間で終わらせると説明する。時間が限られているので、無意味なテストを避ける工夫につながる。			

			第 4 章クイズ
			(1)リファクタリングについて適切なものを選びなさい。
			(2)ペアプログラミングの説明で適切なものを選びなさい。
			・プロダクト・バックログの要求リストごとに、実装に必要な作業を詳細に分割したもののことをタスクということを説明する。
			(1)プロダクト・バックログからユーザーストーリーを作成してください。
2:00	第 5 章 アジャイルな計画	目的：アジャイル計画の種類・詳細について学ぶ ゴール：与えられた課題からタスクの分解や計画の立案ができるようになる	第 5 章[講義]
			アジャイル開発で用いる計画手法紹介
			(1)反復性増加型開発の計画
			(2)リリース計画とスプリント計画
			リリース計画の中で複数のスプリント計画を繰り返す。見直しが連続的に行われるため、早めの見通しを立てることができると説明する。
			(3)全体計画の立案
			5つの工程に分けて立案すること、各工程を説明する。
			(4)直近のリリース計画の立案
			8つの工程に分けて立案する過程を説明する。
			反復にユーザーストーリーを配置していく過程について説明する。1回の反復にストーリーが収まりきらなければ分割もでき、分割のガイドラインについても説明する。
			(5)ストーリーの分割
			ストーリーの分割のガイドラインについて、どんな要素によって分割するのか説明する。
			タスクの定義に付いて解説する。スクラムプロセスの中ではタスクがどこに位置するのかについても図で確認する。
			(6)スプリント計画の立案
			スプリント・ミーティングの流れを説明し、ストーリーとタスクがどこに関連してくるのか確認する。タスクへの分割（タスクの定義）についても説明する。タスクの見積もりには個々人の認識の違いを反映させないよう、プランニング・ポーカーを用いる方法も紹介する。
			(7)タスクの分割と粒度
			スプリント・バックログと、プランニング・ポーカーによるタスクの見積もりについて説明する。
粒度は小さいほうがいいということを前提とし、小さくすることによる利点について、例を交えて説明する。			

			<p>第 5 章クイズ</p> <p>(1)アジャイル開発プロジェクトにおける計画作りについて適切で無いものを選びなさい。</p> <p>(2)ストーリーポイントを付与する際に注意すべき点で適切なものを選びなさい。</p> <p>[演習]</p> <p>5. タスクへの分解</p> <p>(1)『スクラムのプランニング・セッションを実行する (3～4時間)』と言うプロダクト・バックログをタスクに分解する。</p>
1:30	<p>第 6 章</p> <p>品質管理</p>	<p>目的：従来の方法と品質がどのように変わるのかを学ぶ</p> <p>ゴール：振り返りの重要性を理解し、KPT 手法を使用できるようにする</p>	<p>第 6 章[講義]</p> <p>品質管理</p> <p>(1)品質について考える</p> <p>当たり前の品質と魅力的な品質に分けてどんなものが該当するのか説明する。</p> <p>(2)アジャイル開発で品質が向上する理由</p> <p>理由として 5 つの要因を挙げて、各要因について例を交えて説明する。</p> <p>(3)もの作りとシステム作りの相違</p> <p>高品質なシステム作りにおいて、もの作りと対比しながら、相違点について説明し、どうしたら高品質なシステムが完成するか考えさせる。</p> <p>(4)KPT での振り返り (レトロスペクティブ)</p> <p>KPT 法の概要と手法について説明する。</p> <p>(参考) TQM とは</p> <p>総合的品質管理のことであり、高品質なシステムを提供するための一連の活動のこととして紹介する。</p> <p>(5)ソフトウェア製造工程におけるムダの廃除</p> <p>7 つのムダの例を挙げながら廃除の方法を説明する。また、仕事を分析する観点から 7 つのムダがどのように発生するのか説明する。</p> <p>第 6 章クイズ</p> <p>・アジャイル開発における品質の確保について適切では無いものを選びなさい。</p> <p>[演習]</p> <p>6. KPT (振り返り)</p>

			(1)課題を1つ選び、6ヶ月後のあるべき姿を定義し、初めの3ヶ月間の行動計画を立案する。
1:40	第7章 大規模プロジェクト	目的：大規模プロジェクトで 用されるアジャイル開発につ いて具体例から学ぶ ゴール：小規模な開発との手 法の違いについて理解する	第7章[講義]
			(1)スクラム・オブ・スクラムの紹介
			(2)FBI センティネル・プロジェクト
			スクラム・オブ・スクラムは大規模なプロジェクトで複数のスクラムチ ームを構成することと説明し、具体的なスクラム事例を用いて解説 する。
			(3)KANBAN
			大規模プロジェクトにおける KANBAN の有用さを、例を交えなが ら説明。またスクラムとの類似点、相違点についても解説する。
			(4)KANBAN の利用価値
			スクラムとどちらを採用するのか、あるいは組み合わせて活用するの か、利用方法について説明する。
1:30	まとめ	目的：本講座で学んだことを 演習を通して整理する ゴール：受講生なりのアジャイ ル活用方法を模索する	アジャイルをどう活用するか？ (1)グループ・ディスカッション (2)発表

アジャイル開発

目次

第1章 アジャイル開発の概要	4
第2章 スクラム	14
第3章 スクラムのプロセスと役割	20
第4章 XP概説	55
第5章 アジャイルな計画	66
第6章 品質管理	89
第7章 大規模プロジェクト	100

第1章 アジャイル開発の概要

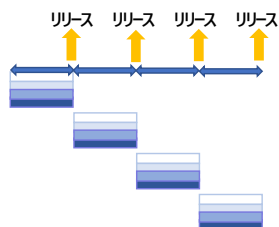
- アジャイル開発の概要
- アジャイル宣言とアジャイル原則
- アジャイルの価値観
- パラダイム・シフト

• 説明の流れ

アジャイル開発がどのような特徴を持った開発手法なのか理解を促す。反復、一人多役、スコープで調整、常時リリース、適応型開発プロセスという用語の意味を解説しながらアジャイル開発の流れを説明する。

アジャイル開発の概要（特徴）1

◆ 反復型（繰り返し）の開発



◆ 一人多役（多能）



・ 説明の流れ

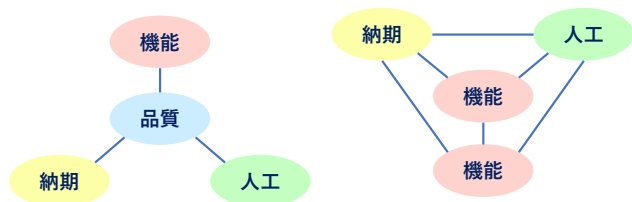
アジャイル開発がどのような特徴を持った開発手法なのか理解を促す。反復、一人多役、スコープで調整、常時リリース、適応型開発プロセスという用語の意味を解説しながらアジャイル開発の流れを説明する。

・ 補足

従来から親しんでいるウォーターフォール型開発と比較してアジャイル開発の特徴をまず理解させて下さい。

アジャイル開発の概要（特徴） 2

◆ スコープで調整



◆ 常時リリース

◆ 適応型開発プロセス

・ 説明の流れ

アジャイル開発がどのような特徴を持った開発手法なのか理解を促す。反復、一人多役、スコープで調整、常時リリース、適応型開発プロセスという用語の意味を解説しながらアジャイル開発の流れを説明する。

・ 補足

従来から親しんでいるウォーターフォール型開発と比較してアジャイル開発の特徴をまず理解させて下さい。

アジャイル開発の概要（必要なスキル）

1. 分析設計スキル

ユーザーストーリー / タスク分解と見積り

2. 開発スキル

テスト駆動開発 / リファクタリング / ペア・プログラミング (XP)

3. テストスキル

継続的インテグレーション / 10分間ビルド (XP)

4. プロジェクト管理スキル

透明化 / タイムボックス

5. 品質管理スキル

DoD (完了基準)

6. チームビルディング・スキル

協調 / コミットメント / 合意 / カイゼン / リーダーシップ / チーム規約

・説明の流れ

順を追って説明する。

アジャイル開発を実践するうえで必要となる、知識&スキルについて羅列しました。これからこのテキストでは、これらの紹介をします。

アジャイル開発概論

アジャイル開発とは、高品質で要求に柔軟なソフトウェア作成を行うため、協調性、自律性を重視した反復的な開発方法である。

- ダイナミックシステムズ開発技法
Dynamic Systems Development Method (Dane Faulkner ほか)
- アダプティブソフトウェア開発
Adaptive Software Development (Jim Highsmith)
- クリスタルメソッド
Crystal Methods (Alistair Cockburn)
- スクラム
Scrum (Ken Schwaber, Jeff Sutherland)
- XP / エクストリームプログラミング
(Kent Beck, Eric Gamma ほか)
- リーンソフトウェア開発
Lean Software Development (Tom and Mary Poppendieck)
- フィーチャ駆動開発
Feature-Driven Development (Peter Code, Jeff DeLuca)
- アジャイル統一プロセス
Agile Unified Process (Scott Ambler)

- 反復 (周期) 的 (Iterative)
--- 定期的なリリース
- 漸進的 (Incremental)
--- 徐々に機能を増加
- 適応主義 (Adaptive)
--- 変化に対応 (即応)
- 自律的 (Self-Organized)
--- 学習する組織
- 多能工 (Cell Production)
--- 一人多役
(SE、プログラマー、テスター)

• 説明の流れ

アジャイル開発の各種方法を順を追って説明して下さい

アジャイルソフトウェア開発宣言

2001年2月 (於 米国ユタ州スノーバード) 制定

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する活動を通じて、より良い開発方法を見つけ出そうとしている。

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、
計画に従う事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置く。

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

<http://agilemanifesto.org/>

・説明の流れ

宣言の意義や目的を説明する。

アジャイル開発の様々な手法を考えた第一人者たちが集まり、アジャイル開発を定義したものであると説明する。

・補足

この宣言は、次ページの原則と合わせて自由にコピーできますが、この注意書きを含めて、全文を掲載しなければなりません。

アジャイルソフトウェア開発の原則（1）

私たちは、以下の原則に従う。

1. 顧客満足を最優先し、
価値あるソフトウェアを早く継続的に提供します。
2. 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。
3. 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。
4. 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。
5. 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。
6. 情報を伝える最も効率的で効果的な方法は
フェイス・ツー・フェイスで話をする事です。

・説明の流れ
原則の各文章を紹介する。

・補足
アジャイルソフトウェア開発宣言には、更にこのような原則が付加されています。この原則はソフトウェア開発で何を重視すべきかが揺われています。これはソフトウェアの価値を最大化するための具体的な行動指針です。

アジャイルソフトウェア開発の原則（２）

7. 動くソフトウェアこそが進捗の最も重要な尺度です。
8. 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。
9. 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。
10. シンプルさが本質です。
11. 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。
12. チームが最も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。

<https://agilemanifesto.org/iso/ja/principles.html>

・説明の流れ

原則の各文章を紹介する。

・補足

アジャイルソフトウェア開発宣言には、更にこのような原則が付加されています。この原則はソフトウェア開発で何を重視すべきかが揺われています。これはソフトウェアの価値を最大化するための具体的な行動指針です。

アジャイルの価値観

アジャイル開発と他のソフトウェア開発手法との違いを整理します。

アジャイル開発

- タイムボックス制
- 完全なチーム作業
- ユーザー第一主義
- 一人多役
- チームの自己組織化

他の開発手法

- (← 長期間)
- (← PMの存在)
- (← 分業制)

アジャイル開発においては、パラダイム・シフトが必須！

正解主義から適応主義への変遷！

説明の流れ

順を追って説明する。

(1)タイムボックス

- ・開発における各作業を時間分けして管理することと、短期間で集中して行動することを重要視することを説明する。
- ・タイムボックスを超えた場合には必ず超えた理由を振り返り改善するも説明する

(2)チーム作業

- ・優先順位の設定、進捗管理をマネージャーに一任するのではなく、チーム内で共有しながら進めることを説明する。

(3)ユーザー第一の考え方

- ・短い期間でリリースを繰り返すため、ユーザーは一部が出来上がるたび動くソフトウェアを確認できることを説明する。
- ・ユーザー自身の自発的な参加を促す（プル型プロセス）についても説明する。

(4)一人多役

- ・分析、設計、実装、テストに担当を専任するのではなく、全てをこなせるエンジニアが必要であると説明する。

(5)チームの自己組織化

- ・チームを自律した自己組織化することで外部からの阻害要因を排除すること、チーム内で同じ目標を共有し開発へのモチベーション向上に寄与することを説明する。

パラダイム・シフト

(1)正解主義から適応主義へ変換

- ・アジャイル開発は正解を導いてからその正解に向かうのではなく、常に変化が状

況に応じて変化がありえることを受け入れる考え方が重要だと説明する。また、より良い成果のために変化し適応していくのが適応主義であると説明する。

第1章クイズ (1)

(問) アジャイル開発でのドキュメント作成について適切なものを選びなさい。

1. 設計根拠となったドキュメントは、プログラムの動作確認後は不要である。
2. 保守に必要なDB設計書は、リリース後の最新状態のものをツールを自動生成により用意した。
3. それぞれのプログラマーの個性を活かすため、コーディングルールなどは作成しない。
4. 保守・運用に必要な情報は全て口頭で伝え、ドキュメントは作成しない。

・ 発問の流れ

正解の番号と正解の理由を説明してください。

他の3つの不正解の理由を簡潔に説明して下さい。

正解は、 2

第1章クイズ (2)

(問) アジャイルソフトウェア開発の原則について適切なものを選びなさい。

1. チーム内での対話によりリスク回避が可能なので、できる限り長く会議の時間を取る。
2. 効率は良いが正確性に欠ける対話ではなく、ドキュメントベースの情報伝達を推進するのが良い。
3. 最も効率的なコミュニケーションの方法は、フェイス・トゥ・フェイスの対話であり、積極的な活用が求められる。
4. 顧客との仕様調整に有効なコミュニケーション手段の対話のため、対話のスペシャリストを選任で用意するべきである。

・ 発問の流れ

正解の番号と正解の理由を説明してください。

他の3つの不正解の理由を簡潔に説明して下さい。

正解は、 3

第2章 スクラム

- スクラムの価値観（働き方）

スクラムの基本構造、進め方と手法の役割について学ぶ

スクラムの有用な点を理解しパラダイム・シフトについて考える

スクラムの価値観（働き方）

スクラムの三つの価値



プラス



・説明の進め方

それぞれの言葉の説明を、具体例を交えて説明する

・補足

(1)監視

・ユーザーが作成物や進捗を頻繁に確認し、変化を認識できる点について説明する。

(2)透明性

・開発に関わる者の認識が共通していることが透明性につながると説明する。

(3)調整

・開発工程における不備の拡大を防ぐ観点から、調整の重要性について説明する。

(4)タイムボックス

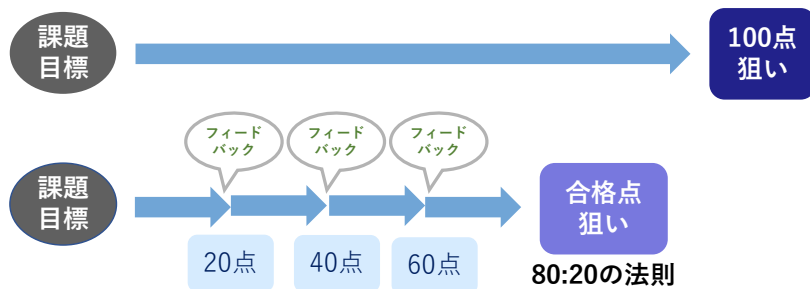
・時間を区切ることで短期間で集中して効果を上げるという考え方を説明する。

アジャイル開発の仕事の基本構造

イテレーション (スプリント)

- 時間の使い方
- 透明性
- 品質

反復的に開発を行った方が
良いのか？
どうして、そうしなければ
いけないのか？



• 説明の進め方

イテレーションの意味の理解をさせる。
ウォーターフォールとの違いで説明する。

• 補足

(5)イテレーション

• 反復して行う工程をイテレーションと呼ぶことと、反復する意義について説明する。

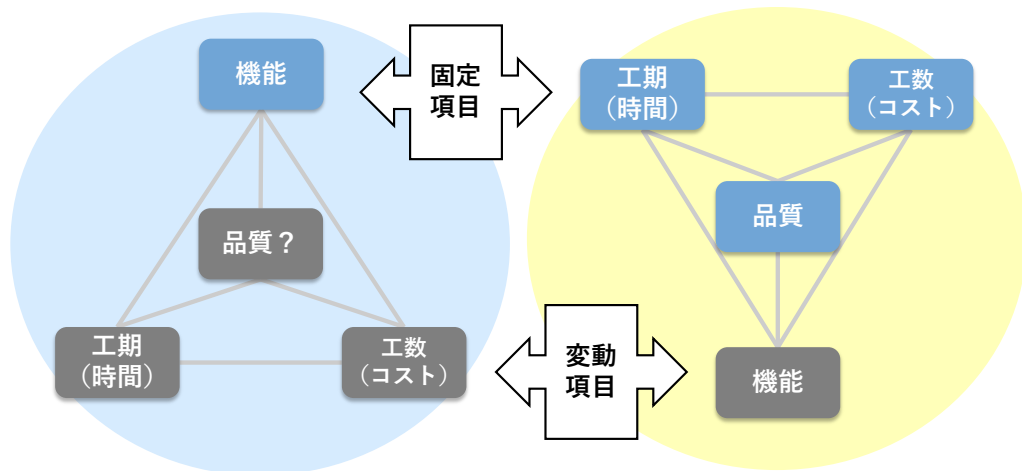
(6)従来の開発アプローチ

• 制作物が持つ機能は決定されており、それを実装するために必要な工数、工期は変動する可能性を持っていることを説明する。

アジャイルの価値観

従来の開発アプローチ

DSDM, Atern(アターン)の開発アプローチ



・説明の進め方

・補足

アジャイル開発の基本的な考え方を紹介します。従来の考え方、アプローチからは大きなパラダイムシフトになります。

従来は、まず開発すべき機能（要求）が大前提で、これだけの機能を実装（開発）する為には、これだけの期間（工期）と何人の体制（コスト）が掛かりますというプロジェクト計画の作成方法です。

アジャイルでは、逆の発想で、現在与えられているリソース、すなわち与えられた時間（工期）と予算（コスト）の範囲内で実現できる機能（要求）はどれだけか？というプロジェクト計画になります。

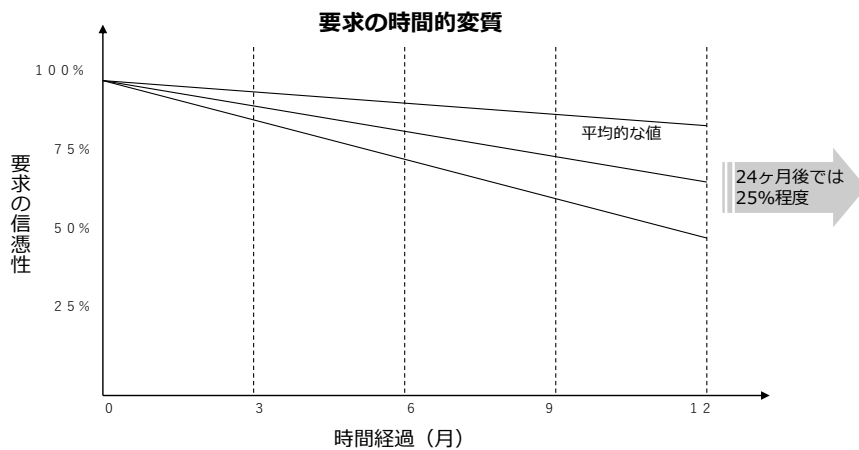
したがって従来の方法では、決められた機能を100%実装するために、工期、工数が不足してくると、品質を犠牲にするアプローチではないでしょうか？

アジャイルでは、工期、工数、品質を満たす事が前提で、実現される機能で調整を行う事になります。

要求の変質

迷信：要求を明確に定義できれば良い

現実：要求は時間と共に変化



- ・ポイント

仮に要求仕様をしっかりと定義できてもその要求自体が変質しています。

- ・補足

開発初期における要求定義

従来の、開発初期に明確な要求定義をすれば、良いシステムができあがるという考え方は、データから見ても迷信である。要求の信憑性は時間の経過とともに下がるため、初期の要求を変更せず開発を進めることは品質の向上に繋がりにくいことを説明する。

演習-1 アジャイルの価値 (パラダイムシフト)

12のアジャイル原則を理解し、

『アジャイル開発に移行するために、現状から変えなければならない事』
を上げてください。

ワーク 1

チームで30分間討論して、重要度(優先順位)の高い順に表記してください。

ワーク 2

チームの結論を5分間で発表してください。

第3章 スクラムのプロセスと役割

- スクラムのプロセス
- プロジェクト管理
- スクラムの役割（登場人物）
- 《WS》 ユーザストーリー（演習）

スクラムのプロセス

スクラム (Scrum) の特徴

軽量なアジャイルプロジェクト管理メソッド

小規模で自律的なチーム

完全な透明性

変更への敏速な適応

- 小さな多数のチーム同士の密接に協力と透明性の高い環境で、反復的にインクリメンタルな製品を公開する
- チームは自己決定権を持ち、それぞれの反復の目的を満たす計画を行う
- スクラムマスターがファシリテーションを行う
- 活動はプロダクトバックログによって組織化される

・説明の流れ

順を追って説明する。

軽量であることや反復して行うリリース、チームが自己裁量権を持つ等、プロセスを説明する上で前提となる情報を説明する。

スクラム (Scrum) の基本

- **Boundary / 境界**
 - スプリントやミーティングは、時間的制約を持つ = Timebox
開発チームのリズムを作る
- **Goal / 共有された目標**
 - 透明化による目標の共有
- **Fixed Team / 固定されたチーム**
 - チームメンバーは専任かつ固定化
少人数で、各々がさまざまな仕事を行う

- 説明の流れ
順を追って説明する。

スクラム・プロセス

プロダクトバックログ

- 1.-----
- 2.-----
- 3.-----
- 4.-----
- 5.-----



プロダクトオーナー
(お客様)

確実な優先順位付け
(同位は無し)
追加、修正、変更可能
(スプリントに入っている物以外)
プロダクトオーナーが
全て責任を持つ

スプリントプランニング

Part1	1時間
Part2	2時間



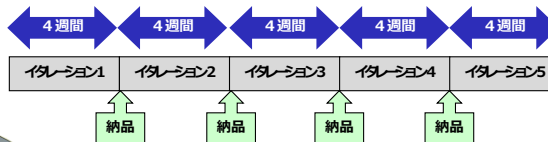
スクラムマスター
(ファシリテーター)

毎朝15分のスタンド
アップ・ミーティング

- ### Sprint Task List (Sprint Backlog)
- a.-----
 - b.-----
 - c.-----
 - d.-----
 - e.-----
 - f.-----

タスクボード

To Do	On Going	Done
c	b	a
d		
e	g	f
h		

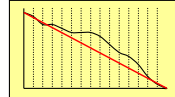


開発チーム

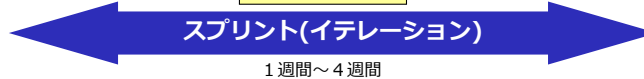
デザイン
開発
テスト

デイリースクラム
(スタンドアップ・ミーティング)

バーンダウン・チャート



スプリントレビュー
1時間
レトロスペクティブ
(振り返り)
1時間



・説明の流れ

図を確認しながら、各工程についてそれぞれの役割とつながりを説明する。

スクラム (Scrum) のプロセス

1. プロダクト・バックログ

要求を優先度の順にまとめる

2. スプリント計画

- ① プロダクト・バックログを元にスプリントの目標設定をする
- ② 開発チームのタスクを決定する

3. スプリント・バックログ

スプリント計画で決めたタスクをリスト化する

4. スプリント

チームの作業期間

5. デイリー・スクラム

毎朝の進捗報告ミーティングで具体的な作業報告を行う

6. スプリント・レビュー

スプリントの成果を報告する

7. スプリント・レトロスペクティブ

毎週行われる進捗報告ミーティングにて、KPTをチーム間で共有する
(KPT … Keep/継続 Problem/問題 Try/挑戦)

・説明の流れ

順を追って説明する。

プロダクト・バックログ

・開発要求のリストでありプロジェクトには1つだけ。プロダクト・オーナーが優先順位の

であることや、従来の要件定義と違い動的であることなどの特徴を説明する。

ユーザーストーリー

・ユーザーからの要求を具体化するための手法の1つであると説明。どんな特徴がありどんな方法で進めていくのか、具体的な手法を提示しながら解説する。第3章の講義終了後演習としてユーザーストーリーを作成する。

スプリント・プランニング・開発チームとプロダクト・オーナー間の会議であり、どのような内容を話し合うのかや、プロダクトへの追加方法を解説する。

タスクへの分割 (タスクの定義)

・スプリント目標達成へのタスクリスト。
・プロダクト・バックログの要求リストごとに、実装に必要な作業を詳細に分割したもののことをタスクということを説明する。

スプリントレビュー

・スプリントで構築したプロダクトを関係者に提示すること。開催の注意点を説明する。

レトロスペクティブ (振り返り)

- ・評価とプロセスの改善が目的。開催ルールを説明する。kpt法が用いられることが多い。

プロダクト・バックログ

- 要求を直感的にわかりやすくリスト化
- プロジェクトにつき1つのみ
- 優先順位に並べる
 - ユーザーにとっての価値やリスクを考慮
 - 実装の及第点も示す
- プロダクト・オーナーが内容や優先順位の保守と支持の責任を負う
- 要求の変動や成果のにより、常時更新され内容が変化する
- 誰でも提案できる、全員から常に見えるところに提示される

・説明の流れ

要求のリストでありプロジェクトには1つだけ。プロダクト・オーナー

が優先順位であることや、従来の要件定義と違い動的であること

などの特徴を説明する。

・補足

開発しなければならない要求のリストは、ユーザーの視点でユーザーが理解できる表現形式をとるようにします。

優先順位では、プロダクト・オーナーが優先順位の責任者です。

優先順位付けではユーザーのビジネス価値やビジネスリスク、システム構築上のリスクを考慮して順位を決めます。

優先順位付けが完了した後は、システム構築価値のパレート図を作成しておくのも、何処まで実装できれば構築するシステムとして及第点が取れるかを明確にするために有

効です。

プロダクトバックログは動的であるために最初は不完全だが、ユーザーが自分のニーズに対する理解を深めていくにつれて明確になっていきます。

ユーザーストーリー

要求が適切に理解されずに、そしてプロジェクトのステークホルダーとコミュニケーションが取れない限り、どのようなソフトウェア/システム開発も成功に至ることはあり得ません。

- Karl E. Wiegers

ユーザーストーリーは要求を具体化する方法の一つ

ユースケースやシナリオと共に**システムの利用法に基づいたアプローチ**（ユーザー視点のシステムの振る舞い）によりユーザーの要求を理解し記述するための技法

簡潔な文書による記述、さらに詳しい情報を得る**会話**、そしてこれらの詳細を確認するためのテスト（受入条件）から構成される

・説明の流れ

ユーザーストーリー

ユーザーからの要求を具体化するための手法の1つであると説明。どんな特徴がありどんな方法で進めていくのか、具体的な手法を提示しながら解説する。第3章の講義終了後、演習としてユーザーストーリーを作成する。

ユーザーストーリー (ビジネス分析のツール)

ユーザーストーリーにより、

工程や作業の意味を明確にする

作業や効果を明快に説明できるようになる

開発者とユーザーの会話のきっかけを作る

開発手法に依存せず、専門知識をも不要である

→ ビジネス分析・改善のツールである

「役割・利用者 (As a role)」

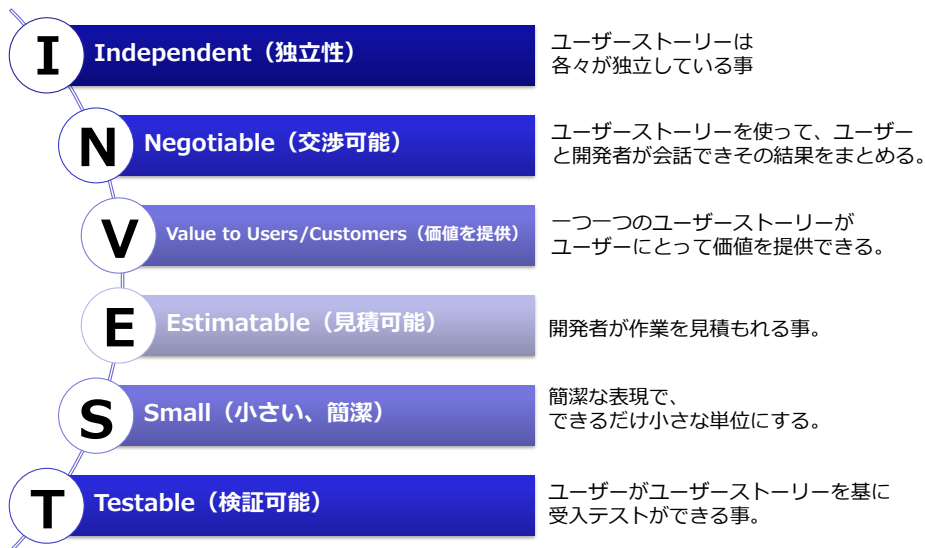
「機能 (can) / 要求 (need)」

「ビジネス価値 / 効果 (In order to)」

の3つに分けて表現される

- ・ 説明の流れ
順を追って説明する。

簡単なルール INVEST



・説明の流れ
順を追って説明する。

・補足

独立性

ユーザーストーリーは各々が独立している事。

交渉可能

ユーザーストーリーを使って、ユーザーと開発者が会話できその結果をまとめる。

価値を提供

一つ一つのユーザーストーリーがユーザーにとって価値を提供できる

見積もり可能

開発者が作業を見積もれる事。

小さい、簡潔

簡潔な表現で、できるだけ小さな単位にする。

検証可能

ユーザーがユーザーストーリーを基に受入テストができる事。

ユーザーストーリーの表記法（書き方）

<利用者の役割>として、
私は、**<機能や性能（システムが〜できる事）>**が必要だ。
それは、**<理由（ビジネス価値）>**の為だ。

(例)

<役割> 予約受付係りとして、
<機能> 私は年間最繁忙期でも最低12件の旅行予約を60分以内に処理できる。
<価値> それは予約電話の不受信を最小限に収めるためだ。

・説明の流れ

ユーザーストーリーの表記方法の例を挙げておきます。

予約受付係り<役割>として、
私は年間最繁忙期でも最低12件の旅行予約を60分以内に処理<機能>できる。
それは予約電話の不受信を最小限に収める<価値>ためだ。

ユーザーストーリー・カード (例)

User Story Card ID: ← 表面

要求者名:
部門名:

役割 (As a Roll):
機能 (I/We can):

環境・条件 (Which I need):

効果・価値 (To):

日付: 0000/00/00 記入者:

裏面 →

<input type="checkbox"/> 確定要求		
<input type="checkbox"/> 未確定要求【仮説・推測・期待・その他】		
<input type="checkbox"/> Body Process	<input type="checkbox"/> Altanative	<input type="checkbox"/> Option
【コメント欄】		

・説明の流れ

ユーザーストーリー作成に使用するカードの一例を示します。

スプリント・プランニング

パート 1

プロダクト・オーナーと開発チームにより次のスプリントのどのような機能を構築するか（実装内容）を決定する。

手順

1. プロダクト・バックログの提示
2. 次のスプリントで対象とするバックログの選定
3. スプリント目標の設定（バックログの実装を通じて満たされるべき目標）
4. ストーリーの確認

・説明の流れ

開発チームとプロダクト・オーナー間の会議であり、どのような内容を話し合うのかや、プロダクトへの追加方法を解説する。

・補足

パート1

- ①プロダクト・バックログの提示
- ②次のスプリントで対象とするバックログの選定
- ③スプリント目標を設定
 - プロダクトバックログの実装を通して満たされるべき目標
- ④ストーリーの確認

スプリント・プランニング

パート 2

開発チームのプロダクトへの実装方法の決定

手順

1. ストーリーの作成
2. リリース内容と予定の決定・報告
3. スプリント・バックログ項目（タスク）の定義
4. タスクの作業時間見積り

・説明の流れ

パート2

- ①ストーリーの作成
- ②リリース内容と予定
 - プロダクト・オーナーへ報告
- ③スプリント・バックログ項目（タスク）の定義
- ④タスクの作業時間見積り

タスクへの分割と定義

スプリント・バックログ (タスク)

スプリント目標達成のためのタスクのリスト
タスクは、要求の実装に必要な作業を細分化したものの
作業時間数によりタスクの見積もる
各メンバーが担当するタスクを選択する (割り当てではない)
スプリント中、チームのメンバーにより更新される
スプリント・バックログを変更できるのはチームだけ

・説明の流れ

タスクへの分割 (タスクの定義)

スプリント目標達成へのタスクリスト。
プロダクト・バックログの要求リストごとに、実装に必要な作業を詳細に分割したもののことをタスクということの説明する。

・補足

新しい作業が必要になったらチームはその作業をスプリントバックログに追加する

不必要に思えるタスクがあったら取り除く

スプリント・レビュー

スプリントでの成果を、管理者、顧客、ユーザ、プロダクトオーナーへ提示
管理者、顧客、ユーザ、プロダクトオーナーは、成果を査定

特徴

- 次のスプリント計画のため、全員がプロダクトへの追加要素を理解する
チームは製品について具体的に説明
質疑応答や議論を行うが、批判などは避ける
- ミーティング準備に時間はかけず、プロダクトを用いて端的に機能を紹介
一般的なプレゼンテーションツールは使わずに、機能するソフトを発表する。

・説明の流れ

スプリントで構築したプロダクトを関係者に提示すること。
開催の注意点を説明する。

次のスプリント計画ミーティングのために全員がプロダクトへの機能追加を理解することが重要です。チームは、作ったプロダクトの内容は何かという情報を提供します。出席者は質問、意見、議論、提案など積極的に行います。ただし、批判や対処の場ではありません。また、チームはミーティングの準備に時間をかけ過ぎてはいけません。PowerPointなどのプレゼンテーションツールも使用せず、ここまでの成果物を発表します。

レトロスペクティブ（振り返り）

毎週行われる進捗報告ミーティング

チームからKPTの共有

Keep / 継続 Problem / 問題 Try / 挑戦

特徴

- チームによる自主運営で、管理職は参加しない
- 透明性の為、幅広く自由な会話を行う（仕事での問題からチームの環境まで）
- 会議後チーム・リーダーが管理者へ報告、要望、依頼を行う
（管理者は間接的に要望解決を図る）

・説明の流れ

レトロスペクティブ（振り返り）

評価とプロセスの改善が目的。開催ルールを説明する。kpt法が用いられることが多い。

補足

- 毎週の振り返りは、管理職は参加せず、チームが自主運営する事が大事
- チームの中で自由に何でも話し合える環境、雰囲気作りが重要
この環境が構築されないとチーム内で透明性を確保できない
- 振り返りの話題、議論は、仕事に直接的に関係する話題だけで無く、チームの置かれている環境など間接的、補助的な事柄や、環境条件面についての話題も取り上げる
- 話題に上がった問題、課題は、チーム全員で解決、改善して行く事になるが、チームが解決できないものは要望事項として後日チーム・リーダーから管理者等へ依頼する
- 管理者は振り返りの会議の場に参加してはいけない
後日、チームリーダーから報告や要望を聞く
注意すべき事は、指示、命令的な発言を差し控える、コメントは直接的な表現を避けてチームに気付きを与えられる言葉に止める、褒める言葉を多用する事
基本的にはチームの要望事項を聞きその解決を図る

プロジェクト管理

自主管理を行う = 外部から状況がわからない

→ チーム内外でのコミュニケーションにより透明化を行う

チームが行うべき事

- チームの規約の作成
- 共有をしやすい環境作り
- リソースの管理
- スキルの管理
- コミットメント

・説明の流れ

見える化が原則。コミュニケーションも重要視されている。チーム規

約の例も提示して説明する。

・補足

プロジェクトの管理はチームが自主管理をします。

そのためには、各自の進捗状況や発生した問題点などをしっかり説明できること、すなわち見える化が必要です。お互いの状況を共有できていないと、プロジェクト全体を管理することはできません。

チームの規約例

- スプリントスケジュール
スプリント計画ミーティング
振り返りの設定
- 一日のスケジュール
スタンドアップ・ミーティング
タスクを取る最終時間
インテグレーションの時間
残業
- コードの基準 / リファクタリング
- ペア・プログラミングの実施ルール
- テスト駆動開発 (TDD)
- タスク終了の基準 (定義)

・補足

「標準」とは自らつくり出すもの

「標準」とは生産現場の人間がつくり上げるべきものである。けっして上からのお仕着せであってはならない。

標準（スタンダード）の設定には慎重な態度が必要である。なぜなら、標準はともすると正しいものより間違っただけを設定することになりがちだからである。

標準化には「惰性」を表すものと「進歩」を表すものがある。したがって標準化について漠然と議論することは危険である。

上からの標準は標準のためにもものをつくることで満足するようになり、人間の能力も鋭くなるどころか、かえって鈍くなるからである。

ヘンリー・フォード一世の考え方のなかに、標準とは上から与えられるものではないという強い信念が感じとれる。企業トップであっても、工場長であっても、どのような上司であっても、要するに「標準」を設定する人間は、企業においては現場の当事者がせよ、そうでないと「進歩」のための標準にはなりえないことを強調している。

大野耐一氏はトヨタ生産方式の著書の中で上記のように説いている。

アジャイル開発における作業規約（作業標準）の策定も、同様に現場が自ら行う事が重要である。

スタンドアップ・ミーティング

- 毎日チーム全員で行う短時間（15分程度）の状況確認ミーティング
- 継続と透明化を重要視する
- 「前日の作業結果」「当日の作業予定」「課題」を全員が同じ順に報告
- 必要に応じてフォローアップ・ミーティングを行う

朝礼ではない！

・説明の流れ

短い時間で行われるため、簡潔にポイントを絞ることが重要と説明

する。

・補足

スタンドアップ・ミーティングに参加する人は全員が等しく順番にこの発表を行います。管理職の方が参加される場合も例外ではありません。スタンドアップ・ミーティングはチーム全員（管理職もチームの構成員です）の情報共有、見える化の重要なコミュニケーションで

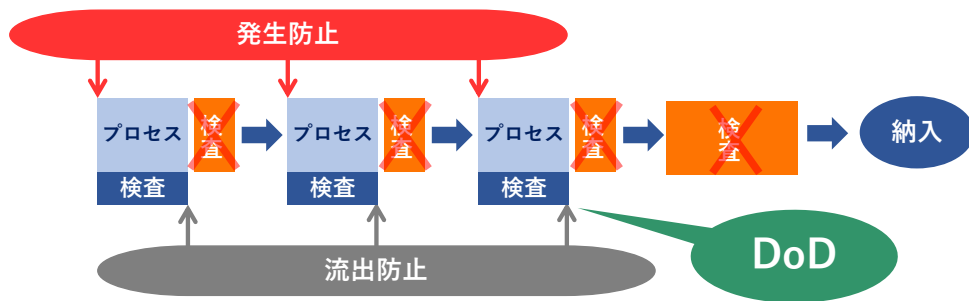
す。

DoD (完了基準)

品質管理の基本

不良をつくらない！
不良な作業をしない！

- 各作業の完了基準
- 閾値の決定
- 作業経過、結果を計測
- 自工程完結の基本的な姿勢
- 他人の作業を肯定的に捉える



• 説明の流れ

品質管理の基本であり、チーム内で明確にしておく。

• 補足

DoDとは、各作業の完了基準です。どこまで作業が進んだら完了とみなせるのかという基準、すなわち閾値（標準値）を決めます。

品質管理には以下の点に注意します。

作業経過、結果を計測する。

自工程完結の基本的な姿勢（現場での意思決定）。

仁 = 他人の努力を無駄にしない思いやり。

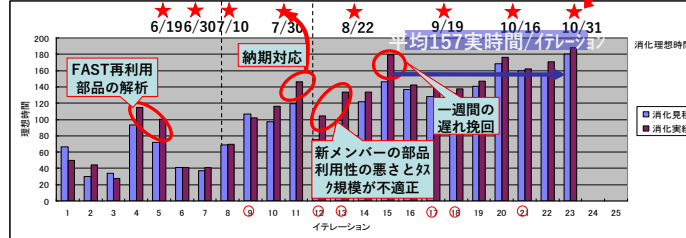
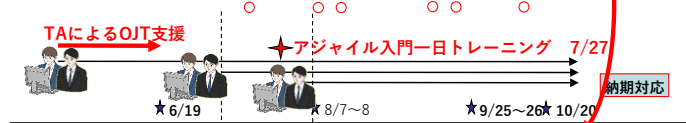
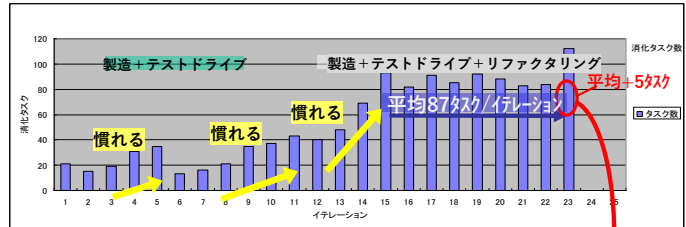
バーンダウン・チャートとベロシティ

[凡例] ★:合宿 ★:リリース ○:実質作業日が5日未満

ベロシティ

チームの作業速度を表す数値

- 1回のスプリントで開発できる量
- 毎週の実施タスク、見積時間、実施時間を収集し平均生産高を見積もる
- ベロシティを一定に保つことが重要



• 説明の流れ

バーンダウン・チャートとベロシティ

チームが進む速度を表し、1回のスプリントで開発できる量と説明する。

コミットメント

スプリントの終了時に、インクリメントを提供
各スプリントのチームの最終目標
これに対し、チームには説明責任が生じる

最善を尽くす！

・説明の流れ

コミットメント

スプリントの終わりに、働くプログラムを提供することで
コミットメントすることを説明する。

スクラムでの役割（登場人物）

スクラムの形態に必要とされる役割は主に以下の3種類

- プロダクト・オーナー
- スクラム・マスター
- 開発チーム / メンバー

・説明の流れ

それぞれの役割の概要を説明する。

プロダクト・オーナー

役割と責任について説明する。優先順位の最終決定等の責任を負う。

スクラム・マスター

役割と責任について説明する。チーム支援が主な役割。

開発チーム

役割と責任について説明する。チーム全体として責任を持って開発を行う。

プロダクト・オーナー

役割と責任

- プロダクトの機能と特徴を定義
- リリースの内容と日程の管理・決定
- プロダクトの収益の見積もりや投資の意義の提示
- 実装する機能の優先順位付け
- チームの作業の成果の認証

・ 補足

【プロダクトオーナー】

1. プロダクトオーナーの役割を簡潔に説明してください。
2. 上記の役割の中で、プロダクトオーナーとしてあなたが重要視する役割は何ですか？またその理由を簡潔に説明してください。
3. ある日声の大きいユーザーからプロダクトバックログ項目の追加が出されました。どの様に考えても優先順位は低そうですが、ユーザーは重要だと言い張っています。さてあなたはこの状況でどの様に対処しますか？
4. プロダクトバックログの記述でプロダクトオーナーとしてあなたが常に気を付けている事はありますか？またそれはどの様な理由からでしょうか？
5. 受け入れ確認（テスト）でプロダクトオーナーのあなたはどの様な基準をもっていますか？またその理由を簡潔に説明してください。
6. プロダクトバックログの記述でプロダクトオーナーとし

てあなたが常に気を付けている事はありますか？またそれはどのような理由からでしょうか？

スクラム・マスター

役割と責任

- チームの機能や効率化、自律を間接的に支援
- チーム開発環境の整備 / モチベーションの向上
- ユーザーへのチームの能力の売り込み
- プロジェクト関係者間の信頼の向上
- ジャスト・イン・タイムの徹底
- ユーザー第一 / 要求に沿った改善の推進

・ 補足

【スクラムマスター】

- 1.あなたのチームのベロシティを安定化させるために、あなたはスクラムマスターとしてチームへの様な気づきを与えますか？
- 2.チームの品質への拘りをどの様に気づかせますか？
- 3.あなたのチームに疲弊感が漂っています。さてあなたはスクラムマスターとしてどの様に対処しますか？
- 4.あなたのチームはカイゼン活動を行っていますか？またその活動はどの様なものですか？
- 5.プロジェクト関係者間のコミュニケーションを円滑化、活発化するためにスクラムマスターとしてあなたが日頃注意、若しくは意識している事を説明してください。
- 6.スクラムマスターとしてのあなたの標準的な一日の行動（タスク）を出社後から退社時まで、順に挙げて説明をしてください。
- 7.上記の中であなたが一番重視する行動（タスク）は何で

すか？その理由を説明ください。

開発チーム

役割と責任

- プロダクトの製作への責任 / リリース日時の徹底
- 4人～10人の範囲で構成
- 一人多能で柔軟性を重視
- 規則に従った上で、各自が自由に開発を進める
- 目標達成のため自ら作業改善を行う
- 争議はチーム内で解決する
- 作業規約を作る

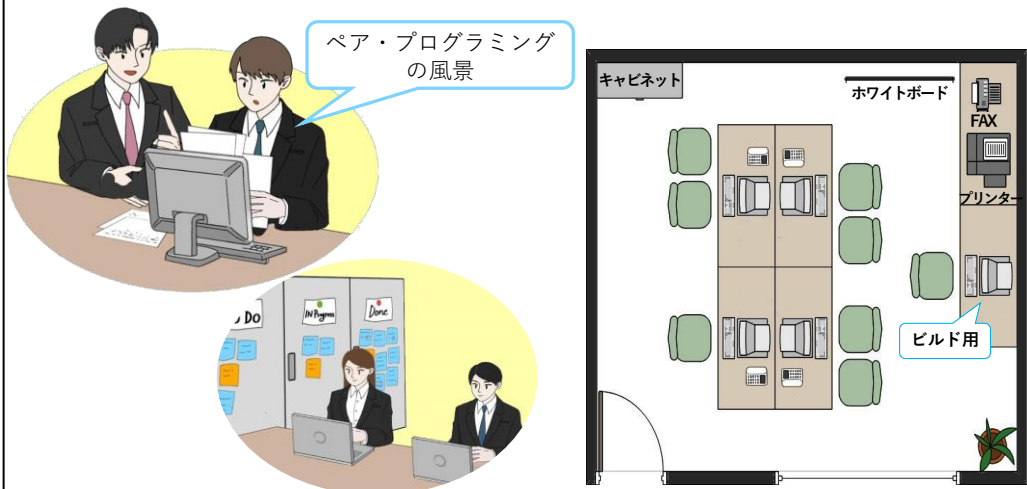
- 説明の流れ

開発チーム

役割と責任について説明する。チーム全体として責任を持って開発を行う。

スクラムの環境

- 透明性の高い作業環境
区画の無い共有作業スペース
→コミュニケーションのとりやすさを意識

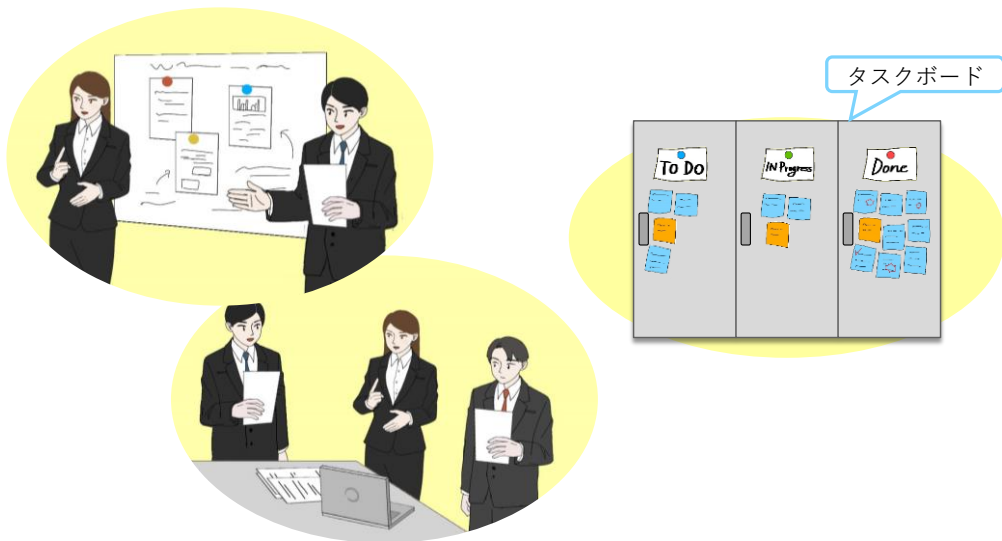


- 説明の流れ

オープンでコミュニケーションの取りやすい環境について説明する。

スクラムの環境

スタンドアップ・ミーティングの風景



・説明の進め方

オープンでコミュニケーションの取りやすい環境について説明する。

第3章クイズ (1)

(問) アジャイル開発チームのチーム編成について説明した文章で適切なものを選びなさい。

1. 開発チームとは異なる顧客側組織のメンバーであるプロダクトオーナーは、開発チームと定期的にコミュニケーションをとるための会議を調整する。
2. 開発チームは、仕切りのない部屋で作業をし、常にフェイス・トゥ・フェイスでの会話行うべきである。
3. 相当のスキルと経験が必要なアジャイル開発のメンバーに、経験の浅いメンバーは採用すべきでない。
4. 開発チームの納期を守るために、スクラムマスターはチームメンバーに的確な指示を出す必要がある。

・ 発問の流れ

正解の番号と正解の理由を説明してください。

他の3つの不正解の理由を簡潔に説明して下さい。

正解は、 2

第3章クイズ (2)

(問) アジャイル開発のスプリント（イテレーション）について説明した文章で適切なものを選びなさい。

1. スプリント計画会議で計画した実施予定のタスクが終了しない場合、プロダクト・オーナーと協議をしスプリントの期間を延長する。
2. アジャイル開発は変化に柔軟である必要があるので、スプリントの期間内でもストーリーの追加や変更を受け入れるべきである。
3. スプリント期間を短ければ短いほど効果的なため、スクラムマスターはチームに短くなるように指導し続けるべきだ。
4. ビジネス価値の探索というスプリントの目的のため、スプリントの最後には動作する成果物をビジネスの関係者にデモを行いFBを受ける必要がある。

・ 発問の流れ

正解の番号と正解の理由を説明してください。

他の3つの不正解の理由を簡潔に説明して下さい。

正解は、 4

第3章クイズ (3)

(問) アジャイル・プロジェクトの初期作業について適切なものを選びなさい。

1. プロジェクトに必要なタスクをリスト化し、最初のスプリントの計画を実施する。
2. 開発チーム、プロダクト・オーナーなど利害関係者の間で、プロジェクトの予算や期間、品質などの制約の優先順位について合意を得る。
3. 開発チームを構成する多様で優秀な人材の経験を重視して、認識合わせは最低限にする。
4. 開発で必要な技術や開発環境の準備は、その都度スプリントの期間内で決定する。

・ 発問の流れ

正解の番号と正解の理由を説明してください。

他の3つの不正解の理由を簡潔に説明して下さい。

正解は、 2

第3章クイズ (4)

(問) 振り返りを実施するタイミングとして適切なものを選びなさい。

1. 問題が生じるたびに、振り返りの実施をスクラムマスターに提案する。
2. 振り返りはスプリントのレビュー会議と一緒に実施すると効果的なので、スプリントの最後に必ず実施する。
3. 振り返りはスプリントの期間と関係なく毎週定期的に行う。
4. 振り返りの実施毎に、話し合いで、次回の振り返りの日程を決める。

・ 発問の流れ

正解の番号と正解の理由を説明してください。

他の3つの不正解の理由を簡潔に説明して下さい。

正解は、 3

第3章クイズ (5)

(問) なかなか全員で守ることのできないチームの規約があります。次の中でチームの取るべき対応策として適切なものを選びなさい。

1. 少し気を付ければ誰でも守れる規約なので、スクラムマスターが翌日のスタンドアップ・ミーティングで全員に注意を促した。
2. 全員が守れていないので、振り返りの場で規約から削除することを提案した。
3. 振り返りの場でその問題(Problem)に対し全員で対策 (Try)を考え、次週に全員で取り組むことにした。
4. 少し気を付ければ誰でもできることなので、スクラムマスターがもう少し様子を見ることにした。

・発問の流れ

正解の番号と正解の理由を説明してください。

他の3つの不正解の理由を簡潔に説明して下さい。

正解は、3

演習-2 ユーザーストーリーの作成（1）

『日用雑貨や書籍を販売するショッピングサイト』のシステムを構築しようとしています。

次のようなプロダクト・バックログ（ユーザー要求）があります。

1. 必要なセキュリティを確保する
2. 画像付の商品カタログを提供する（画像は正面、物によっては背面、側面もある）
3. 在庫している商品を販売する（手配品は今回の範囲外とする）
4. 注文は簡単な操作で行なう
5. 欠品している商品は「在庫なし」と表示する
6. 欠品している商品は注文を受付けない
7. 注文確定時に在庫を引き当てる（在庫管理システムに必要情報を提供する）
8. 在庫が引当てられた場合には受付番号を表示する
9. 在庫が引当てられた場合には注文請書のメールを送信する
10. 引当て時に欠品した場合にはメッセージを表示する（受注残は管理しない）
11. 支払方法は、クレジット、代引き、銀行振込を選択できる
12. 銀行振込の場合には注文請書のメールにて振込先の情報をお知らせする
13. 代引き手数料は350円、送料は全国一律480円とし2万円以上お買い上げの場合は無料（当方負担）とする
14. 初回のお客様には顧客情報を登録して頂く
15. 初回のお客様には顧客情報登録後、ユーザーIDを発行する
16. 出荷業務、配送業務は物流事業者に委託する（出荷管理システムに必要情報を提供する）
17. 銀行振込の場合には入金を確認してから商品を出荷する
18. 発注警告情報、欠品情報などの在庫管理情報をサイト運営担当者に提供する
19. 売れ筋情報・死に筋情報などの売上管理情報をサイト運営担当者に提供する

演習-2 ユーザストーリーの作成（2）

プロダクト・バックログからユーザストーリーを作成してください。
（できるところまでで、結構です。）

ワーク 1

演習時間： 60分

ワーク 2

発表時間： 各チーム5分

第4章 XP概説

- XPの価値感
- 手法の紹介

XPの手法や進め方を学ぶ

アジャイル開発の手法により違いについて理解を深める

XP (エクストリーム・プログラミング)

欠陥 (バグ) のないプログラムの作成手法 (自工程完結の思想の導入)

- Kent Beckにより1999年に発表される
- テスト駆動開発 / Test-Driven Development
 - ① 機能をテストするコードの作成
 - ② テストで問題がなくなるまで修正を加えながら、コードを実装
- リファクタリング
機能に変更を加えず、完成したコードの単純化・明確化の為の任意作業
- ペア・プログラミング
ドライバー(実装担当)とナビゲーター(確認担当)を交代しながら作業
毎日ペアを変更
- 10分間ビルド
自動的にシステム全体をビルド
全てのテストを10分以内に実行

• 説明の流れ

順を追って説明する。

テスト駆動開発 (TDD)

TDDの流れを図とともに解説。

リファクタリング

出来上がったソースコードに対して作業レベル、使用レベルに応じて調整を行っていく。注意点についても説明する。リファクタリング基本方針についても説明する。

ペア・プログラミング

ペアを高頻度で変更することによる緊張感の保持や、1人で考え込む時間を減らすことが目的と説明する。

10分間ビルド

自動的にシステム全体をビルドしていくという手法で時間を短縮し、全てのテストを10分間で終わらせると説明する。時間が限られているので、無意味なテストを避ける工夫につながる。

XPの価値感

1. コミュニケーション

多くの問題の原因

- ・エンドユーザー、プログラマー、マネージャー相互間
- ・システム要件
- ・設計の意図
- ・陣営、トラブル
- ・進捗

XPの多くの実践はコミュニケーションなしには行えない

- ・ペアプログラミング
- ・チーム全体など

2. シンプル

今日、シンプルな実装をする。

- ・明日、必要なもの1まそのとき実装する。
 - ・今日実装した複雑な機能は、明日使わないかもしれない。
 - ・システムがシンプルなら、コミュニケーションはしやすい。
- 最もシンプルなコミュニケーション方法は？
- ・ユーザとの直話会話によるコミュニケーション

4つの価値

3. フィードバック

株動するシステムがもっとも情報量が多いフィードバック

- ・要求：ユーザーにシステムを操作してもらいながら
- ・設計：実際のコードを見たり、機能追加を試してみる
- ・品質：線助しているシステムに対してテストを行う
- ・進捗：動作している機能を見る／実績をみる

4. 勇気

コミュニケーション、シンプル、フィードバックと結びつくと、勇気は極めて価値のあるもの

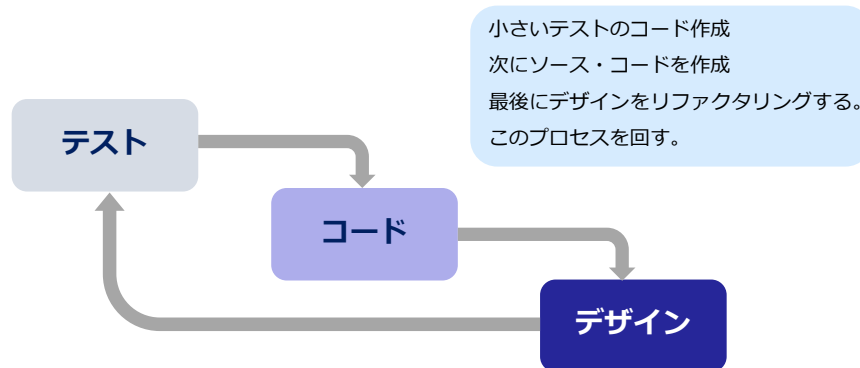
- ・欠陥の修正、コードを捨て去る勇気
 - ・システムをシンプルに保つ勇気
 - ・嘘のないコミュニケーションを行う勇気
 - ・メンバーからのフィードバックを受け入れる勇気
- 人がもっとも必要な要素であると考えている証

- ・説明の流れ
順を追って説明する。

手法の紹介 - テスト駆動開発 (TDD)

•テスト駆動開発

- テストの自動化
- アクセプタンス・テスト (完了・終了の定義)



- 説明の流れ
順を追って説明する。

リファクタリング

特徴

動作確認後、ソースコードに対し以下を実行する

- 可読性の向上やクラス化・単純化
- 使用の容易さ、誤作動抑止などの仕様レベルでの調整

ルール

- 同時にリファクタリングと機能改変を行うことの禁止
- 変更前後の機能変化がないように留意
- メンテナンス性を重視
- 一回の変更は小さくし、繰り返し変更を行う

- 説明の流れ
順を追って説明する。

リファクタリング基本方針

- コードをできるだけ小さな単位に分割する
 - 処理のかたまりに名前をつける
 - 処理を重複して記述しない
 - コードをできるだけ小さな単位に分割する（部品が小さいほうが理解しやすく、バグも少なくなる）
- また、コードを小さく分割することで、そのまとまりごとにメソッド名やクラス名などの名前を付けることができます。実は、この名前がコードがわかりやすくなるかどうかの分かれ目になります。コメントを充実させるよりも、リファクタリングによってコードの断片をまとめ、適切な名前を付けるべしというのがリファクタリングの教えです。（もちろん、実際にはコメントも重要ではありませんが。）
- そして最後に重要なのが、コードの重複をできるだけ少なくすることです。重複コードを見つけたらまず間違いなくリファクタリングの対象になります。重複するコードがあるということは、プログラムの変更時に変更しなくてはならない箇所が分散してしまい、変更時のバグが入り込みやすくなってしまいますということです。それを防ぐためには、重複するコードをできるだけ少なくするというのはコードの保守性を高めるためにきわめて重要な作業となります。ただし、あまりに共通ルーチンを使い回すと、逆にメンテナンスのしやすさを低下させてしまうこともあることを覚えておいてください。あまりに多くのモジュールから、ひとつの共通ルーチンが参照されているときは、その共通ルーチンを変更するときの影響が大きすぎることになってしまいます。このへんのさじ加減を広い視点で判断し、コードの保守を行っていくことが重要です。

・ 補足

どんなプログラマーでもコンピュータに理解できるコードは書ける。しかし優秀なプログラマーだけが、人間に理解できるコードを書くことができる。

～Martin Fowler

ペア・プログラミング

特徴

- 2人で1台の端末を利用し、プログラムを実装する
- ペアは毎日変更
- ドライバー/実装担当とナビゲーター/確認担当に分かれる
- 役割は一定時間で交代
- 品質と可読性の向上
- 共同作業による思考時間の削減



ペアの交替（定期的：毎日）
作業の交替（一定時間毎）



- 説明の流れ
順を追って説明する。

10分間ビルド

特徴

- 自動的にシステム全体をビルド
- 全てのテストを10分以内に実行
- ビルドの対象となるシステムとテストの対象になる部分の把握が重要

• 補足

自動ビルド&自動テストを導入したチームは、全体のプロセスを10分を超えない様にしていた。10分以上掛かった時には、10分以内に収まるように、誰かが最適化をしていた。

テストの自動化

「各々のスプリントで、テスト・カバレッジの小さな向上を目指す」

カバレッジの向上を行う方法

- テストケースを一覧の作成
- テストケースのリスクによる分類
- 優先順位をつけて並べ替え
- テストの部分的な自動化

それぞれのスプリント内で、
一つはテスト自動化のストーリーを実装する。

- 説明の流れ
順を追って説明する。

第4章クイズ (1)

(問) リファクタリングについて適切なものを選びなさい。

1. アジャイル開発のための手法のリファクタリングは、ウォーターフォール開発では利用できない。
2. リファクタリングは特定のタイミングでまとめて実施する方が良い。
3. レガシーコードでテストコードが存在しない場合は、リスクが高いためリファクタリングを避ける。
4. リファクタリングによりユーザーにもメリットが生じる。

・ 発問の流れ

正解の番号と正解の理由を説明してください。

他の3つの不正解の理由を簡潔に説明して下さい。

正解は、 4

第4章クイズ (2)

(問) ペア・プログラミングの説明で適切なものを選びなさい。

1. ペア・プログラミングを行うときは、休憩をこまめに取得させるようなルールが必要である。
2. 高いスキルのメンバーにとっては生産性を下げるため、スキル不足のメンバー間でのみ採用する。
3. ドライバーは、ナビゲーターの指示に従ってソースコードを作成するので、ドライバーは自主的なソースコード設計を行わない。
4. ペアを頻繁に替えるとコーディング対象の機能についての情報を何度も確認する必要が出るため一つの機能のコーディングが完了するまでペアの役割を替えるべきではない。

・発問の流れ

正解の番号と正解の理由を説明してください。

他の3つの不正解の理由を簡潔に説明して下さい。

正解は、4

第5章 アジャイルな計画

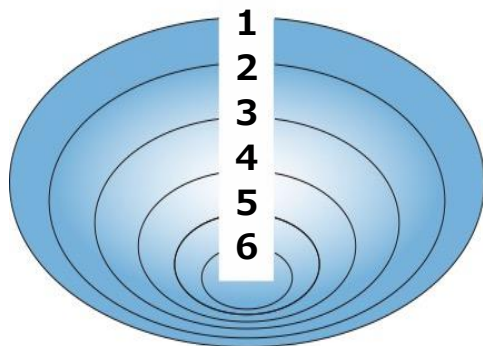
- アジャイル開発で用いる計画手法
- プランニング・オニオン / 2レベル・プランニング
- 《WS》タスク分解（演習）

アジャイル計画の種類・詳細について学ぶ

与えられた課題からタスクの分解や計画の立案ができるようになる

アジャイルな計画づくり

プランニング・オニオン
包含関係の強い計画群



1. 戦略
2. ポートフォリオ計画
3. プロダクト計画
4. **リリース計画**
5. **スプリント計画**
6. **デイリー計画**

アジャイルチームが対象とするのは、リリース計画、スプリント計画、デイリー計画

・説明の流れ

プランニング・オニオンと言う概念の意味を再確認する。

・補足

ウォーターフォール開発は開発、ソフトウェア（システム）を作る手法で、プロジェクト単体での計画立案にフォーカスしているが、アジャイル開発はITサービスの実装する手法であるので、そのITサービスを利用するビジネスとの連携が最大の課題となる。

従って、ビジネス戦略からの一連の計画立案の流れの中にアジャイル開発の計画は組み込まれている。

アジャイルチームの計画づくりで対象とするのは、リリース計画、イテレーション計画、デイリー計画

反復性増加型開発の計画



- 2レベル計画
 - リリース計画…全体の大まかな計画
 - スプリント計画…作業に対する詳細な計画
- 連続的な計画
- 実測駆動によるベロシティの測定

→ 作業の見通しを、より早く行う

・補足

リリース計画がユーザーへのコミットメントや開発スケジュールの目安となる。

スプリント計画（イテレーション計画）は開発チームの為のモノ。

常にこの二つの計画を見直し、修正を加える。（毎スプリントごとに）

リリース計画とスプリント計画

リリース計画

期間

1ヶ月先

対象

ユーザーストーリー

単位

ストーリーポイント等

スプリント計画

期間

スプリント1回分
(1週間先)

対象

タスク

単位

時間

- ・ 説明の流れ
順を追って説明する。

リリース計画の中で複数のスプリント計画を繰り返す。見直しが連続的に行われるため、早めの見通しを立てることができると説明する。

リリース計画とスプリント計画

アジャイルプロジェクトはリリースの繰り返しから、
リリースはスプリントの繰り返しから成る

プロダクトオーナーが責任を持ち、プロダクトバックログを基準とする
ユーザや顧客は、レビューやリリース終了ごとに関わる
顧客のFBをもとに、計画を改善する

- ・ 説明の流れ
順を追って説明する。

全体リリース計画の立案

1. ユーザストーリーやエピックの規模の見積もり
※エピック … 複数のストーリーの集合
2. スプリントの期間の設定
3. ユーザストーリー、エピックの優先順位の決定
4. スプリントでのベロシティの見積もり
5. ユーザストーリー、エピックを配分

- 説明の流れ
計画はあくまでもチームの管理のためのもの

直近のリリース計画の立案

1. リリースの目標を確認
2. エピックのユーザーストーリーへの分割
3. ユーザーストーリーの規模を見積もり
4. スプリントの期間の設定
5. ユーザーストーリーの優先順位の決定
6. スプリントでのベロシティの見積もり
7. スプリントにユーザーストーリーを配分
8. リリース日時の決定

- ・ 説明の流れ
順を追って説明する。

8つの工程に分けて立案する過程を説明する。

反復にユーザーストーリーを配置していく過程について説明する。1回の反復にストーリーが収まりきらなければ分割もでき、分割のガイドラインについても説明する。

リリース計画の立案 反復にユーザーストーリーを配置する



- 1スプリントに期待できるペロシティを基準にする
- 最初のスプリントは詳細に計画
- 二つ目以降のスプリントは変更が生じうるので柔軟に変更
- 各スプリントでのストーリーの数は異なる(ペロシティが一定)

• 説明の流れ

反復にユーザーストーリーを配置していくとは左図のようになる。

ストーリーの分割

小さい単位であることが求められるストーリーは以下の場合等で分割される

- ベロシティの基準により、ストーリーがスプリントに収まらない場合
- 計画しているスプリントに組み込めない場合
- 正確な見積もりを要する場合

- 説明の流れ
順を追って説明する。

ストーリーの分割のガイドラインについて、どんな要素によって分割するのか説明する。

タスクの定義に付いて解説する。スクラムプロセスの中ではタスクがどこに位置するのかについても図で確認する。

ストーリーの分割のガイドライン (1)

- ユーザーの役割による分割
- データの境界による分割
- 操作の境界による分割
 - プロセスによる分割
 - ファンクション(CRUD)による分割
 - Create/作成 Read/参照 Update/更新 Delete/削除

• 説明の流れ

分割のガイドラインを順を追って説明する。

A T Mの利用者→入金、出金、振込

バランスシートのデータの場合→資産のデータ、負債のデータ、資本のデータなど

ストーリーの分割のガイドライン (2)

- 優先度による分割

Ex.オンラインゲームの対戦者→1人の対戦者、2人の対戦者、複数の対戦者

- ビジネスルールによる分割

Ex.商品の発注→定番品、特注品など

商品の値引→1万円以上、2万円以上、3万円以上など

- 横断的機能の括り出しによる分割

Ex.セキュリティ、エラーハンドリング、ロギング

- 説明の流れ

分割のガイドラインを順を追って説明する。

オンラインゲームの対戦者→1人の対戦者、2人の対戦者、複数の対戦者

商品の発注→定番品、特注品など

商品の値引→1万円以上、2万円以上、3万円以上など

ストーリーとタスク

タスクの特徴

- ストーリーの実現に必要な作業を細分化したもの
- 作業量が一時間以内となる量
- スプリント・バックログでリスト化
- スプリント中にも書き換えられる

- 説明の流れ
順を追って説明する。

スクラム・プロセス

プロダクトバックログ

- 1.-----
- 2.-----
- 3.-----
- 4.-----
- 5.-----



プロダクトオーナー
(お客様)

確実な優先順位付け
(同位は無し)
追加、修正、変更可能
(スプリントに入っている物以外)
プロダクトオーナーが
全て責任を持つ

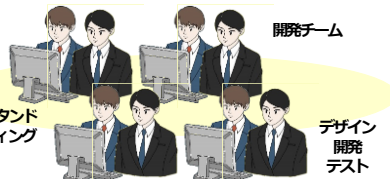
スプリントプランニング

Part1	1時間
Part2	2時間

スクラムマスター
(ファシリテーター)



毎朝15分のスタンド
アップ・ミーティング



開発チーム

デザイン
開発
テスト

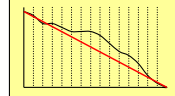
タスクボード

To Do	On Going	Done
c	b	a
d		
e	g	f
h		

Sprint Task List
(Sprint Backlog)
a.-----
b.-----
c.-----
d.-----
e.-----
f.-----

デイリースクラム
(スタンドアップ・ミーティング)

バーンダウン・チャート



スプリントレビュー
1時間

レトロスペクティブ
(振り返り)
1時間

スプリント(イテレーション)

1週間~4週間

- ・ 説明の流れ
順を追って説明する。

スプリント計画ミーティング

1. プロダクト・バックログ
2. チームの作業能力
3. ビジネス状況
4. 技術、品質の安定性
5. 実行可能な作業量

レビュー
検証
整理

次のスプリント目標
スプリント・バックログ

パート1

開発チームはプロダクト・オーナーと一緒に次のスプリントでどのような機能を構築するかを決定する

- ①プロダクト・バックログの提示
- ②次のスプリントで対象とするバックログの選定
- ③スプリント目標を設定
 - プロダクトバックログの実装を通して満たされるべき目標
- ④ストーリーの確認

パート2

開発チームがその機能をどのようにプロダクトに追加するかを決定する

- ①ストーリーの作成
- ②リリース内容と予定
 - プロダクト・オーナーへ報告
- ③スプリント・バックログ項目(タスク)の定義
- ④タスクの作業時間見積り

・説明の流れ

下記2つを発問してみる

タスクとは

ストーリーとは

スプリント計画の立案 ストーリーをタスクに展開

ストーリー	タスク		
ストーリー1	タスク 1-1 1	タスク 1-2 2	タスク 1-3 1
ストーリー2	タスク 2-1 2	タスク 2-2 1	
ストーリー3	タスク 3-1 1	タスク 3-2 2	タスク 3-3 1.5

・説明の流れ

順を追って説明する。

スプリント・ミーティングの流れを説明し、ストーリーとタスクがどこに関連してくるのか確認する。タスクへの分割（タスクの定義）についても説明する。

タスクの見積もりには個々人の認識の違いを反映させないよう、プランニング・ポーカーを用いる方法も紹介する。

タスクへの分割（タスクの定義）

スプリント・バックログ（タスク）

- スプリントの目標を達成するために行わなければならないタスクのリスト
- タスクは、プロダクト・バックログ（要求）を動作するソフトウェアとして実装するために必要な作業を詳細に分割したもの
- タスクの作業量は時間数で見積る
 - 実測駆動
- タスクは割り当てるのではなく、メンバーが自ら担当するタスクを選択する
- チームはスプリントを通して必要に応じてスプリント・バックログを更新し続ける
 - スプリント・バックログを変更できるのはチームだけ

- 補足

新しい作業が必要になったらチームはその作業をスプリントバックログに追加する

不必要に思えるタスクがあったら取り除く

スプリント・バックログと、プランニング・ポーカーによるタスクの見積もりについて説明する。

粒度は小さいほうがいいということを前提とし、小さくすることによる利点について、例を交えて説明する。

タスクの見積

プランニング・ポーカー



相対的に見積もる
実測駆動での見積（前回の実績値）

- ・ 説明の流れ
プランニングポーカーのやり方。

タスクの粒度

- タスクの粒度を小さくすることで

- 流れを作る
- 負荷を一樣にする
- 柔軟性を高める

- 1時間～1日を基準
- テスト実行の容易さを重視
- 作業指示書と同義で、運営が円滑になる
- 作業内容に依存させない

Statements of Source code	1	10	100	1,000	10,000	100,000	1,000,000	10,000,000
Test Cases	1	2	5	15	250	4,000	50,000	350,000
Test Coverage	1	1	1	1	1	0	0	0

Application size, Test Cases, and Test Coverage.
Logical source code statements

By Caper Jones

- 補足
タスクの粒度へのこだわりが品質管理上重要です。

タスク粒度 (例)

タスクの粒度 : 1時間のタスク

タスクの均一性: 作業内容に関係なくできるだけ同じ粒度

	見積 総時間	実質 総時間	タスク 総数	見積 粒度 (H)	実質 粒度 (H)
第1週~第7週 インテレーション ユーザー操作系が主な機能1回目	197.8	211.9	134	1.476	1.581
第8週~第16週 インテレーション 機械制御系が主な機能1回目	418.4	452.2	295	1.418	1.533

平準化すると多様化に対応できる。

全く異なった機能実装
インテレーションでも
同じ粒度で平準化している。

・補足

どのような性格のアプリでもタスクへの分割には、影響を与えない。

理由は作業アイテムに分解しているから。

タスクを小さくする

例えば、レポートを作る業務（仕事）をタスク分解する。

1. レポートの主旨の確認	30分
2. レポートの章立てを決定	10分
3. 各章の基本構造を決定	30分
4. 文章の下書き	30分
5. データの収集	30分
6. PC、アプリケーションの起動	5分
7. データのインポート	20分
8. グラフの作成	10分
9. 文章の構成	30分
10. レイアウトの決定	5分
11. レポート全体の校正	30分
12. 体裁の調整	5分
13. レポートの提出	5分
合計	240分

・補足

タスク分解の練習をしてみよう。
慣れると素早くできる様になる。

第5章クイズ (1)

(問) アジャイル開発プロジェクトにおける計画作りについて適切で無いものを選びなさい。

1. 計画はストーリーとその優先順位をもとに決定する。
2. タスク単位での計画は、スプリント計画時に立案する。
3. 綿密な計画を遅延やトラブルなく完遂することがプロジェクトの成功の証である。
4. 顧客のビジネス環境の変化によって発生した新たに必要なストーリーや、リストアップされているストーリーの価値に変化に柔軟に対応しながら、計画は随時変更する。

・発問の流れ

正解の番号と正解の理由を説明してください。

他の3つの不正解の理由を簡潔に説明して下さい。

正解は、3

第5章クイズ (2)

(問) ストーリーポイントを付与する際に注意すべき点で適切なものを選びなさい。

1. 類似したストーリーを集め、できるだけ大きな塊で見積もりを行う。
2. 詳細なタスク単位での見積もりは、プロジェクト全体の計画時に行う。
3. ストーリーの実現に必要な全作業を見積もる。
4. チームリーダーやスクラムマスターなど特定の一人が見積もりを行い、ばらつきを回避する。

・ 発問の流れ

正解の番号と正解の理由を説明してください。

他の3つの不正解の理由を簡潔に説明して下さい。

正解は、 3

演習-3 タスクへの分解

『スクラムのプランニング・セッションを実行する (3～4時間)』
と言うプロダクト・バックログをタスクに分解する。

ワーク 1

演習時間： 30分

ワーク 2

発表時間： 各チーム5分

第6章 品質管理

- アジャイル開発での品質
- 《WS》振り返り（KPT）（演習）

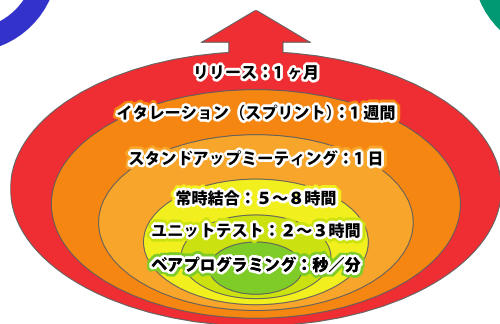
従来の方法と品質がどのように変わるのかを学ぶ

振り返りの重要性を理解し、KPT手法を使用できるようになる

品質について考える



- ・ バグフリー
- ・ アジャイル・プラクティスの徹底 (守)
- ・ 頻繁なフィードバック・ループの設計



- ・ メンテの容易性
- ・ ソースコード = ドキュメント・メソッド名と変数名 (名は体を表す)
- ・ 簡易に書けるセカンド・ランゲージ (自分のツール作成)

・ 説明の流れ

当たり前の品質と魅力的な品質に分けてどんなものが該当するのか説明する。

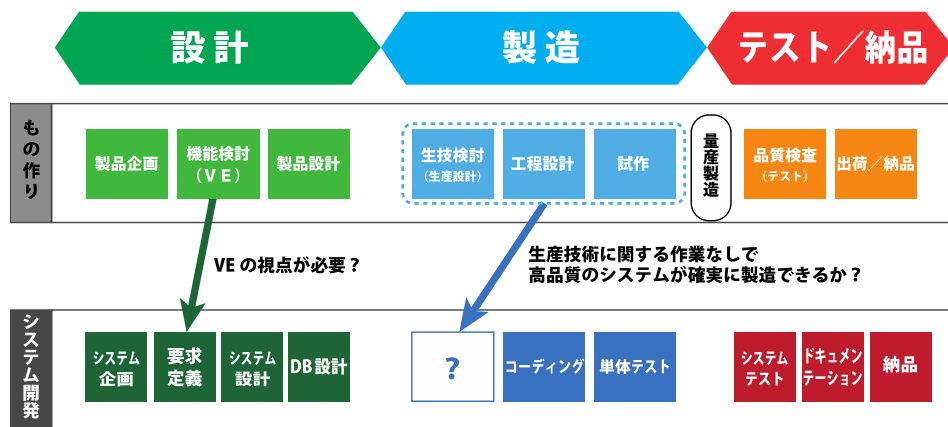
アジャイル開発で品質が向上する理由

- ムダ取りの原則
時間的な境界を明確化 / 作業のムラの減少
- タスクの粒度の極小化
作業の流れを向上 / ミスの早期発見
- 全員の作業の透明化
協力の易化 / 思考時間・修正時間の削減
- 開発の自律性
自工程完結で周囲への影響が減少
全員が他人の作業を把握
- 開発者とユーザの直接的な交流
変更への柔軟性 / 意思の伝達

• 説明の流れ

理由として5つの要因を挙げて、各要因について例を交えて説明する。

『もの作り』と『システム作り』の相違



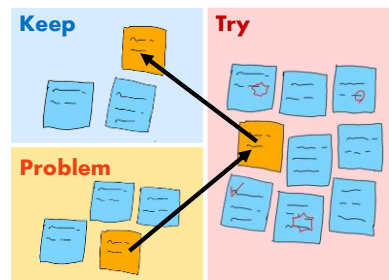
設計仕様を図面上のみの検討で十分か?
擦り合わせ、微調整は必要ないだろうか?

・説明の流れ

高品質なシステム作りにおいて、もの作りと対比しながら、相違点について説明し、どうしたら高品質なシステムが完成するか考えさせる。

KPTでの振り返り（レトロスペクティブ）

- 毎週行われる進捗報告ミーティングで、プロセスの改善を実施
- KPT(Keep/継続 Problem/問題 Try/挑戦)をチームが共有
- 小さなPDCA



小さなPDCAを回すための方法です。

KPTボードは振り返りのツール

- ① 気づきや問題、課題を付箋紙に書き、Problem（問題）のエリアに貼る。
- ② Problemのエリアに出てきた気づきや問題に対してTry（対策）を立ててTryのエリアに移動させる。
- ③ 実際にやってみて、対策が不十分だった場合には、再度、Problem（問題）のエリアに移動し課題を追記する。再度、対策を立て直す。
- ④ Problemに対する効果が十分出たと判断できた場合には、Keep（継続）のエリアに移動し習慣化、定着化を図る。

ここで大事な事は、**人を責めずにやり方を責める**事です。（誰でも同じようにできる様にする。）

・説明の流れ

KPT法の概要と手法について説明する。

（参考）TQMとは

総合的品の一連の活動のこととして紹介する。質管理のことであり高品質なシステムを提供するための一連の活動のこととして紹介する。

(参考) TQMとは

TQM (Total Quality Management)総合的品質管理とは、トップマネジメントのリーダーシップのもとに組織が一丸となって、お客様が高度に満足する製品を生産したり、サービスを提供するための一連の活動です。TQMは製造業、サービス業など全ての業種において、また小さな組織から大きな組織まで組織の規模に関係なく有効な活動です。

- ◆ 品質の良さはお客様の満足によって測られる。
- ◆ 管理とは目標を決め達成するための活動です。
 - 維持と改善
 - プロセスの良さと結果の良さ
- ◆ 総合的な活動の意味
 - 品質を総合的に考える
 - 組織全体が総合的に参加する
- ◆ 基本原理は、PDCAと継続的な改善
- ◆ 結果を押さえ込むのではなく、プロセスで作り込む
- ◆ 後工程はお客様
- ◆ 応急対策と再発防止策の探索
- ◆ TQMの中核はデータで語る
- ◆ プロセスのレベルアップは5S（整理、整頓、清掃、清潔、躰）
- ◆ プロセスの基礎は標準化
 - アメリカは手順重視、日本は教育重視
- ◆ 改善のステップ
 - 背景の整理、現状の分析、要因の探索（5回の何故）、対策の立案、効果の検証

・説明の流れ

(参考) TQMとは
総合的品の一連の活動のこととして紹介する。質管理のことであり高品質なシステムを提供するための一連の活動のこととして紹介する

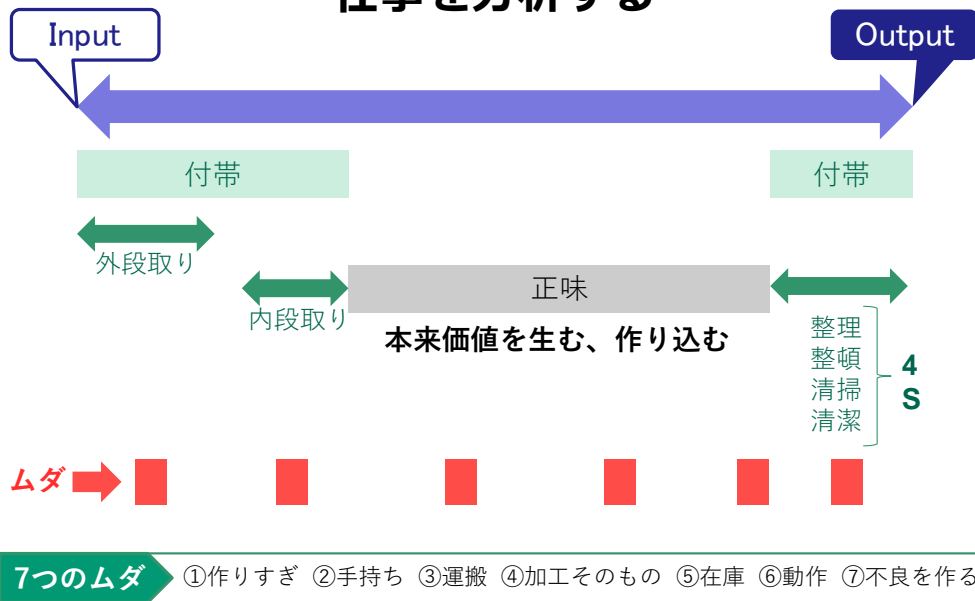
ソフトウェア製造工程におけるムダの廃除

- **作りすぎのムダ**
不必要・価値のない機能の実装
- **手持ち（停滞）のムダ**
指示・許可待ちや障害などによる作業停止
- **運搬のムダ**
作業の切り替えや重複チェック
- **加工そのもののムダ**
事務作業などの過剰な作業
- **在庫のムダ**
中間的な生成物の残留
- **動作（作業）のムダ**
作業場所や開発ツールの途中変更
- **不良をつくるムダ**
バグの発生、コードの欠陥

・説明の流れ

7つのムダの例を挙げながら廃除の方法を説明する。

仕事を分析する



・説明の流れ

また、仕事を分析する観点から7つのムダがどのように発生するのか説明する。

第6章クイズ

(問) アジャイル開発における品質の確保について適切では無いものを選びなさい。

1. 不良を素早い発見と除去を重視し、不良のない状態を維持する。
2. スプリントの成果物の品質について、スプリント開始時にプロダクトオーナーと合意を形成する。
3. 品質管理を重要視するので、進捗を計画通りに進めるための残業はやむをえない。
4. 継続的インテグレーションはアジャイルの品質確保に有効な代表的なプラクティスの一つである。

・ 発問の流れ

正解の番号と正解の理由を説明してください。

他の3つの不正解の理由を簡潔に説明して下さい。

正解は、 3

演習-4 : KPT (振り返り)

ワーク 1	第一スプリント演習	30分
	振り返り	10分
	休憩	5分
ワーク 2	第二スプリント演習	30分
	振り返り	10分
	休憩	5分
ワーク 3	第三スプリント演習	30分
	振り返り	10分
	休憩	5分
ワーク 4	チームの発表 (各チーム5分)	
ワーク 5	アドバイス	
	良い点	
	改善したほうが良い点	

演習課題の選択

6ヶ月後のあるべき姿を定義し、初めの3ヶ月間の行動計画を立案する。

1. フルマラソン (42.195km) を走れる様になり、競技会へ参加できる。
2. メタボなのでダイエットを行う。
3. ゴルフのシングル・プレーヤーになる。
4. 英語が流暢に話せるようになる。(TOIEC x x x点をクリアー)
5. 仕事量は落とさずに残業をしないで、毎日定時に帰宅できるようになる。(ワークライフ・バランスの取れた日常生活)

第7章 大規模プロジェクト

- スクラム・オブ・スクラムの紹介
- 他の大規模プロジェクトの手法紹介

大規模プロジェクトで使用されるアジャイル開発について
具体例から学ぶ

小規模な開発との手法の違いについて理解する

スクラム・オブ・スクラム

大規模なプロジェクトで複数のスクラムチームを構成



プロダクトオーナー

デイリースクラム



スクラムマスター

デイリースクラム



チーム



スクラムマスター

デイリースクラム



チーム



スクラムマスター

デイリースクラム



チーム



スクラムマスター

デイリースクラム



チーム

スクラム
オブ
スクラム

・説明の流れ

FBI センティネル・プロジェクト
スクラム・オブ・スクラムは大規模なプロジェクトで複数のスクラムチームを構成することと説明し、具体的なスクラム事例を用いて解説する。

FBI センティネル・プロジェクト

2005年プロジェクト開始

- 予算4億5100万ドル
- 2009年完成予定

2010年3月時点

- 4億500万ドル(予算の90%)を使用、追加予算見積もり3億5000万ドル
- プロジェクトの進捗は50%、完成まであと6年から8年
- FBIの契約スタッフ数220名、FBI職員数30名

スクラムによる解決案

- 予算残額2000万ドル
- 12ヶ月以内で完成
- FBIの契約スタッフ数40名、FBI職員数12名

結果

- 予算の5%
- 20ヶ月で完成

・説明の流れ

具体的なスクラム事例で説明する。

KANBANの特徴

KANBANシステム

- プル型のシステム
- KANBANボードにより仕事が可視化
- チームの許容量に応じて、カンバンで仕掛（WIP）を制限可能
- 漸次改善を促すための改善ツール

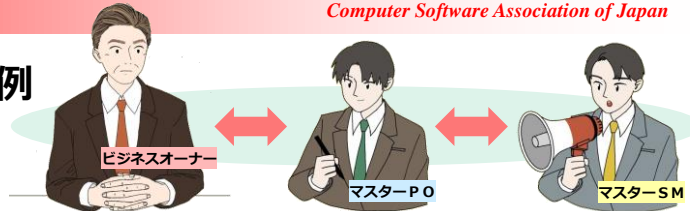
メリット

- プロセス上の問題やボトルネックなどのスループットが可視化
- 品質向上や生産性向上を促進

・説明の流れ

大規模プロジェクトにおけるKANBANの有用さを、例を交えながら説明。またスクラムとの類似点、相違点についても解説する。

KANBANの例



プロダクトバックログ	担当PO	分析&設計	ユーザーストーリー	開発チーム	テスト待ち	テスト	ユーザー受け入れテスト	
XXXXX	■ 山雄	X	X-1		X			
AAAA		A	X-2					
BBBB		B	X-3			B		
CCCC		C	X-4					
DDDD		D	A-1				C	
EEEE	○ 川子	G	G-1		G			
FFFF		H	G-2					
GGGG		I	G-3				H	
HHHH		J	G-4					
IIIIIIII		K	J-1					
JJJJJ		L	J-2					
KKKK		M	J-3					K
LLLL								
MMMM								



・説明の流れ

具体例を交えて説明する。

KANBANとスクラムの類似点

- どちらもリーンであり、アジャイルです。
- どちらもプル・スケジューリングに基づいています。
- どちらもWIP（仕掛）を制限します。
- どちらもプロセス改善を推進するために透明性を活用します。
- どちらもリリース可能なソフトウェアを早く頻繁に提供する事を重視します。
- どちらも自己組織化したチームに基づきます。
- どちらもワークを部品に分割する事を要求します。
- どちらの場合でも、リリース計画は、経験値（データ：ベロシティー/リードタイム）に基づき常に修正（最適化）されます。

Henrik Kniberg (www.crisp.se) のAgile2010, Orlandoのプレゼンテーション
“KANBAN and Scrum - Making the most of both-”より抜粋

・説明の流れ

一つ一つ順を追って説明する。

KANBANとスクラムの相違点

スクラム

1. タイムボックス化されたイタレーションが規定されている。
2. チームがそのイタレーション内での明確な作業量をコミットする。
3. 計画やプロセス改善のための測定基準にベロシティを使用する。
4. 機能横断的なチームが規定されている。
5. 項目が細分化されるので、1スプリント内で完了できる。
6. バーンダウンチャートが規定されている。
7. (スプリント毎に)WIPが間接的に制限されている。
8. 見積りが、規定されている。
9. 実施中のイタレーションに項目を追加できない。
10. スプリント・バックログは特定のチームが所有する。
11. 三つの役割が規定されている。
12. スクラム・ボードはスプリントでリセットされる。
13. 優先順位が付いたプロダクト・バックログが規定されている。

KANBAN

1. タイムボックス化したイタレーションは任意である。
2. コミットメントは任意である。
3. 計画やプロセス改善のための測定基準はリードタイムを使用する。
4. 機能横断的なチームは任意であり、専門家のチームも許容している。
5. 特に項目のサイズは規定されていない。
6. 特に使用する図表は規定されていない。
7. (ワークフローの状態)WIPは直接的に制限されている。
8. 見積りは、任意です。(開発チームが見積りをする事はムダ)
9. 容量が空いている限りいつでも新しい項目を追加できる。
10. KANBANボードは複数のチームや個人で共有される。
11. 如何なる役割も規定されていない。
12. KANBANボードは持続性がある。
13. 優先順位付けは、任意です。

Henrik Kniberg (www.crisp.se) のAgile2010, Orlandoのプレゼンテーション
"KANBAN and Scrum - Making the most of both-"より抜粋

・説明の流れ

一つ一つ順を追って説明する。

KANBANの利用価値

- プロセスの可視化のツールとして進捗状況表示に利用可能
- スクラムと組み合わせ、効果を増強
 - チーム外のステークホルダー間への状況の透明化
 - ソフトウェアツール（RTCやTFS）による支援
- スクラムとの相違点を理解した上での導入の検討が重要

• 説明の流れ

スクラムとどちらを採用するのか、あるいは組み合わせで活用するのか、利用方法について説明する。

まとめ

アジャイル開発をどう活用するか？
(グループ・ディスカッションと発表)

ワーク 1

演習時間：45分 (ディカッションとまとめ)

ワーク 2

発表時間：各チーム5分

アジャイル開発

目次

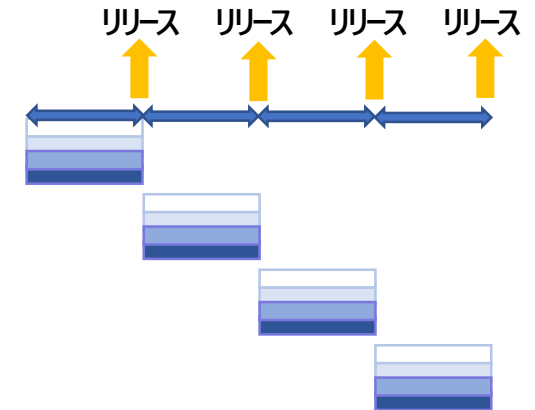
第1章	アジャイル開発の概要	4
第2章	スクラム	14
第3章	スクラムのプロセスと役割	20
第4章	XP概説	55
第5章	アジャイルな計画	66
第6章	品質管理	89
第7章	大規模プロジェクト	100

第1章 アジャイル開発の概要

- アジャイル開発の概要
- アジャイル宣言とアジャイル原則
- アジャイルの価値観
- パラダイム・シフト

アジャイル開発の概要（特徴） 1

◆ 反復型（繰り返し）の開発

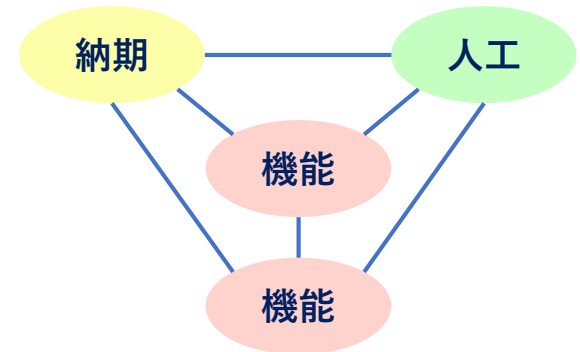
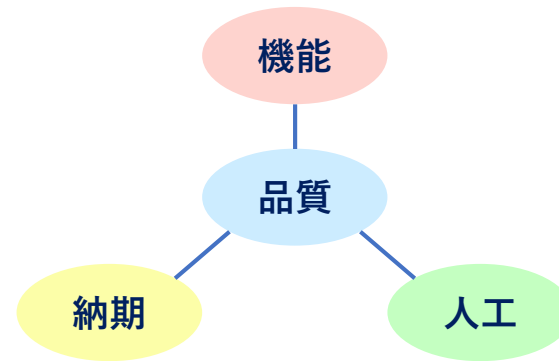


◆ 一人多役（多能）



アジャイル開発の概要（特徴） 2

◆ スコープで調整



◆ 常時リリース

◆ 適応型開発プロセス

アジャイル開発の概要（必要なスキル）

1. 分析設計スキル

ユーザーストーリー / タスク分解と見積り

2. 開発スキル

テスト駆動開発 / リファクタリング / ペア・プログラミング (XP)

3. テストスキル

継続的インテグレーション / 10分間ビルド (XP)

4. プロジェクト管理スキル

透明化 / タイムボックス

5. 品質管理スキル

DoD (完了基準)

6. チームビルディング・スキル

協調 / コミットメント / 合意 / カイゼン / リーダーシップ / チーム規約

アジャイル開発概論

アジャイル開発とは、高品質で要求に柔軟なソフトウェア作成を行うため、協調性、自律性を重視した反復的な開発方法である。

- ダイナミックシステムズ開発技法
Dynamic Systems Development Method (Dane Faulkner ほか)
- アダプティブソフトウェア開発
Adaptive Software Development (Jim Highsmith)
- クリスタルメソッド
Crystal Methods (Alistair Cockburn)
- スクラム
Scrum (Ken Schwaber, Jeff Sutherland)
- XP / エクストリームプログラミング
(Kent Beck, Eric Gamma ほか)
- リーンソフトウェア開発
Lean Software Development (Tom and Mary Poppendieck)
- フィーチャ駆動開発
Feature-Driven Development (Peter Code, Jeff DeLuca)
- アジャイル統一プロセス
Agile Unified Process (Scott Ambler)

- 反復（周期）的(Iterative)

- 定期的なリリース

- 漸進的(Incremental)

- 徐々に機能を増加

- 適応主義(Adaptive)

- 変化に対応（即応）

- 自律的(Self-Organized)

- 学習する組織

- 多能工(Cell Production)

- 一人多役

（SE、プログラマー、テスター）

アジャイルソフトウェア開発宣言

2001年2月 (於 米国ユタ州スノーボード) 制定

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する活動を通じて、より良い開発方法を見つけ出そうとしている。

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、
計画に従う事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置く。

Kent Beck
Mike Beedle
Arie van Bennekum
Alistair Cockburn
Ward Cunningham
Martin Fowler

James Grenning
Jim Highamith
Andrew Hunt
Rin Jeffries
Lon Ker
Brian Marick

Robert C.Martin
Steve Mellor
Ken Schwaber
Jeff Sutherland
Dave Thomas

<http://agilemanifesto.org/>

アジャイルソフトウェア開発の原則（1）

私たちは、以下の原則に従う。

1. 顧客満足を最優先し、
価値あるソフトウェアを早く継続的に提供します。
2. 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。
3. 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。
4. 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。
5. 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。
6. 情報を伝える最も効率的で効果的な方法は
フェイス・ツー・フェイスで話をすることです。

アジャイルソフトウェア開発の原則（２）

7. 動くソフトウェアこそが進捗の最も重要な尺度です。
8. 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。
9. 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。
10. シンプルさが本質です。
11. 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。
12. チームが最も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。

<https://agilemanifesto.org/iso/ja/principles.html>

アジャイルの価値観

アジャイル開発と他のソフトウェア開発手法との違いを整理します。

アジャイル開発

- タイムボックス制
- 完全なチーム作業
- ユーザー第一主義
- 一人多役
- チームの自己組織化

他の開発手法

- (← 長期間)
- (← PMの存在)
- (← 分業制)

アジャイル開発においては、パラダイム・シフトが必須！

正解主義から適応主義への変遷！

第1章クイズ (1)

(問) アジャイル開発でのドキュメント作成について適切なものを選びなさい。

1. 設計根拠となったドキュメントは、プログラムの動作確認後は不要である。
2. 保守に必要なDB設計書は、リリース後の最新状態のものをツールを自動生成により用意した。
3. それぞれのプログラマーの個性を活かすため、コーディングルールなどは作成しない。
4. 保守・運用に必要な情報は全て口頭で伝え、ドキュメントは作成しない。

第1章クイズ (2)

(問) アジャイルソフトウェア開発の原則について適切なものを選びなさい。

1. チーム内での対話によりリスク回避が可能なので、できる限り長く会議の時間を取る。
2. 効率は良いが正確性に欠ける対話ではなく、ドキュメントベースの情報伝達を推進するのが良い。
3. 最も効率的なコミュニケーションの方法は、フェイス・トゥ・フェイスの対話であり、積極的な活用が求められる。
4. 顧客との仕様調整に有効なコミュニケーション手段の対話のため、対話のスペシャリストを選任で用意するべきである。

第2章 スクラム

- スクラムの価値観（働き方）

スクラムの価値観（働き方）

スクラムの三つの価値

- ◆ 監視 Inspection
- ◆ 透明性 Transparency
- ◆ 調整 Adaptation

プラス

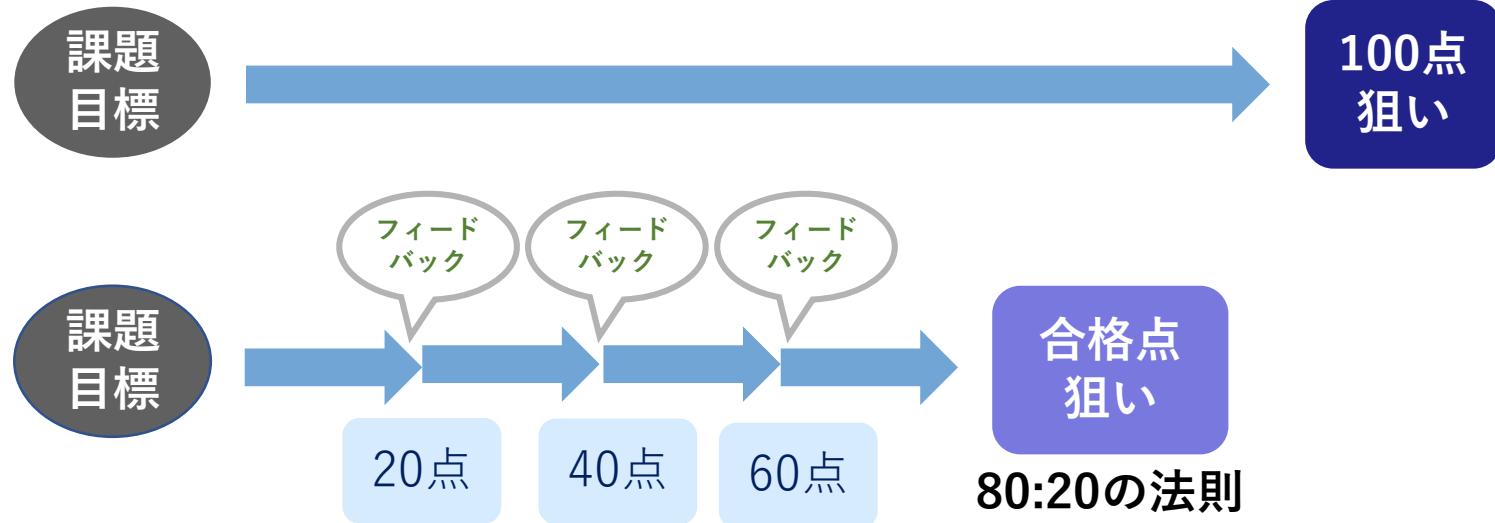
タイムボックスの利用

アジャイル開発の仕事の基本構造

イテレーション（スプリント）

- 時間の使い方
- 透明性
- 品質

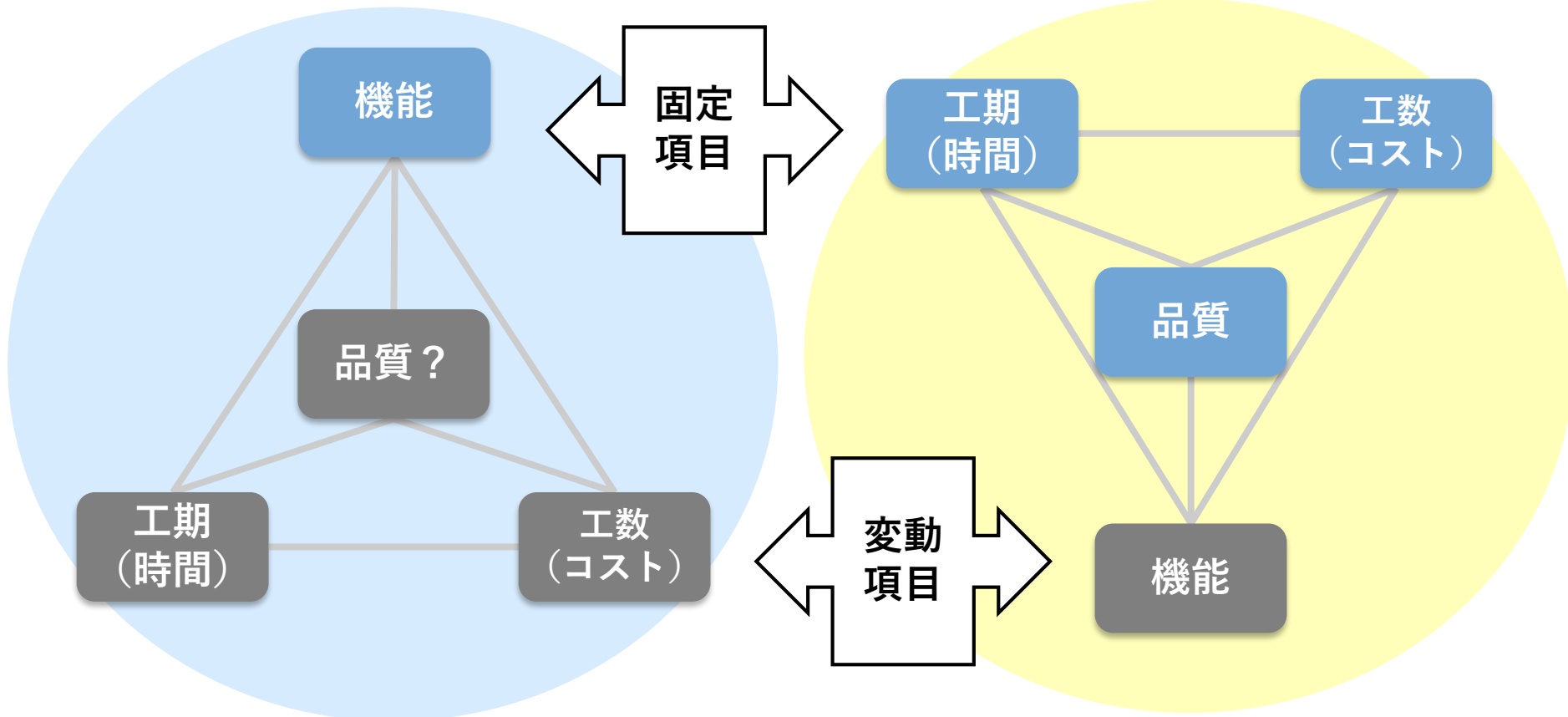
反復的に開発を行った方が
良いのか？
どうして、そうしなければ
いけないのか？



アジャイルの価値観

従来の開発アプローチ

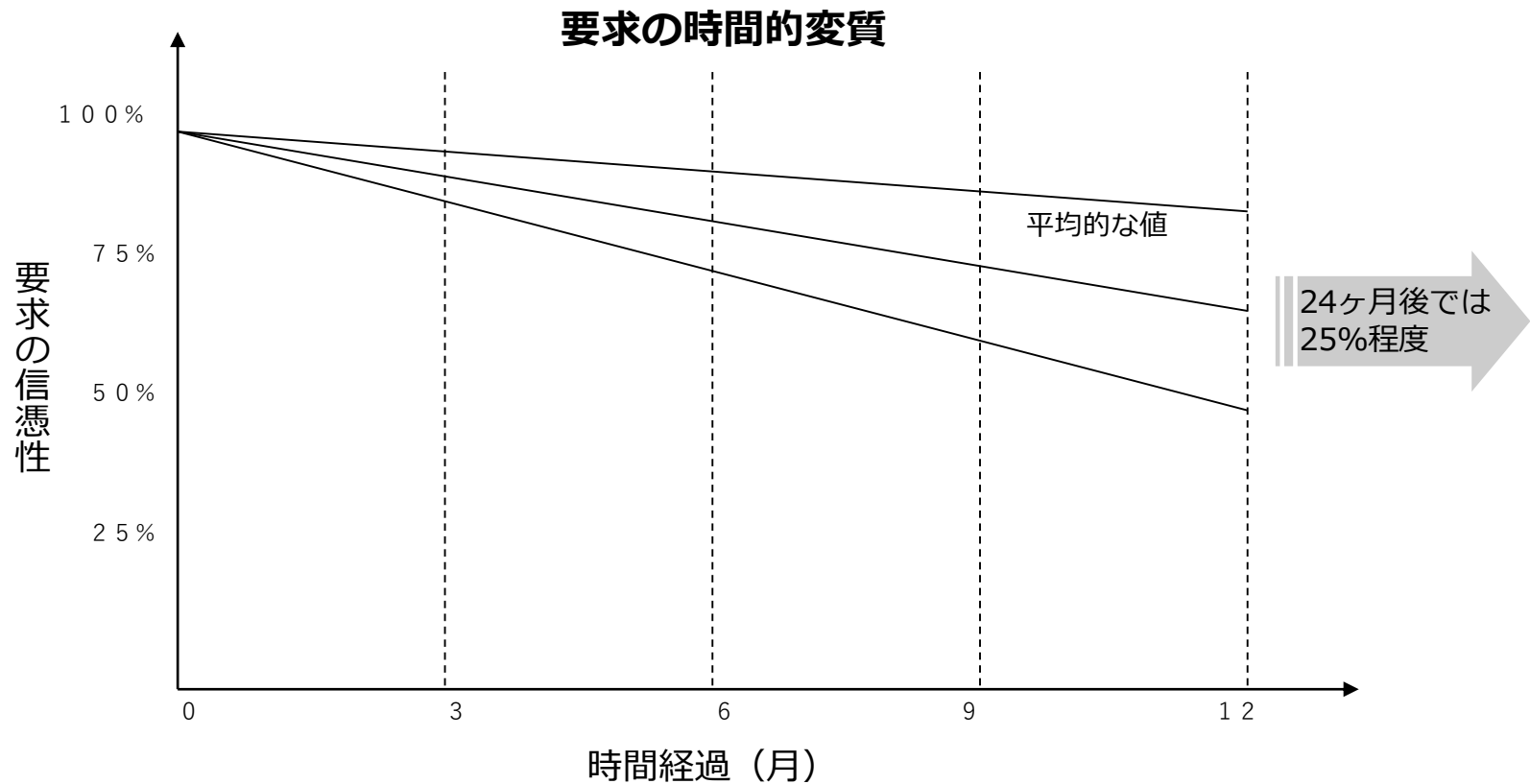
DSDM, Atern(アターン)の開発アプローチ



要求の変質

迷信：要求を明確に定義できれば良い

現実：要求は時間と共に変化



演習-1 アジャイルの価値 (パラダイムシフト)

12のアジャイル原則を理解し、

『アジャイル開発に移行するために、現状から変えなければならない事』
を上げてください。

ワーク 1

チームで30分間討論して、重要度(優先順位)の高い順に表記してください。

ワーク 2

チームの結論を5分間で発表してください。

第3章 スクラムのプロセスと役割

- スクラムのプロセス
- プロジェクト管理
- スクラムの役割（登場人物）
- 《WS》 ユーザーストーリー（演習）

スクラムのプロセス

スクラム (Scrum) の特徴

軽量なアジャイルプロジェクト管理メソッド

小規模で自律的なチーム

完全な透明性


変更への敏速な適応

- 小さな多数のチーム同士の密接に協力と透明性の高い環境で、反復的にインクリメンタルな製品を公開する
- チームは自己決定権を持ち、それぞれの反復の目的を満たす計画を行う
- スクラムマスターがファシリテーションを行う
- 活動はプロダクトバックログによって組織化される

スクラム (Scrum) の基本

- **Boundary / 境界**
 - スプリントやミーティングは、時間的制約を持つ = Timebox
開発チームのリズムを作る
- **Goal / 共有された目標**
 - 透明化による目標の共有
- **Fixed Team / 固定されたチーム**
 - チームメンバーは専任かつ固定化
少人数で、各々がさまざまな仕事を行う

スクラム・プロセス



プロダクトバックログ

- 1.-----
- 2.-----
- 3.-----
- 4.-----
- 5.-----

プロダクトオーナー
(お客様)

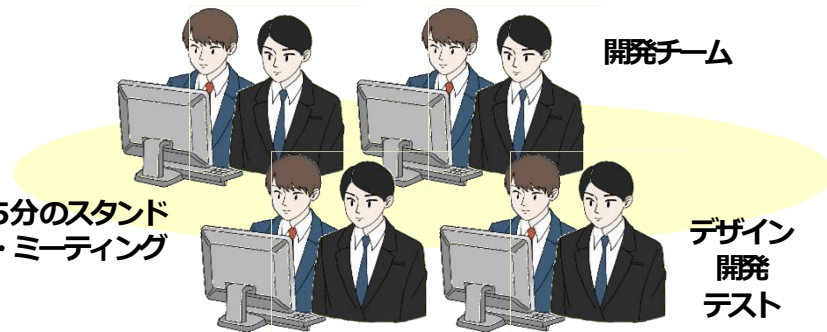
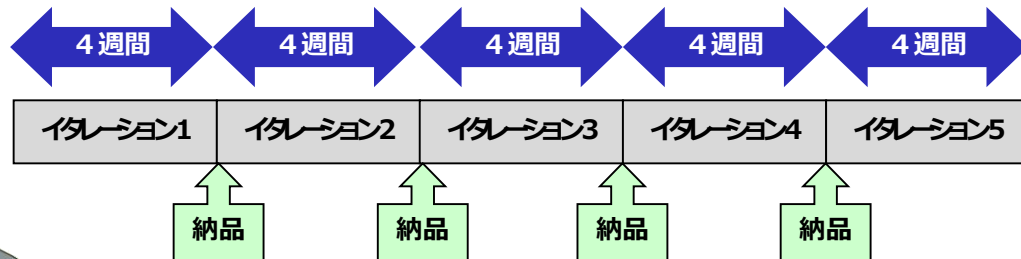
確実な優先順位付け
(同位は無し)
追加、修正、変更可能
(スプリントに入っている物以外)
プロダクトオーナーが
全て責任を持つ

スプリント・プランニング

Part1	1時間
Part2	2時間



スクラムマスター
(ファシリテーター)



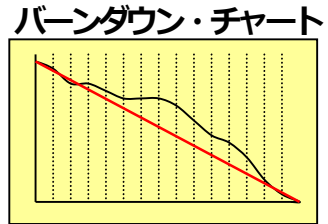
タスクボード

To Do	On Going	Done
c	b	a
d		
e	g	f
h		

Sprint Task List
(Sprint Backlog)

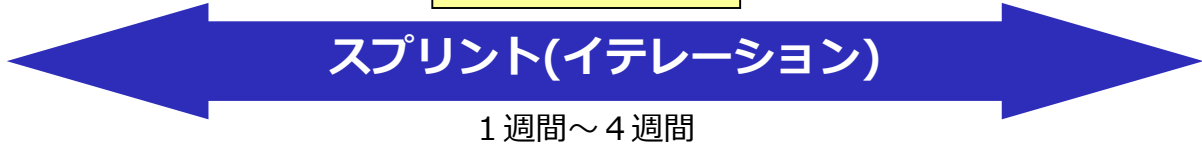
- a.-----
- b.-----
- c.-----
- d.-----
- e.-----
- f.-----

デイリースクラム
(スタンドアップ・ミーティング)



スプリントレビュー
1時間

レトロスペクティブ
(振り返り)
1時間



スクラム (Scrum) のプロセス

1. プロダクト・バックログ

要求を優先度の順にまとめる

2. スプリント計画

- ① プロダクト・バックログを元にスプリントの目標設定をする
- ② 開発チームのタスクを決定する

3. スプリント・バックログ

スプリント計画で決めたタスクをリスト化する

4. スプリント

チームの作業期間

5. デイリー・スクラム

毎朝の進捗報告ミーティングで具体的な作業報告を行う

6. スプリント・レビュー

スプリントの成果を報告する

7. スプリント・レトロスペクティブ

毎週行われる進捗報告ミーティングにて、KPTをチーム間で共有する
(KPT … Keep/継続 Problem/問題 Try/挑戦)

プロダクト・バックログ

- 要求を直感的にわかりやすくリスト化
- プロジェクトにつき1つのみ
- 優先順位に並べる
 - ユーザーにとっての価値やリスクを考慮
 - 実装の及第点も示す
- プロダクト・オーナーが内容や優先順位の保守と支持の責任を負う
- 要求の変動や成果のにより、常時更新され内容が変化する
- 誰でも提案できる、全員から常に見えるところに提示される

ユーザーストーリー

要求が適切に理解されずに、そしてプロジェクトのステークホルダーとコミュニケーションが取れない限り、どのようなソフトウェア/システム開発も成功に至ることはありません。

- Karl E. Wiegers

ユーザーストーリーは要求を具体化する方法の一つ

ユースケースやシナリオと共にシステムの利用法に基づいたアプローチ（ユーザー視点のシステムの振る舞い）によりユーザーの要求を理解し記述するための技法

簡潔な文書による記述、さらに詳しい情報を得る会話、そしてこれらの詳細を確認するためのテスト（受入条件）から構成される

ユーザーストーリー (ビジネス分析のツール)

ユーザーストーリーにより、

工程や作業の意味を明確にする

作業や効果を明快に説明できるようになる

開発者とユーザーの会話のきっかけを作る

開発手法に依存せず、専門知識をも不要である

→ ビジネス分析・改善のツールである

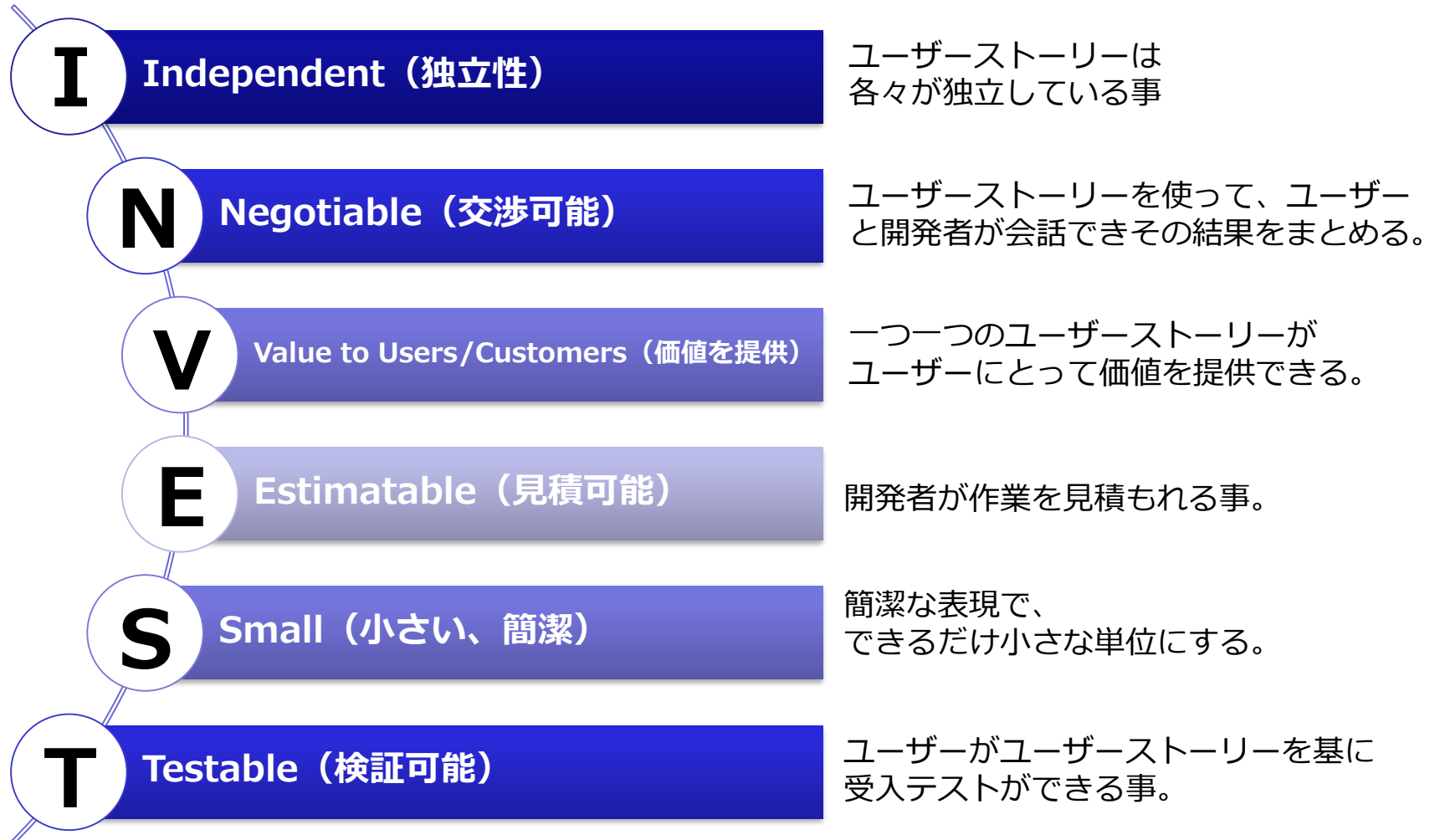
「役割・利用者 (As a role)」

「機能 (can) / 要求 (need)」

「ビジネス価値 / 効果 (In order to)」

の3つに分けて表現される

簡単なルール INVEST



ユーザーストーリーの表記法（書き方）

<利用者の役割>として、

私は、**<機能や性能（システムが～できる事）>**が必要だ。

それは、**<理由（ビジネス価値）>**の為だ。

(例)

<役割> 予約受付係りとして、

<機能> 私は年間最繁忙期でも最低12件の旅行予約を60分以内に処理できる。

<価値> それは予約電話の不受信を最小限に収めるためだ。

ユーザーストーリー・カード（例）

User Story Card ID:

要求者名:

部門名:

役割 〈As a Roll〉:

機能 〈I/We can〉:

環境・条件
〈Which I need〉:

効果・価値 〈To〉:

日付: 0000/00/00 記入者:

← 表面

確定要求		
未確定要求【仮説・推測・期待・その他】		
Body Process	Altanative	Option
【コメント欄】		

裏面 →

スプリント・プランニング

パート 1

プロダクト・オーナーと開発チームにより次のスプリントのどのような機能を構築するか（実装内容）を決定する。

手順

1. プロダクト・バックログの提示
2. 次のスプリントで対象とするバックログの選定
3. スプリント目標の設定（バックログの実装を通じて満たされるべき目標）
4. ストーリーの確認

スプリント・プランニング

パート 2

開発チームのプロダクトへの実装方法の決定

手順

1. ストーリーの作成
2. リリース内容と予定の決定・報告
3. スプリント・バックログ項目（タスク）の定義
4. タスクの作業時間見積り

タスクへの分割と定義

スプリント・バックログ（タスク）

スプリント目標達成のためのタスクのリスト

タスクは、要求の実装に必要な作業を細分化したものの

作業時間数によりタスクの見積もる

各メンバーが担当するタスクを選択する（割り当てではない）

スプリント中、チームのメンバーにより更新される

スプリント・バックログを変更できるのはチームだけ

スプリント・レビュー

スプリントでの成果を、管理者、顧客、ユーザ、プロダクトオーナーへ提示
管理者、顧客、ユーザ、プロダクトオーナーは、成果を査定

特徴

- 次のスプリント計画のため、全員がプロダクトへの追加要素を理解する
チームは製品について具体的に説明
質疑応答や議論を行うが、批判などは避ける
- ミーティング準備に時間はかけず、プロダクトを用いて端的に機能を紹介
一般的なプレゼンテーションツールは使わずに、機能するソフトを発表する。

レトロスペクティブ（振り返り）

毎週行われる進捗報告ミーティング

チームからKPTの共有

Keep / 継続 Problem / 問題 Try / 挑戦

特徴

- チームによる自主運営で、管理職は参加しない
- 透明性の為、幅広く自由な会話を行う（仕事での問題からチームの環境まで）
- 会議後チーム・リーダーが管理者へ報告、要望、依頼を行う
(管理者は間接的に要望解決を図る)

プロジェクト管理

自主管理を行う = 外部から状況がわからない

→ チーム内外でのコミュニケーションにより透明化を行う

チームが行うべき事

- チームの規約の作成
- 共有をしやすい環境作り
- リソースの管理
- スキルの管理
- コミットメント

チームの規約例

- スプリントスケジュール
スプリント計画ミーティング
振り返りの設定
- 一日のスケジュール
スタンドアップ・ミーティング
タスクを取る最終時間
インテグレーションの時間
残業
- コードの基準 / リファクタリング
- ペア・プログラミングの実施ルール
- テスト駆動開発 (TDD)
- タスク終了の基準 (定義)

スタンドアップ・ミーティング

- 毎日チーム全員で行う短時間（15分程度）の状況確認ミーティング
- 継続と透明化を重要視する
- 「前日の作業結果」「当日の作業予定」「課題」を全員が同じ順に報告
- 必要に応じてフォローアップ・ミーティングを行う



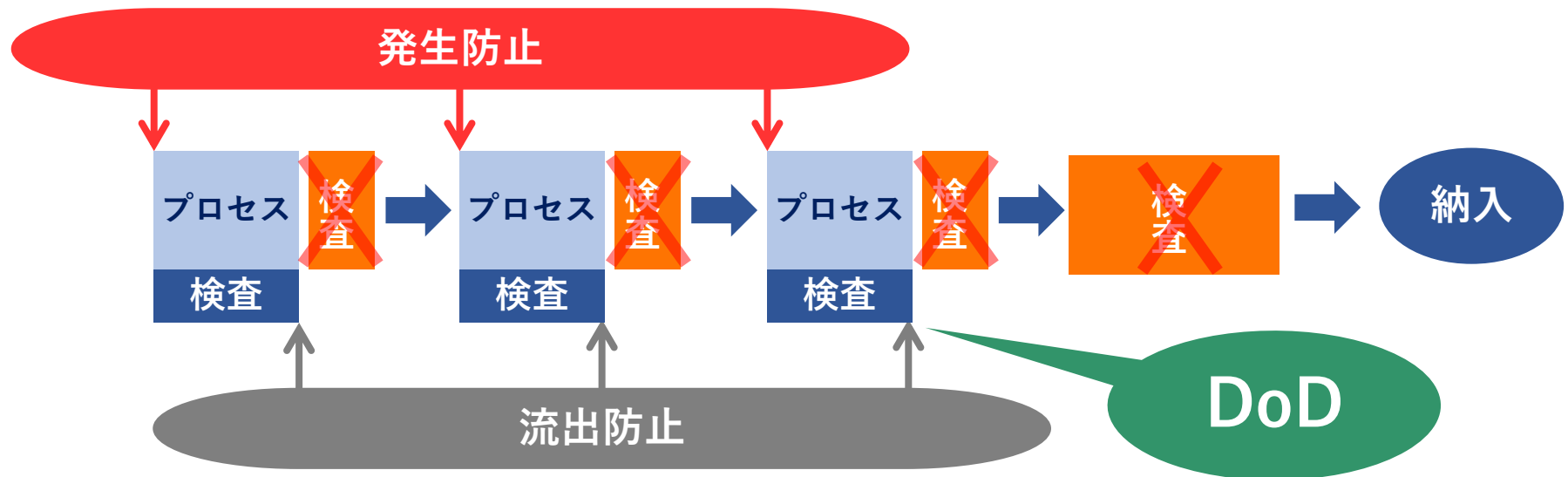
朝礼ではない！

DoD (完了基準)

品質管理の基本

不良をつくらない！
不良な作業をしない！

- 各作業の完了基準
- 閾値の決定
- 作業経過、結果を計測
- 自工程完結の基本的な姿勢
- 他人の作業を肯定的に捉える



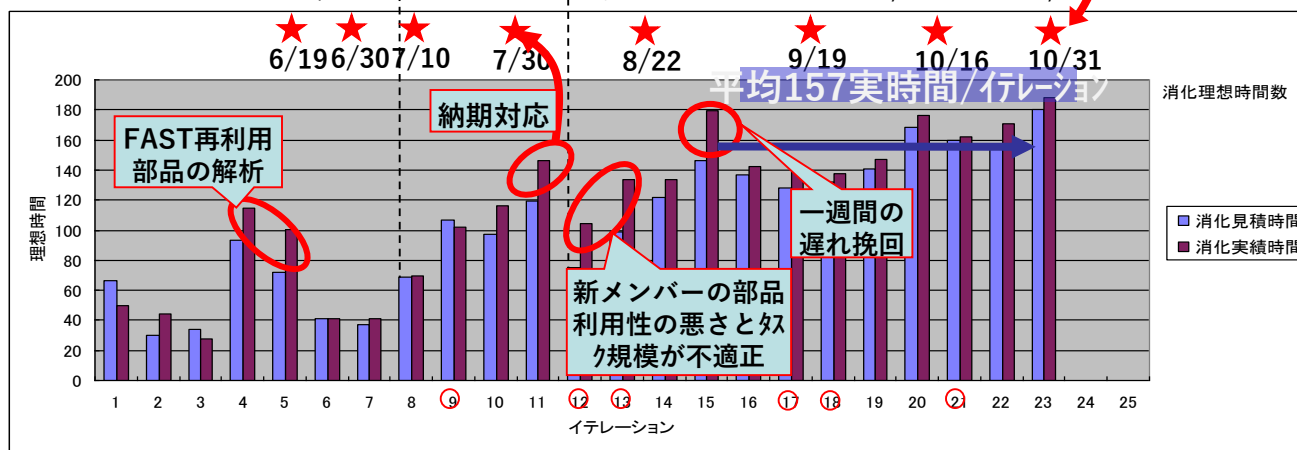
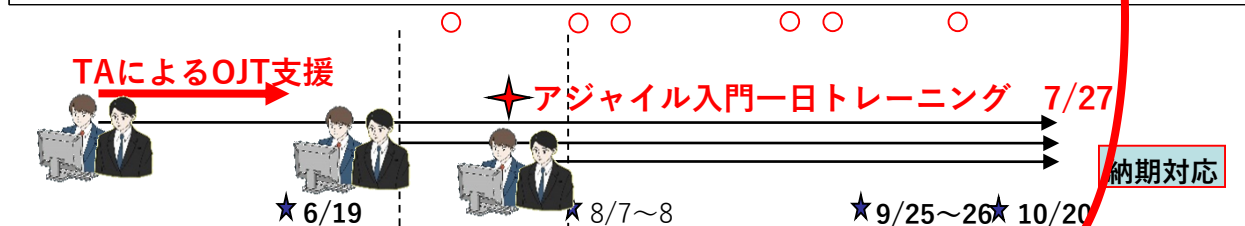
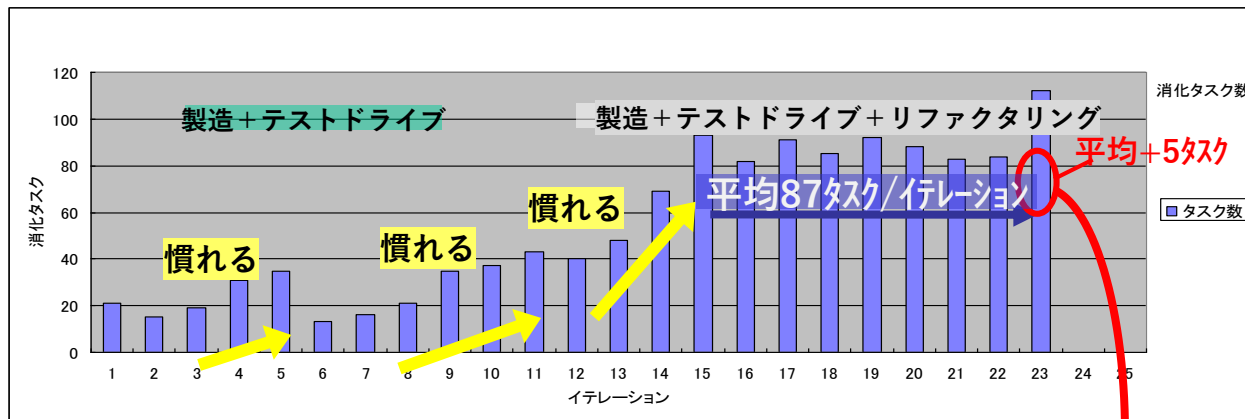
バーンダウン・チャートとベロシティ

[凡例] ★:合宿 ★:リリース ○:実質作業日が5日未満

ベロシティ

チームの作業速度を表す数値

- 1回のスプリントで開発できる量
- 毎週の実施タスク、見積時間、実施時間を収集し平均生産高を見積もる
- ベロシティを一定に保つことが重要



コミットメント

スプリントの終了時に、インクリメントを提供
各スプリントのチームの最終目標
これに対し、チームには説明責任が生じる

最善を尽くす!

スクラムでの役割 (登場人物)

スクラムの形態に必要とされる役割は主に以下の3種類

- プロダクト・オーナー
- スクラム・マスター
- 開発チーム / メンバー

プロダクト・オーナー

役割と責任

- プロダクトの機能と特徴を定義
- リリースの内容と日程の管理・決定
- プロダクトの収益の見積もりや投資の意義の提示
- 実装する機能の優先順位付け
- チームの作業の成果の認証

スクラム・マスター

役割と責任

- チームの機能や効率化、自律を間接的に支援
- チーム開発環境の整備 / モチベーションの向上
- ユーザーへのチームの能力の売り込み
- プロジェクト関係者間の信頼の向上
- ジャスト・イン・タイムの徹底
- ユーザー第一 / 要求に沿った改善の推進

開発チーム

役割と責任

- プロダクトの製作への責任 / リリース日時の徹底
- 4人～10人の範囲で構成
- 一人多能で柔軟性を重視
- 規則に従った上で、各自が自由に開発を進める
- 目標達成のため自ら作業改善を行う
- 争議はチーム内で解決する
- 作業規約を作る

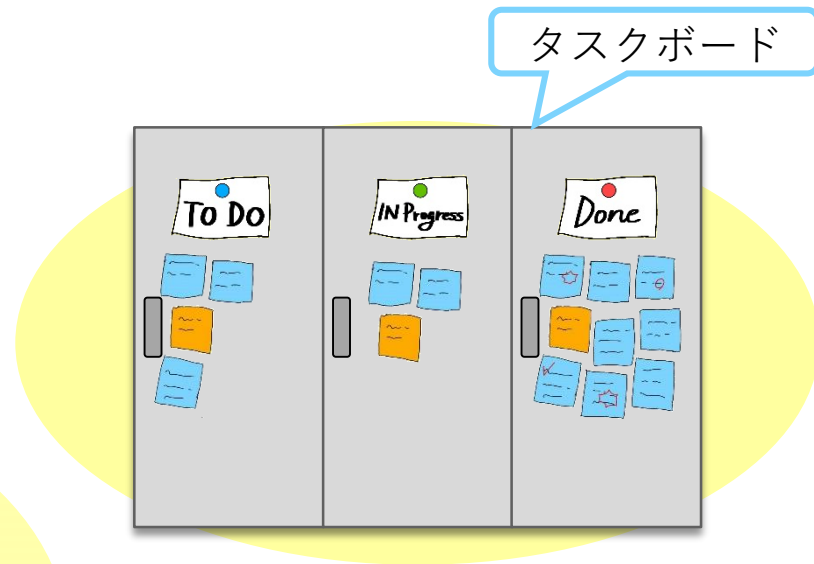
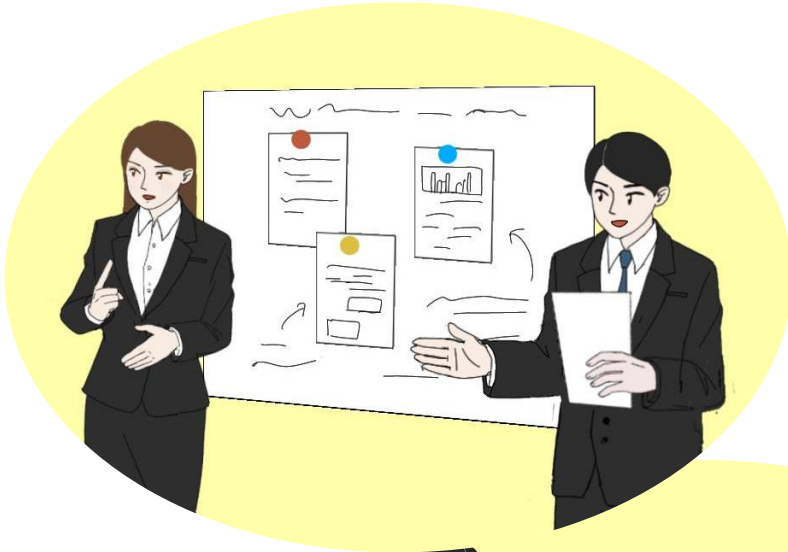
スクラムの環境

- 透明性の高い作業環境
区画の無い共有作業スペース
→コミュニケーションのとりやすさを意識



スクラムの環境

スタンドアップ・ミーティングの風景



第3章クイズ (1)

(問) アジャイル開発チームのチーム編成について説明した文章で適切なものを選びなさい。

1. 開発チームとは異なる顧客側組織のメンバーであるプロダクトオーナーは、開発チームと定期的にコミュニケーションをとるための会議を調整する。
2. 開発チームは、仕切りのない部屋で作業をし、常にフェイス・トゥ・フェイスでの会話行うべきである。
3. 相当のスキルと経験が必要なアジャイル開発のメンバーに、経験の浅いメンバーは採用すべきでない。
4. 開発チームの納期を守るために、スクラムマスターはチームメンバーに的確な指示を出す必要がある。

第3章クイズ (2)

(問) アジャイル開発のスプリント（イテレーション）について説明した文章で適切なものを選びなさい。

1. スプリント計画会議で計画した実施予定のタスクが終了しない場合、プロダクト・オーナーと協議をしスプリントの期間を延長する。
2. アジャイル開発は変化に柔軟である必要があるので、スプリントの期間内でもストーリーの追加や変更を受け入れるべきである。
3. スプリント期間を短ければ短いほど効果的なため、スクラムマスターはチームに短くなるように指導し続けるべきだ。
4. ビジネス価値の探索というスプリントの目的のため、スプリントの最後には動作する成果物をビジネスの関係者にデモを行いFBを受ける必要がある。

第3章クイズ (3)

(問) アジャイル・プロジェクトの初期作業について適切なものを選びなさい。

1. プロジェクトに必要なタスクをリスト化し、最初のスプリントの計画を実施する。
2. 開発チーム、プロダクト・オーナーなど利害関係者の間で、プロジェクトの予算や期間、品質などの制約の優先順位について合意を得る。
3. 開発チームを構成する多様で優秀な人材の経験を重視して、認識合わせは最低限にする。
4. 開発に必要な技術や開発環境の準備は、その都度スプリントの期間内で決定する。

第3章クイズ (4)

(問) 振り返りを実施するタイミングとして適切なものを選びなさい。

1. 問題が生じるたびに、振り返りの実施をスクラムマスターに提案する。
2. 振り返りはスプリントのレビュー会議と一緒に実施すると効果的なので、スプリントの最後に必ず実施する。
3. 振り返りはスプリントの期間と関係なく毎週定期的に行う。
4. 振り返りの実施毎に、話し合いで、次回の振り返りの日程を決める。

第3章クイズ (5)

(問) なかなか全員で守ることのできないチームの規約があります。次の中でチームの取るべき対応策として適切なものを選びなさい。

1. 少し気を付ければ誰でも守れる規約なので、スクラムマスターが翌日のスタンドアップ・ミーティングで全員に注意を促した。
2. 全員が守れていないので、振り返りの場で規約から削除することを提案した。
3. 振り返りの場でその問題(Problem)に対し全員で対策 (Try)を考え、次週に全員で取り組むことにした。
4. 少し気を付ければ誰でもできることなので、スクラムマスターがもう少し様子を見ることにした。

演習-2 ユーザーストーリーの作成（1）

『日用雑貨や書籍を販売するショッピングサイト』のシステムを構築しようとしています。

次のようなプロダクト・バックログ（ユーザー要求）があります。

1. 必要なセキュリティを確保する
2. 画像付の商品カタログを提供する（画像は正面、物によっては背面、側面もある）
3. 在庫している商品を販売する（手配品は今回の範囲外とする）
4. 注文は簡単な操作で行なう
5. 欠品している商品は「在庫なし」と表示する
6. 欠品している商品は注文を受付けない
7. 注文確定時に在庫を引き当てる（在庫管理システムに必要情報を提供する）
8. 在庫が引当てられた場合には受付番号を表示する
9. 在庫が引当てられた場合には注文請書のメールを送信する
10. 引当て時に欠品した場合にはメッセージを表示する（受注残は管理しない）
11. 支払方法は、クレジット、代引き、銀行振込を選択できる
12. 銀行振込の場合には注文請書のメールにて振込先の情報をお知らせする
13. 代引き手数料は350円、送料は全国一律480円とし2万円以上お買い上げの場合は無料（当方負担）とする
14. 初回のお客様には顧客情報を登録して頂く
15. 初回のお客様には顧客情報登録後、ユーザーIDを発行する
16. 出荷業務、配送業務は物流事業者に委託する（出荷管理システムに必要情報を提供する）
17. 銀行振込の場合には入金を確認してから商品を出荷する
18. 発注警告情報、欠品情報などの在庫管理情報をサイト運営担当者に提供する
19. 売れ筋情報・死に筋情報などの売上管理情報をサイト運営担当者に提供する

演習-2 ユーザーストーリーの作成（2）

プロダクト・バックログからユーザーストーリーを作成してください。
（できるところまでで、結構です。）

ワーク 1

演習時間： 60分

ワーク 2

発表時間： 各チーム5分

第4章 XP概説

- XPの価値感
- 手法の紹介

XP（エクストリーム・プログラミング）

欠陥（バグ）のないプログラムの作成手法（自工程完結の思想の導入）

- Kent Beckにより1999年に発表される
- テスト駆動開発 / Test-Driven Development
 - ① 機能をテストするコードの作成
 - ② テストで問題がなくなるまで修正を加えながら、コードを実装
- リファクタリング
機能に変更を加えず、完成したコードの単純化・明確化の為の任意作業
- ペア・プログラミング
ドライバー(実装担当)とナビゲーター(確認担当)を交代しながら作業
毎日ペアを変更
- 10分間ビルド
自動的にシステム全体をビルド
全てのテストを10分以内に実行

XPの価値感

1.コミュニケーション

多くの問題の原因

- ・エンドユーザー、プログラマー、マネージャー相互間
- ・システム要件
- ・設計の意図
- ・陣営、トラブル
- ・進捗

XPの多くの実践はコミュニケーションなしには行えない

- ・ペアプログラミング
- ・チーム全体など

2.シンプル

今日、シンプルな実装をする。

- ・明日、必要なもの1まそのとき実装する。
 - ・今日実装した複雑な機能は、明日使わないかもしれない。
 - ・システムがシンプルなら、コミュニケーションはしやすい。
- 最もシンプルなコミュニケーション方法は？
- ・ユーザとの直撮会話によるコミュニケーション

4つの価値

3.フィードバック

株動するシステムがもっとも情報量が多いフィードバック

- ・要求：ユーザーにシステムを操作してもらいながら
- ・段計：実際のコードを見たり、機能追加をしてみる
- ・品質：線助しているシステムに対してテストを行う
- ・進捗：助作している機能を見る／実績をみる

4.勇気

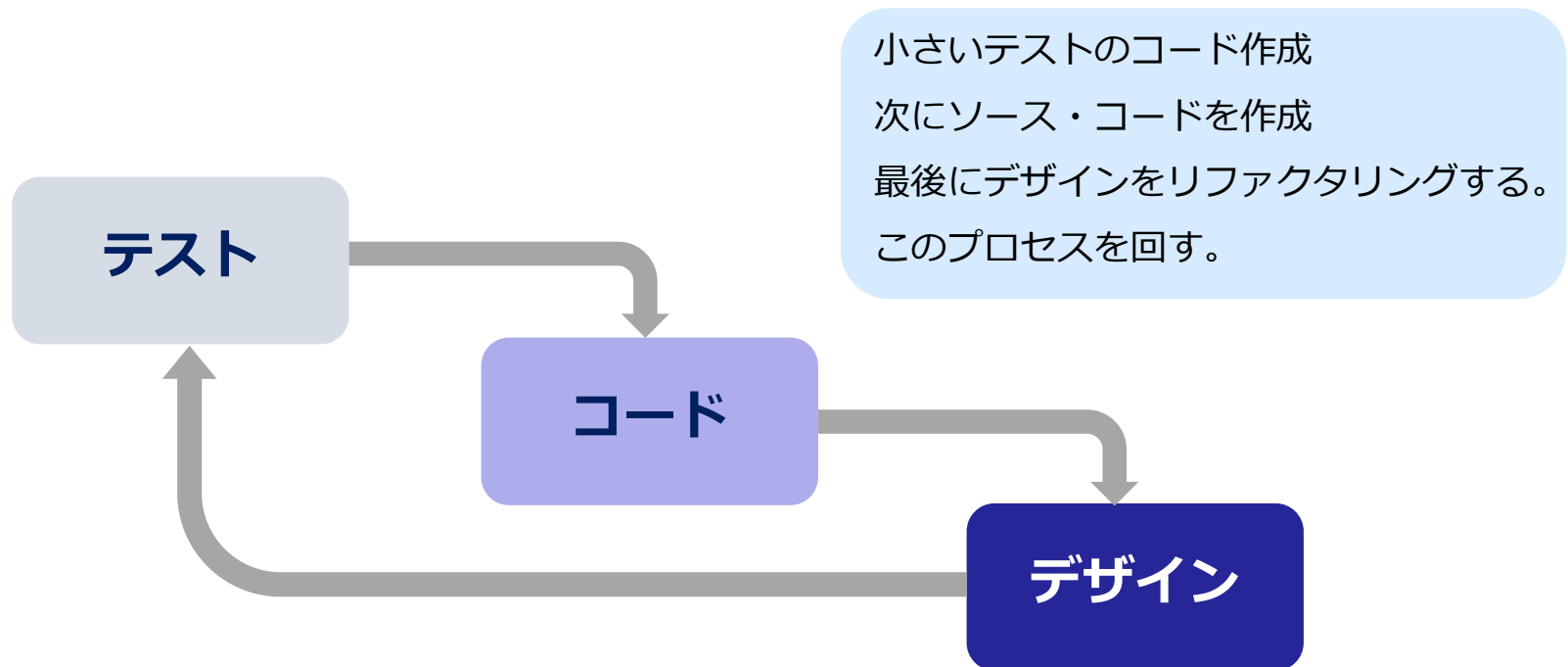
コミュニケーション、シンプル、フィードバックと結びつくと、勇気は極めて価値のあるもの

- ・欠陥の修正、コードを捨て去る勇気
 - ・システムをシンプルに保つ勇気
 - ・嘘のないコミュニケーションを行う勇気
 - ・メンバーからのフィードバックを受け入れる勇気
- 人がもっとも必要な要諦であると考えている証

手法の紹介 – テスト駆動開発 (TDD)

•テスト駆動開発

- テストの自動化
- アクセプタンス・テスト (完了・終了の定義)



リファクタリング

特徴

動作確認後、ソースコードに対し以下を実行する

- 可読性の向上やクラス化・単純化
- 使用の容易さ、誤作動抑止などの仕様レベルでの調整

ルール

- 同時にリファクタリングと機能改変を行うことの禁止
- 変更前後の機能変化がないように留意
- メンテナンス性を重視
- 一回の変更は小さくし、繰り返し変更を行う

リファクタリング基本方針

- コードをできるだけ小さな単位に分割する
 - 処理のかたまりに名前をつける
 - 処理を重複して記述しない
 - コードをできるだけ小さな単位に分割する（部品が小さいほうが理解しやすく、バグも少なくなる）
- また、コードを小さく分割することで、そのまとまりごとにメソッド名やクラス名などの名前を付けることができます。
- 実は、この名前がコードがわかりやすくなるかどうかの分かれ目になります。コメントを充実させるよりも、リファクタリングによってコードの断片をまとめ、適切な名前を付けるべしというのがリファクタリングの教えです。
- （もちろん、実際にはコメントも重要ではありますが。）
- そして最後に重要なのが、コードの重複をできるだけ少なくすることです。重複コードを見つけたらまず間違いなくリファクタリングの対象になります。
- 重複するコードがあるということは、プログラムの変更時に変更しなくてはならない箇所が分散してしまい、変更時のバグが入り込みやすくなってしまうということです。それを防ぐためには、重複するコードをできるだけ少なくするというのはコードの保守性を高めるためにきわめて重要な作業となります。
- ただし、あまりに共通ルーチンを使い回すと、逆にメンテナンスのしやすさを低下させてしまうこともあることを覚えておいてください。
- あまりに多くのモジュールから、ひとつの共通ルーチンが参照されているときは、その共通ルーチンを変更するときの影響が大きすぎることになってしまいます。
- このへんのさじ加減を広い視点で判断し、コードの保守を行っていくことが重要です。

ペア・プログラミング

特徴

- 2人で1台の端末を利用し、プログラムを実装する
- ペアは毎日変更
- ドライバー/実装担当とナビゲーター/確認担当に分かれる
- 役割は一定時間で交代
- 品質と可読性の向上
- 共同作業による思考時間の削減



ペアの交替（定期的：毎日）
作業の交替（一定時間毎）



10分間ビルド

特徴

- 自動的にシステム全体をビルド
- 全てのテストを10分以内に実行
- ビルドの対象となるシステムとテストの対象になる部分の把握が重要

テストの自動化

「各々のスプリントで、テスト・カバレッジの小さな向上を目指す」

カバレッジの向上を行う方法

- テストケースを一覧の作成
- テストケースのリスクによる分類
- 優先順位をつけて並べ替え
- テストの部分的な自動化

それぞれのスプリント内で、
一つはテスト自動化のストーリーを実装する。

第4章クイズ (1)

(問) リファクタリングについて適切なものを選びなさい。

1. アジャイル開発のための手法のリファクタリングは、ウォーターフォール開発では利用できない。
2. リファクタリングは特定のタイミングでまとめて実施する方が良い。
3. レガシーコードでテストコードが存在しない場合は、リスクが高いためリファクタリングを避ける。
4. リファクタリングによりユーザーにもメリットが生じる。

第4章クイズ (2)

(問) ペア・プログラミングの説明で適切なものを選びなさい。

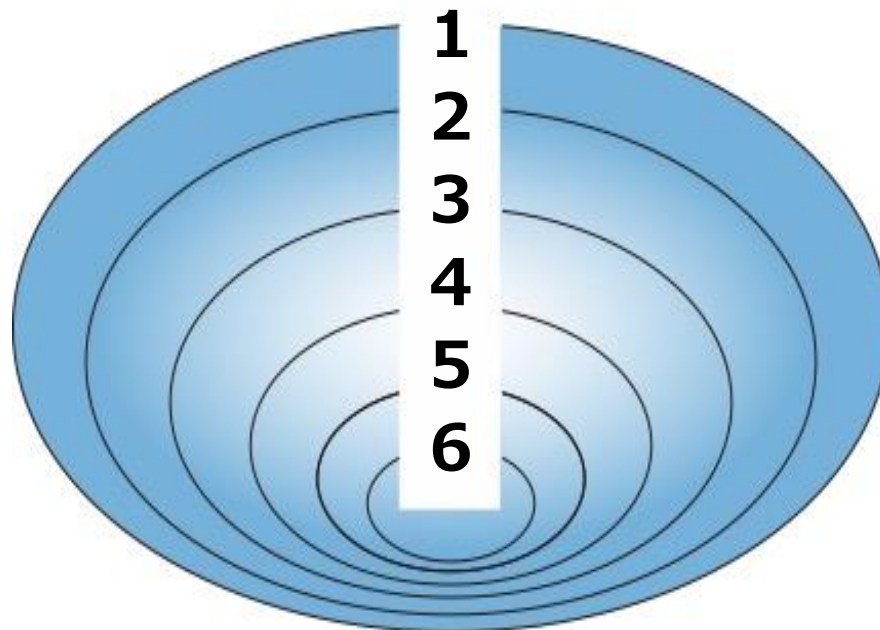
1. ペア・プログラミングを行うときは、休憩をこまめに取得させるようなルールが必要である。
2. 高いスキルのメンバーにとっては生産性を下げるため、スキル不足のメンバー間でのみ採用する。
3. ドライバーは、ナビゲーターの指示に従ってソースコードを作成するので、ドライバーは自主的なソースコード設計を行わない。
4. ペアを頻繁に替えるとコーディング対象の機能についての情報を何度も確認する必要が出るため一つの機能のコーディングが完了するまでペアの役割を替えるべきではない。

第5章 アジャイルな計画

- アジャイル開発で用いる計画手法
- プランニング・オニオン / 2レベル・プランニング
- 《WS》タスク分解（演習）

アジャイルな計画づくり

プランニング・オニオン
包含関係の強い計画群



1. 戦略
2. ポートフォリオ計画
3. プロダクト計画
4. リリース計画
5. スプリント計画
6. デイリー計画

アジャイルチームが対象とするのは、リリース計画、スプリント計画、デイリー計画

反復性増加型開発の計画



- 2レベル計画
 - リリース計画…全体の大まかな計画
 - スプリント計画…作業に対する詳細な計画
- 連続的な計画
- 実測駆動によるベロシティの測定

→ 作業の見通しを、より早く行う

リリース計画とスプリント計画

リリース計画

期間

1ヶ月先

対象

ユーザーストーリー

単位

ストーリーポイント等

スプリント計画

期間

スプリント1回分
(1週間先)

対象

タスク

単位

時間

リリース計画とスプリント計画

アジャイルプロジェクトはリリースの繰り返しから、
リリースはスプリントの繰り返しから成る

プロダクトオーナーが責任を持ち、プロダクトバックログを基準とする
ユーザや顧客は、レビューやリリース終了ごとに関わる
顧客のFBをもとに、計画を改善する

全体リリース計画の立案

1. ユーザーストーリーやエピックの規模の見積もり
※エピック … 複数のストーリーの集合
2. スプリントの期間の設定
3. ユーザーストーリー、エピックの優先順位の決定
4. スプリントでのベロシティの見積もり
5. ユーザーストーリー、エピックを配分

直近のリリース計画の立案

1. リリースの目標を確認
2. エピックのユーザーストーリーへの分割
3. ユーザーストーリーの規模を見積もり
4. スプリントの期間の設定
5. ユーザーストーリーの優先順位の決定
6. スプリントでのベロシティの見積もり
7. スプリントにユーザーストーリーを配分
8. リリース日時の決定

リリース計画の立案

反復にユーザーストーリーを配置する



- 1スプリントに期待できるベロシティを基準にする
- 最初のスプリントは詳細に計画
- 二つ目以降のスプリントは変更が生じうるので柔軟に変更
- 各スプリントでのストーリーの数は異なる(ベロシティが一定)

ストーリーの分割

小さい単位であることが求められるストーリーは以下の場合等で分割される

- ベロシティの基準により、ストーリーがスプリントに収まらない場合
- 計画しているスプリントに組み込めない場合
- 正確な見積もりを要する場合

ストーリーの分割のガイドライン (1)

- ユーザーの役割による分割
- データの境界による分割
- 操作の境界による分割

プロセスによる分割

ファンクション(CRUD)による分割

Create/作成 Read/参照 Update/更新 Delete/削除

ストーリーの分割のガイドライン (2)

- 優先度による分割

Ex. オンラインゲームの対戦者→1人の対戦者、2人の対戦者、複数の対戦者

- ビジネスルールによる分割

Ex. 商品の発注→定番品、特注品など

商品の値引→1万円以上、2万円以上、3万円以上など

- 横断的機能の括り出しによる分割

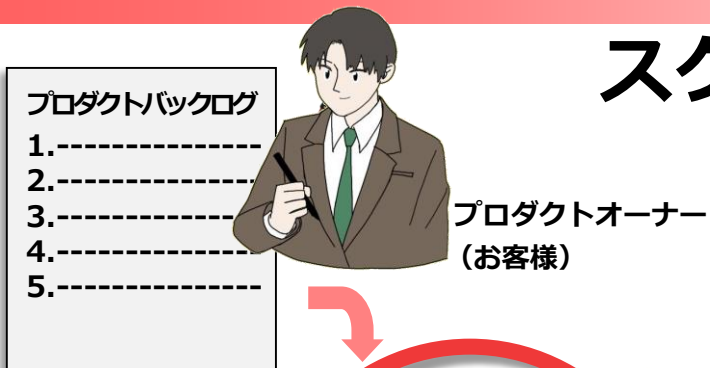
Ex. セキュリティ、エラーハンドリング、ロギング

ストーリーとタスク

タスクの特徴

- ストーリーの実現に必要な作業を細分化したものの
- 作業量が一時間以内となる量
- スプリント・バックログでリスト化
- スプリント中にも書き換えられる

スクラム・プロセス



プロダクトバックログ

- 1.-----
- 2.-----
- 3.-----
- 4.-----
- 5.-----

プロダクトオーナー
(お客様)

確かな優先順位付け
(同位は無し)

追加、修正、変更可能
(スプリントに入っている物以外)

プロダクトオーナーが
全て責任を持つ

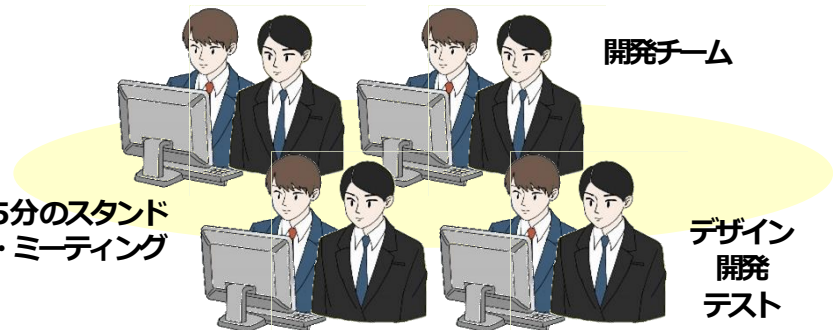
スプリント・プランニング

Part1	1時間
Part2	2時間



スクラムマスター
(ファシリテーター)

毎朝15分のスタンド
アップ・ミーティング



開発チーム

デザイン
開発
テスト

タスクボード

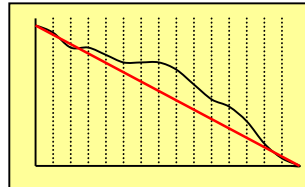
To Do	On Going	Done
c d	b	a
e h	g	f

Sprint Task List (Sprint Backlog)

- a.-----
- b.-----
- c.-----
- d.-----
- e.-----
- f.-----

デイリースクラム
(スタンドアップ・ミーティング)

バーンダウン・チャート



スプリントレビュー

1時間

レトロスペクティブ
(振り返り)

1時間

スプリント(イテレーション)

1週間～4週間

スプリント計画ミーティング

1. プロダクト・バックログ
2. チームの作業能力
3. ビジネス状況
4. 技術、品質の安定性
5. 実行可能な作業量

レビュー
検証
整理

次のスプリント目標
スプリント・バックログ

パート1

開発チームはプロダクト・オーナーと一緒に次のスプリントでどのような機能を構築するかを決定する

- ①プロダクト・バックログの提示
- ②次のスプリントで対象とするバックログの選定
- ③スプリント目標を設定
 - プロダクトバックログの実装を通して満たされるべき目標
- ④ストーリーの確認

パート2

開発チームがその機能をどのようにプロダクトに追加するかを決定する

- ①ストーリーの作成
- ②リリース内容と予定
 - プロダクト・オーナーへ報告
- ③スプリント・バックログ項目(タスク)の定義
- ④タスクの作業時間見積り

スプリント計画の立案 ストーリーをタスクに展開

ストーリー	タスク		
ストーリー1	タスク 1-1 1	タスク 1-2 2	タスク 1-3 1
ストーリー2	タスク 2-1 2	タスク 2-2 1	
ストーリー3	タスク 3-1 1	タスク 3-2 2	タスク 3-3 1.5

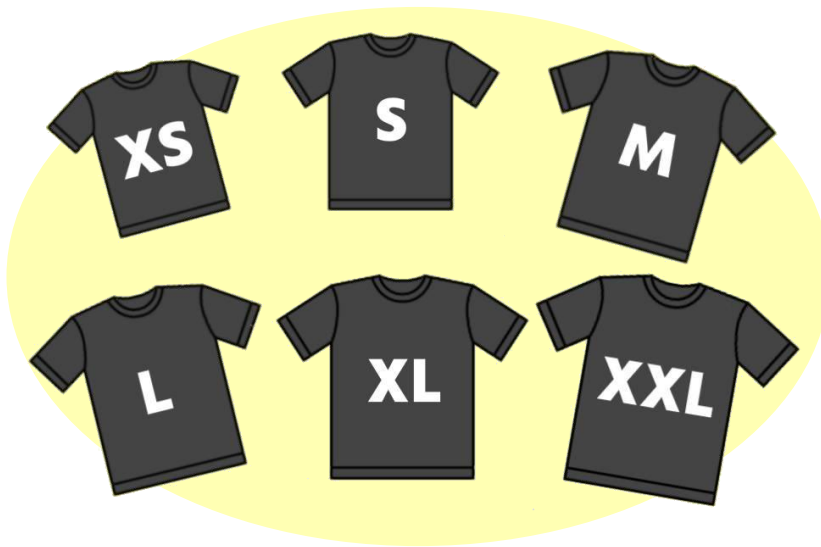
タスクへの分割（タスクの定義）

スプリント・バックログ（タスク）

- スプリントの目標を達成するために行わなければならないタスクのリスト
- タスクは、プロダクト・バックログ（要求）を動作するソフトウェアとして実装するために必要な作業を詳細に分割したもの
- タスクの作業量は時間数で見積る
 - 実測駆動
- タスクは割り当ててるのではなく、メンバーが自ら担当するタスクを選択する
- チームはスプリントを通して必要に応じてスプリント・バックログを更新し続ける
 - スプリント・バックログを変更できるのはチームだけ

タスクの見積

プランニング・ポーカー



相対的に見積もる
実測駆動での見積（前回の実績値）

タスクの粒度

- タスクの粒度を小さくすることで

- 流れを作る

- 負荷を一様にする

- 柔軟性を高める

- 1時間～1日を基準
- テスト実行の容易さを重視
- 作業指示書と同義で、運営が円滑になる
- 作業内容に依存させない

Statements of Source code	1	10	100	1,000	10,000	100,000	1,000,000	10,000,000
Test Cases	1	2	5	15	250	4,000	50,000	350,000
Test Coverage	1	1	1	1	1	0	0	0

Application size, Test Cases, and Test Coverage.
Logical source code statements

By Caper Jones

タスク粒度（例）

タスクの粒度 : 1時間のタスク

タスクの均一性 : 作業内容に関係なくできるだけ同じ粒度

	見積 総時間	実質 総時間	タスク 総数	見積 粒度 (H)	実質 粒度 (H)
第1週～第7週 インテレーション ユーザー操作系が主な機能 1回目	197.8	211.9	134	1.476	1.581
第8週～第16週 インテレーション 機械制御系が主な機能 1回目	418.4	452.2	295	1.418	1.533

平準化すると多様化に対応できる。

全く異なった機能実装
インテレーションでも
同じ粒度で平準化している。

タスクを小さくする

例えば、レポートを作る業務（仕事）をタスク分解する。

1. レポートの主旨の確認	30分
2. レポートの章立てを決定	10分
3. 各章の基本構造を決定	30分
4. 文章の下書き	30分
5. データの収集	30分
6. PC、アプリケーションの起動	5分
7. データのインポート	20分
8. グラフの作成	10分
9. 文章の構成	30分
10. レイアウトの決定	5分
11. レポート全体の校正	30分
12. 体裁の調整	5分
13. レポートの提出	5分
合計	240分

第5章クイズ (1)

(問) アジャイル開発プロジェクトにおける計画作りについて適切で無いものを選びなさい。

1. 計画はストーリーとその優先順位をもとに決定する。
2. タスク単位での計画は、スプリント計画時に立案する。
3. 綿密な計画を遅延やトラブルなく完遂することがプロジェクトの成功の証である。
4. 顧客のビジネス環境の変化によって発生した新たに必要なストーリーや、リストアップされているストーリーの価値に変化に柔軟に対応しながら、計画は随時変更する。

第5章クイズ (2)

(問) ストーリーポイントを付与する際に注意すべき点で適切なものを選びなさい。

1. 類似したストーリーを集め、できるだけ大きな塊で見積もりを行う。
2. 詳細なタスク単位での見積もりは、プロジェクト全体の計画時に行う。
3. ストーリーの実現に必要な全作業を見積もる。
4. チームリーダーやスクラムマスターなど特定の一人が見積もりを行い、ばらつきを回避する。

演習-3 タスクへの分解

『スクラムのプランニング・セッションを実行する (3～4時間)』
と言うプロダクト・バックログをタスクに分解する。

ワーク 1

演習時間： 30分

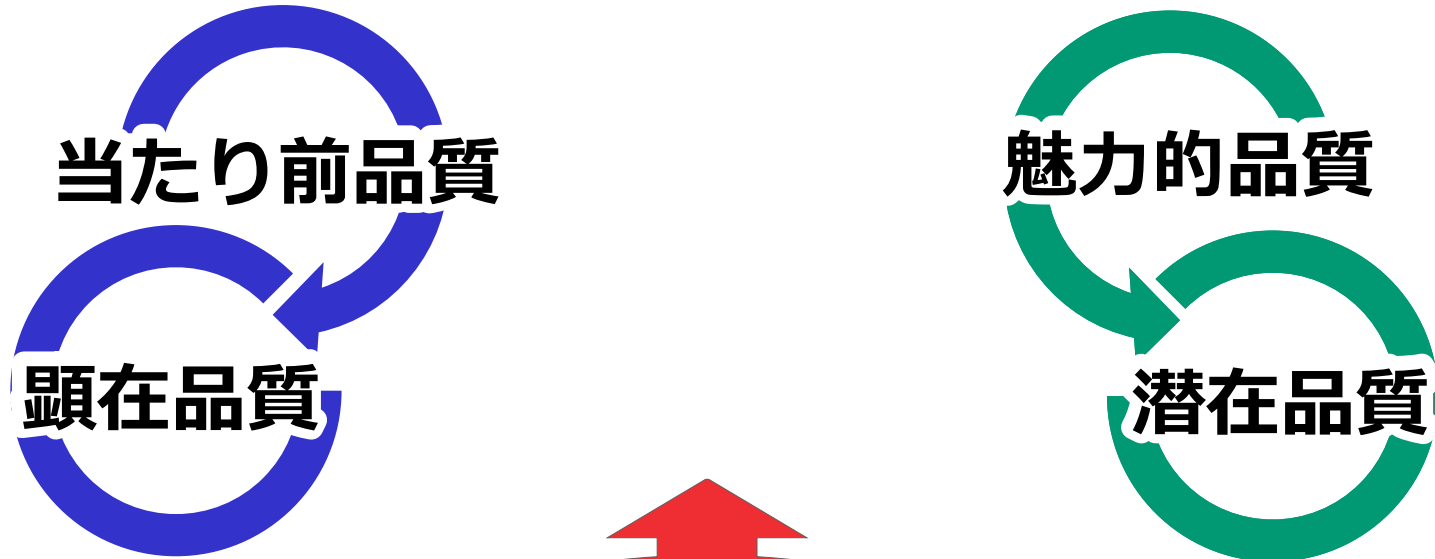
ワーク 2

発表時間： 各チーム5分

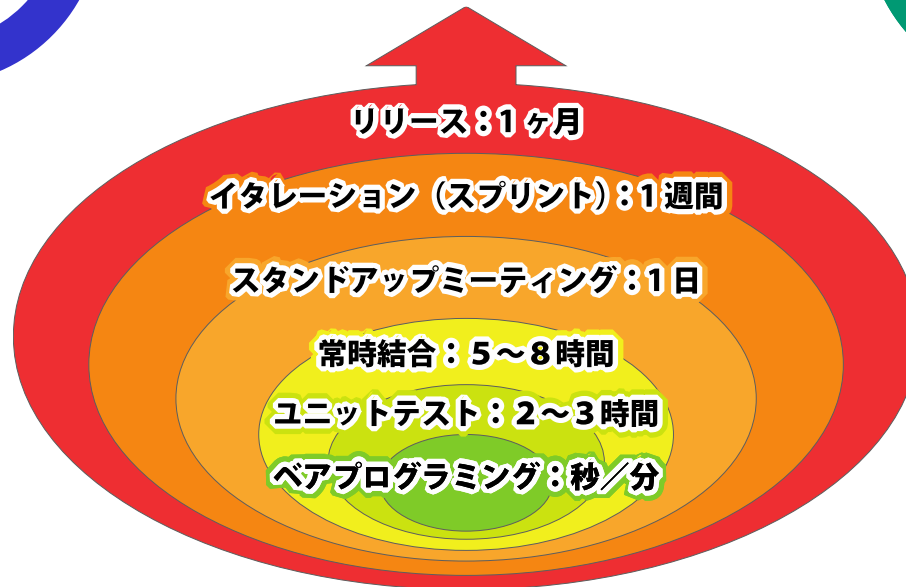
第6章 品質管理

- アジャイル開発での品質
- 《WS》振り返り (KPT) (演習)

品質について考える



- ・ バグフリー
- ・ アジャイル・プラクティスの徹底 (守)
- ・ 頻繁なフィードバック・ループの設計

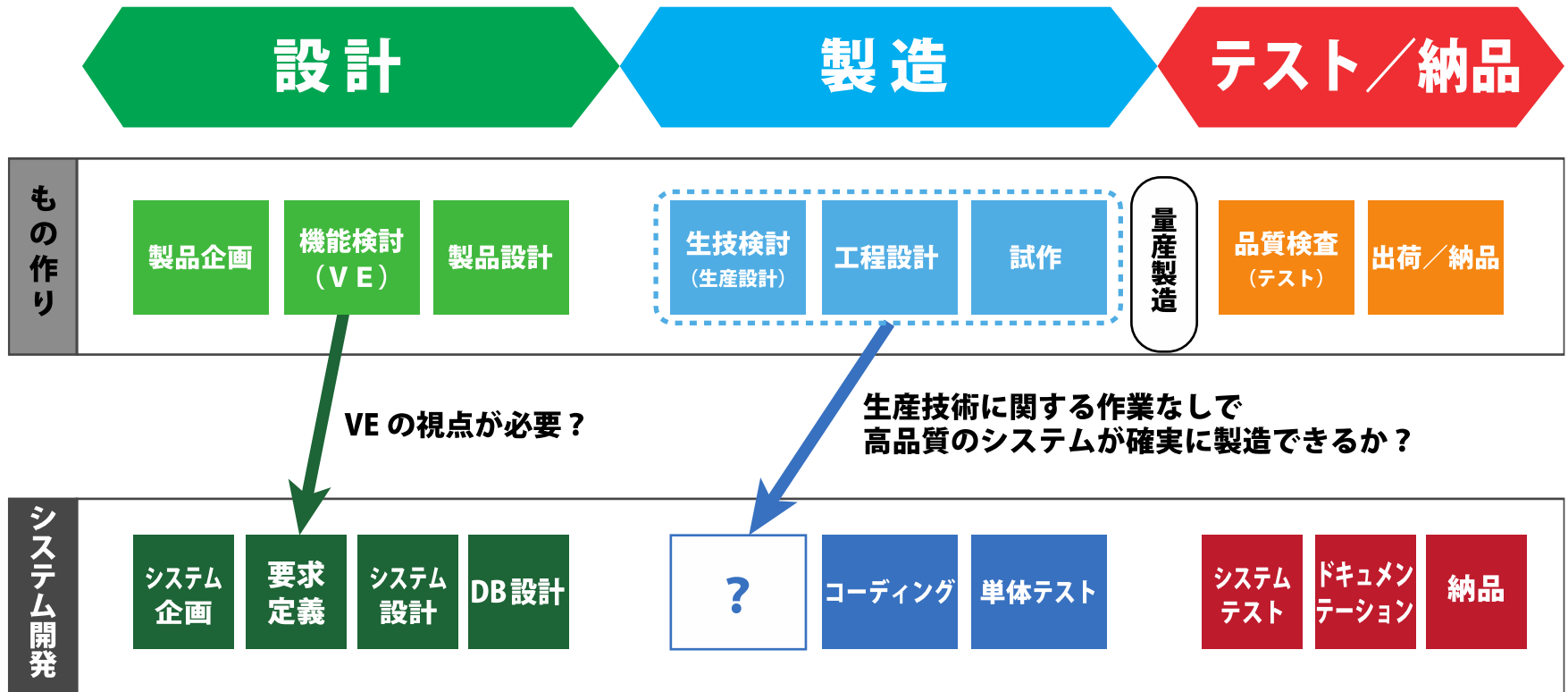


- ・ メンテの容易性
- ・ ソースコード = ドキュメント・メソッド名と変数名 (名は体を表す)
- ・ 簡易に書けるセカンド・ランゲージ (自分のツール作成)

アジャイル開発で品質が向上する理由

- ムダ取りの原則
時間的な境界を明確化 / 作業のムラの減少
- タスクの粒度の極小化
作業の流れを向上 / ミスの早期発見
- 全員の作業の透明化
協力の易化 / 思考時間・修正時間の削減
- 開発の自律性
自工程完結で周囲への影響が減少
全員が他人の作業を把握
- 開発者とユーザの直接的な交流
変更への柔軟性 / 意思の伝達

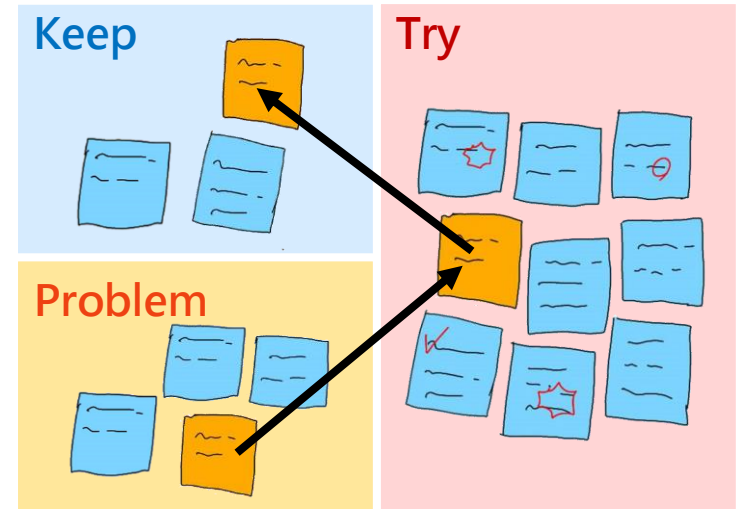
『もの作り』と『システム作り』の相違



設計仕様を図面上のみの検討で十分か？
擦り合わせ、微調整は必要ないだろうか？

KPTでの振り返り（レトロスペクティブ）

- 毎週行われる進捗報告ミーティングで、プロセスの改善を実施
- KPT(Keep/継続 Problem/問題 Try/挑戦)をチームが共有
- 小さなPDCA



小さなPDCAを回すための方法です。

KPTボードは振り返りのツール

- ① 気づきや問題、課題を付箋紙に書き、Problem（問題）のエリアに貼る。
- ② Problemのエリアに出てきた気づきや問題に対してTry（対策）を立ててTryのエリアに移動させる。
- ③ 実際にやってみて、対策が不十分だった場合には、再度、Problem（問題）のエリアに移動し課題を追記する。再度、対策を立て直す。
- ④ Problemに対する効果が十分出たと判断できた場合には、Keep（継続）のエリアに移動し習慣化、定着化を図る。

ここで大事な事は、人を責めずにやり方を責める事です。（誰でも同じようにできる様にする。）

(参考) TQMとは

TQM (Total Quality Management)総合的品質管理とは、トップマネジメントのリーダーシップのもとに組織が一丸となって、お客様が高度に満足する製品を生産したり、サービスを提供するための一連の活動です。TQMは製造業、サービス業など全ての業種において、また小さな組織から大きな組織まで組織の規模に関係なく有効な活動です。

- ◆ 品質の良さはお客様の満足によって測られる。
- ◆ 管理とは目標を決め達成するための活動です。
 - 維持と改善
 - プロセスの良さと結果の良さ
- ◆ 総合的な活動の意味
 - 品質を総合的に考える
 - 組織全体が総合的に参加する
- ◆ 基本原理は、PDCAと継続的な改善
- ◆ 結果を押し込めるのではなく、プロセスで作り込む
- ◆ 後工程はお客様
- ◆ 応急対策と再発防止策の探索
- ◆ TQMの中核はデータで語る
- ◆ プロセスのレベルアップは5S（整理、整頓、清掃、清潔、躰）
- ◆ プロセスの基礎は標準化
 - アメリカは手順重視、 日本は教育重視
- ◆ 改善のステップ
 - 背景の整理、現状の分析、要因の探索（5回の何故）、対策の立案、効果の検証

ソフトウェア製造工程におけるムダの廃除

- **作りすぎのムダ**
不必要・価値のない機能の実装
- **手持ち（停滞）のムダ**
指示・許可待ちや障害などによる作業停止
- **運搬のムダ**
作業の切り替えや重複チェック
- **加工そのもののムダ**
事務作業などの過剰な作業
- **在庫のムダ**
中間的な生成物の残留
- **動作（作業）のムダ**
作業場所や開発ツールの途中変更
- **不良をつくるムダ**
バグの発生、コードの欠陥

仕事を分析する



- 7つのムダ** ①作りすぎ ②手持ち ③運搬 ④加工そのもの ⑤在庫 ⑥動作 ⑦不良を作る

第6章クイズ

(問) アジャイル開発における品質の確保について適切では無いものを選びなさい。

1. 不良を素早い発見と除去を重視し、不良のない状態を維持する。
2. スプリントの成果物の品質について、スプリント開始時にプロダクトオーナーと合意を形成する。
3. 品質管理を重要視するので、進捗を計画通りに進めるための残業はやむをえない。
4. 継続的インテグレーションはアジャイルの品質確保に有効な代表的なプラクティスの一つである。

演習-4 : KPT (振り返り)

ワーク 1	第一スプリント演習	30分
	振り返り	10分
	休憩	5分
ワーク 2	第二スプリント演習	30分
	振り返り	10分
	休憩	5分
ワーク 3	第三スプリント演習	30分
	振り返り	10分
	休憩	5分
ワーク 4	チームの発表 (各チーム5分)	
ワーク 5	アドバイス	
	良い点	
	改善したほうが良い点	

演習課題の選択

6ヶ月後のあるべき姿を定義し、初めの3ヶ月間の行動計画を立案する。

1. フルマラソン（42.195km）を走れる様になり、競技会へ参加できる。
2. メタボなのでダイエットを行う。
3. ゴルフのシングル・プレーヤーになる。
4. 英語が流暢に話せる様になる。（TOIEC x x x点をクリアー）
5. 仕事量は落とさずに残業をしないで、毎日定時に帰宅できるようになる。（ワークライフ・バランスの取れた日常生活）

第7章 大規模プロジェクト

- スクラム・オブ・スクラムの紹介
- 他の大規模プロジェクトの手法紹介

スクラム・オブ・スクラム

大規模なプロジェクトで複数のスクラムチームを構成



プロダクトオーナー



デイリースクラム



スクラムマスター

デイリースクラム



チーム



スクラムマスター

デイリースクラム



チーム



スクラムマスター

デイリースクラム



チーム



スクラムマスター

デイリースクラム



チーム

スクラム
オブ
スクラム

FBI センティネル・プロジェクト

2005年プロジェクト開始

- 予算4億5100万ドル
- 2009年完成予定

2010年3月時点

- 4億500万ドル(予算の90%)を使用、追加予算見積もり3億5000万ドル
- プロジェクトの進捗は50%、完成まであと6年から8年
- FBIの契約スタッフ数220名、FBI職員数30名

スクラムによる解決案

- 予算残額2000万ドル
- 12ヶ月以内で完成
- FBIの契約スタッフ数40名、FBI職員数12名

結果

- 予算の5%
- 20ヶ月で完成

KANBANの特徴

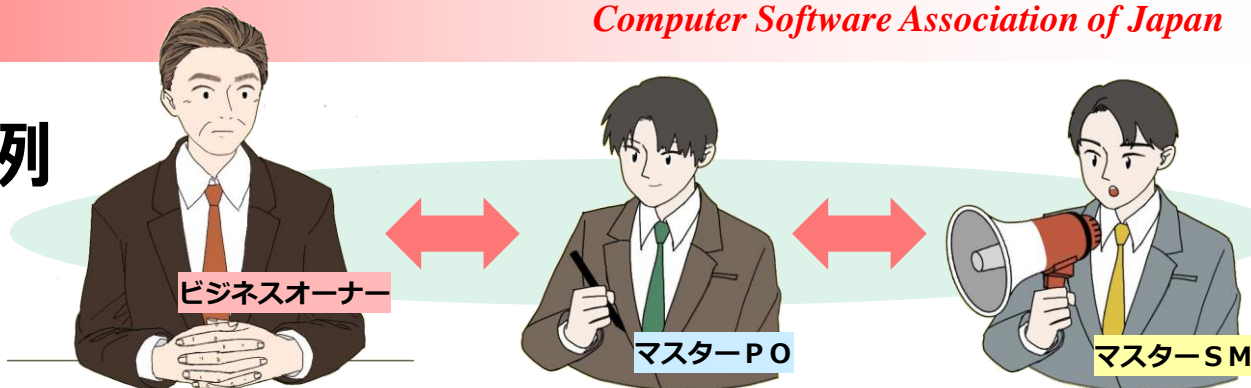
KANBANシステム


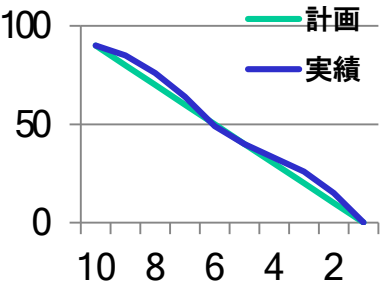

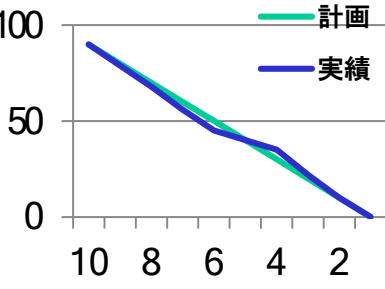

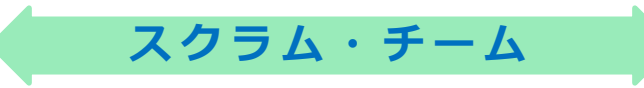

- プル型のシステム
- KANBANボードにより仕事が可視化
- チームの許容量に応じて、カンバンで仕掛（WIP）を制限可能
- 漸次改善を促すための改善ツール

メリット

- プロセス上の問題やボトルネックなどのスループットが可視化
- 品質向上や生産性向上を促進

KANBANの例



プロダクトバックログ	担当PO	分析&設計	ユーザーストーリー	開発チーム	テスト待ち	テスト	ユーザー受け入れテスト
XXXXX AAAA BBBB CCCC DDDD EEEEE FFFFF GGGG HHHH IIIIIIII JJJJJ KKKK LLLL MMMM	■ 山雄 	X A B C D	X-1 X-2 X-3 X-4 A-1 A-2 A-3		X B	C	D
	○ 川子 	G H J K M	G-1 G-2 G-3 G-4 J-1 J-2 J-3		G	H	K
							

KANBANとスクラムの類似点

- どちらもリーンであり、アジャイルです。
- どちらもプル・スケジューリングに基づいています。
- どちらもWIP（仕掛）を制限します。
- どちらもプロセス改善を推進するために透明性を活用します。
- どちらもリリース可能なソフトウェアを早く頻繁に提供する事を重視します。
- どちらも自己組織化したチームに基づきます。
- どちらもワークを部品に分割する事を要求します。
- どちらの場合でも、リリース計画は、経験値(データ：ベロシティー/リードタイム)に基づき常に修正(最適化)されます。

Henrik Kniberg (www.crisp.se) のAgile2010, Orlandoのプレゼンテーション
“KANBAN and Scrum – Making the most of both-”より抜粋

KANBANとスクラムの相違点

スクラム

1. タイムボックス化されたイタレーションが規定されている。
2. チームがそのイタレーション内での明確な作業量をコミットする。
3. 計画やプロセス改善のための測定基準にベロシティを使用する。
4. 機能横断的なチームが規定されている。
5. 項目が細分化されるので、1スプリント内で完了できる。
6. バーンダウンチャートが規定されている。
7. (スプリント毎に)WIPが間接的に制限されている。
8. 見積りが、規定されている。
9. 実施中のイタレーションに項目を追加できない。
10. スプリント・バックログは特定の一つのチームが所有する。
11. 三つの役割が規定されている。
12. スクラム・ボードはスプリントでリセットされる。
13. 優先順位が付いたプロダクト・バックログが規定されている。

KANBAN

1. タイムボックス化したイタレーションは任意である。
2. コミットメントは任意である。
3. 計画やプロセス改善のための測定基準はリードタイムを使用する。
4. 機能横断的なチームは任意であり、専門家のチームも許容している。
5. 特に項目のサイズは規定されていない。
6. 特に使用する図表は規定されていない。
7. (ワークフローの状態で)WIPは直接的に制限されている。
8. 見積りは、任意です。(開発チームが見積りをする事はムダ)
9. 容量が空いている限りいつでも新しい項目を追加できる。
10. KANBANボードは複数のチームや個人で共有される。
11. 如何なる役割も規定されていない。
12. KANBANボードは持続性がある。
13. 優先順位付けは、任意です。

Henrik Kniberg (www.crisp.se) のAgile2010, Orlandoのプレゼンテーション
"KANBAN and Scrum - Making the most of both-"より抜粋

KANBANの利用価値

- プロセスの可視化のツールとして進捗状況表示に利用可能
- スクラムと組み合わせ、効果を增強
 - チーム外のステークホルダー間への状況の透明化
 - ソフトウェアツール（RTCやTFS）による支援
- スクラムとの相違点を理解した上での導入の検討が重要

まとめ

アジャイル開発をどう活用するか？
(グループ・ディスカッションと発表)

ワーク 1

演習時間：45分 (ディカッションとまとめ)

ワーク 2

発表時間：各チーム5分



アジャイル開発講座

目次

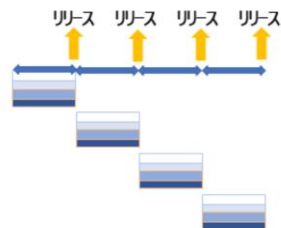
- 第1章 アジャイル開発の概要 4
- 第2章 スクラム（軽量なプロジェクト管理） 12

第1章 アジャイル開発の概要

- ・ アジャイル開発の概要
- ・ アジャイル宣言とアジャイル原則
- ・ アジャイルの価値観
- ・ パラダイム・シフト

アジャイル開発の概要（特徴）1

◆ 反復型（繰り返し）の開発



◆ 一人多役（多能）



アジャイル開発は、従来の開発手法と比べてどのような特徴があるのでしょうか？

まずは、親しんでいる方も多いウォーターフォール開発の手法と比較しながら、アジャイル開発の特徴を確認していきます。

ウォーターフォール開発の特徴から確認していきましょう。

ウォーターフォール開発は、工程ごとに作業を完了し、前工程には立ち戻らないような開発手法を指します。具体的には、企画、要件定義、設計、実装、テストなど、開発工程をいくつかに分けていきます。要件定義が終わったら設計に、設計が終わったら実装に、というように、1つの工程が終わると次の工程に進みます。この手法は、作業の流れが一方通行であり、非常に簡潔でわかりやすいというメリットがあります。また、前工程に立ち戻らないということは、各工程を確実に終わらせて次に進まなくてはならないような、工程内でのミスが許されないシステム開発に向い

ているといえます。しかしながら、前工程に立ち戻れないために発生するデメリットもあります。開発途中で顧客のニーズに変化があった場合には、途中で仕様を変更することができないため、ニーズに沿った商品を納品することは難しくなります。近年IT技術の進歩は目まぐるしく、顧客からのニーズも多様化しています。刻一刻と変化するニーズに対して、ウォーターフォール開発では対応しきれなくなってきました。

アジャイル開発は、開発するシステムを機能ごとに分割し、分割した機能に優先順位を付けて、優先順位の高い順に開発を進めていく手法です。ウォーターフォール開発が工程ごとに開発を進めることに対して、アジャイル開発は機能ごとに開発を進めます。つまり、アジャイル開発では、各機能の開発ごとに、分析、設計、実装、テストといった工程を繰り返します。開発途中でシステムの仕様に変更が出た場合、ウォーターフォール開発では、影響を及ぼす範囲が大きくなりすぎる懸念がありました。しかしアジャイル開発では、小さな機能が1つ完成した時点でその機能が顧客のニーズにマッチしているか確認することができます。もし、ニーズが変化していた場合でも、小さな機能を1つを変更することで対応が可能となります。これは、顧客のニーズに変化が生じた場合、最小限のシステム変更でニーズに対応することができるということを意味しています。一方で、アジャイル開発にもデメリットはあります。仕様が変更しやすいといっても、何でも変更できるような状態で開発を進めてしまえば、結果システム自体が完成しないということも在りえます。変更できない前提条件を明確にしておくことが非常に大切です。また、複数のチームに分かれて短期的な開発を繰り返すため、プロジェクト全体の進捗管理が困難であるともいえます。ですから、開発者の経験値に大きく依存してしまい、習得が難しいということがアジャイル開発のデメリットと言えるでしょう。

◆ スコープで調整



◆ 常時リリース

◆ 適応型開発プロセス

アジャイル開発の概要 (必要なスキル)

1. 分析設計スキル

ユーザーストーリー / タスク分解と見積り

2. 開発スキル

テスト駆動開発 / リファクタリング / ペア・プログラミング (XP)

3. テストスキル

継続的インテグレーション / 10分間ビルド (XP)

4. プロジェクト管理スキル

透明化 / タイムボックス

5. 品質管理スキル

DoD (完了基準)

6. チームビルディング・スキル

協調 / コミットメント / 合意 / カイゼン / リーダーシップ / チーム規約

ここでは、本講座で学んでいくアジャイル開発を実践する上で必要な知識やスキルをご紹介します。

分析設計スキルでは、ユーザーストーリーやタスク分解と見積もりについて学習します。

ユーザーストーリーとは、顧客の要求を簡潔にまとめたものです。つまり、顧客の要求を具体化・詳細化したものと言い換えることもできます。ユーザーストーリーの重要性や使用方法、作成方法について学んでいきます。

アジャイル開発では、短い期間の開発を繰り返し行いますが、その短い期間1つの間にチームがやらなくてはならないことのリストがタスクです。このタスクのうち一人一人がどこを担当するのか決めるためにタスクを分解しますが、適切に分解するための手法や、そのタスクを適切に見積もる方法についても紹介します。

開発スキルでは、アジャイル開発の中でもXPと呼ばれる手法を例に取り、テスト駆動開発、リファクタリング、ペア

プログラミングについて学習します。

テスト駆動開発やリファクタリングの流れを確認するとともに、ペアプログラミングではどのようにペアを組むべきなのか、なんのためのペアなのか等を紹介します。

テストスキルでは、継続的インテグレーションと10分間ビルドについて学習します。

継続的インテグレーションと10分間ビルドが、アジャイル開発においてどのように有益なのかを紹介します。

プロジェクト管理スキルでは、見える化とタイムボックスについて学習します。

見える化実現のポイントやその有用性、タイムボックスという概念の重要性について紹介します。

品質管理スキルでは、DoDについて学習します。

DoDとは、各作業をどこまで行えば完了なのかという基準のことです。品質管理の基本ともいえる重要な項目ですから、基本的なDoDの考え方を学んでいきます。

チームビルディング・スキルでは、協調、合意、リーダーシップ、コミットメント、カイゼン、チーム規約というような、良質なチームを形成するための要素について学習します。

一見当たり前のことばかりかもしれませんが、チームをいかに上手に形成するかは、プロジェクト成功に向けて最も重要な要素の1つですから、しっかり学んでいきます。

アジャイル開発概論

アジャイル開発とは、高品質で要求に柔軟なソフトウェア作成を行うため、協調性、自律性を重視した反復的な開発方法である。

- ダイナミックシステムズ開発技法
Dynamic Systems Development Method (Dane Faulkner ほか)
- アダプティブソフトウェア開発
Adaptive Software Development (Jim Highsmith)
- クリスタルメソッド
Crystal Methods (Alistair Cockburn)
- スクラム
Scrum (Ken Schwaber, Jeff Sutherland)
- XP / エクストリームプログラミング
(Kent Beck, Eric Gamma ほか)
- リーンソフトウェア開発
Lean Software Development (Tom and Mary Poppendieck)
- フィーチャ駆動開発
Feature-Driven Development (Peter Code, Jeff DeLuca)
- アジャイル統一プロセス
Agile Unified Process (Scott Ambler)

- 反復 (周期) 的 (Iterative)
--- 定期的なリリース
- 漸進的 (Incremental)
--- 徐々に機能を増加
- 適応主義 (Adaptive)
--- 変化に対応 (即応)
- 自律的 (Self-Organized)
--- 学習する組織
- 多能工 (Cell Production)
--- 一人多役
(SE、プログラマー、テスター)

ここでは、アジャイル開発の概論と、その技法の種類を紹介します。

アジャイルソフトウェア開発宣言

2001年2月 (於 米国ユタ州スノーバード) 制定

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する活動を通じて、より良い開発方法を見つけ出そうとしている。

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、
計画に従う事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置く。

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

<http://agilemanifesto.org/>

アジャイルソフトウェア開発宣言は、2001年2月、従来広く使われてきたソフトウェア開発手法とは違う開発手法を実践していた17名の開発者が集まり、それぞれの手法について議論が行われた結果生まれた宣言です。アジャイルソフトウェア開発宣言に記されているのは、具体的な手法ではなく、重視すべきマインドセットです。アジャイル開発を正しく身につけるためには、アジャイルソフトウェア開発宣言の解釈を誤り無く理解することが重要です。

宣言内にある「左記のことがら」とは、プロセスやツール、包括的なドキュメント、契約交渉、計画に従う事、を指しています。アジャイルソフトウェア開発宣言を一見すると、左記のことがらは重要視されていないように誤解されがちですが、「左記のことがらに価値があることを認めながらも」とあるように、決してアジャイル開発には不要なものであるということが言いたいわけではありません。より価値を置くとされている「右記のことがら」、個人との対話、動くソフトウェア、顧客との協調、変化への対応、というマインドセットがしっかりあることを前提に、左記

のことがらを考えていかななくてはならないことを示しています。

この宣言は、次ページで紹介する「アジャイル宣言の背後にある原則」と合わせて自由にコピーできますが、この注意書きを含めて、全文を掲載しなければなりません。

アジャイルソフトウェア開発の原則（1）

私たちは、以下の原則に従う。

1. 顧客満足を最優先し、
価値あるソフトウェアを早く継続的に提供します。
2. 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。
3. 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。
4. 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。
5. 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。
6. 情報を伝える最も効率的で効果的な方法は
フェイス・ツー・フェイスで話をすることです。

アジャイルソフトウェア開発宣言には、更にこのような原則が付加さ

れています。この原則はソフトウェア開発で何を重視すべきかが揺わ

れています。これはソフトウェアの価値を最大化するための具体的な

行動指針です。

アジャイルソフトウェア開発の原則（2）

7. 動くソフトウェアこそが進捗の最も重要な尺度です。
8. 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。
9. 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。
10. シンプルさが本質です。
11. 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。
12. チームが最も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。

<https://agilemanifesto.org/iso/ja/principles.html>

アジャイルの価値観

アジャイル開発と他のソフトウェア開発手法との違いを整理します。

アジャイル開発	他の開発手法
• タイムボックス制	(← 長期間)
• 完全なチーム作業	(← PMの存在)
• ユーザー第一主義	
• 一人多役	(← 分業制)
• チームの自己組織化	

アジャイル開発においては、パラダイム・シフトが必須！

正解主義から適応主義への変遷！

アジャイル開発と他のソフトウェア開発手法との大まかな特徴の違いは前述しましたが、ここでは、もう少し具体的に違いを見ていきましょう。

アジャイル開発は、時間の使い方に特徴があります。アジャイル開発における、区切られた短い開発期間のことをイテレーションといい、イテレーションの中でやるべきことのリストをタスクといいます。イテレーションの開始時にゴールを設定し、ゴールから逆算してタスクを設定します。このような時間の設定方法をタイムボックスといいます。つまり、固定の時間（タイムボックス）内でやるべきことを明確にするということです。タイムボックスもできるだけ短い時間で区切るようにして、ダラダラと作業することを避けます。

チームでの開発にも重きを置いています。少数精鋭のチームを編成し、チーム内でも一人一人が自分の責任を常に意識しながら開発に取り組んでいくことで、結果、自己組織的なチームとして開発を有利に進められます。

要求の変化に対応できるのがアジャイル開発の特徴の1つでした。つまり、顧客の要求の変化にいかに対応していくかが重要視さ

れています。イテレーションの最後に顧客から受けたフィードバックを、積極的にプロジェクトに反映させていくことが大切です。そして、正解を求めて開発を進めるのではなく、変化への適応を受け入れながら開発を進めるのが、アジャイルの特徴です。

第2章 スクラム

- スクラムの価値観（働き方）
- パラダイム・シフト

スクラムの三つの価値

- ◆監視 Inspection
- ◆透明性 Transparency
- ◆調整 Adaptation

プラス

タイムボックスの利用

スクラムガイド（

<https://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-Japanese.pdf>）によると、「スクラムは、経験的プロセス制御の理論（経験主義）を基本にしている。経験主義とは、実際の経験と既知に基づく判断によって知識が獲得できるというものである。スクラムでは、反復的かつ漸進的な手法を用いて、予測可能性の最適化とリスクの管理を行う。」とあります。つまり、ウォーターフォール開発のように予測から成るプロセスではなく、短い期間で何度も経験して成果に近づいていくプロセスであるといえます。また、スクラムガイドには、「経験的プロセス制御の実現は、透明性・検査・適応の3本柱に支えられている。」ともあります。経験を繰り返して開発を進めるために重要なのが、透明性・検査・適応の3つということです。

透明性とは、チーム内においてチームの現状（進捗状況や問題点等）が見える化されており、チーム全員が共通の認識を持つことができているかということです。チームの状況が明確にわかり全員がそれを共有していれば、自ずと次

にすべきことは見えてくるはず。透明性が保たれることで、次の行動につながっていきます。

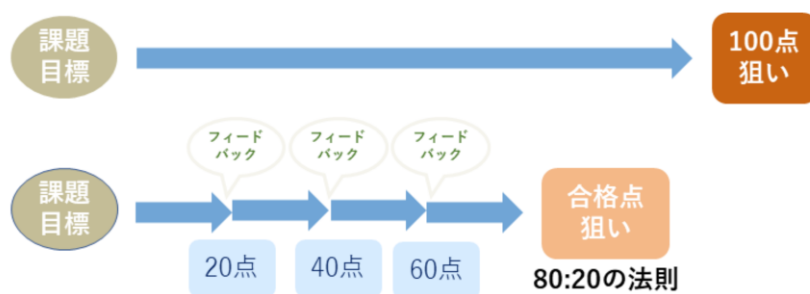
検査とは、成果物の出来やプロジェクトの進捗に問題がないか検査し、問題点を洗い出すことを指します。検査はその頻度設定が重要で、多いほど好ましいといえますが作業の妨げになるほど増やしてはいけません。

アジャイル開発の仕事の基本構造

イテレーション (スプリント)

- 時間の使い方
- 透明性
- 品質

反復的に開発を行った方が
良いのか？
どうして、そうしなければ
いけないのか？



アジャイル開発は、短い期間での開発を反復的に何度も行うことでゴールに近づいていきます。その短い一回の開発期間の事をイテレーションといいます。一般的なアジャイル開発ではイテレーションといいます。スクラムではイテレーションのことをスプリントといいます。

では、なぜ反復的に開発を行うのでしょうか？

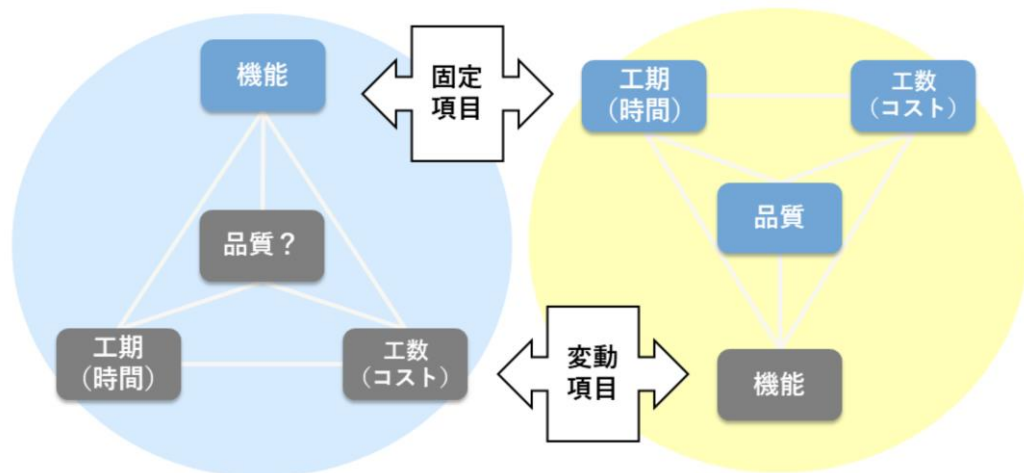
反復的に開発を行う利点として、要求の変化に対応できること、優先順位の高い順に進めるため動くソフトウェアを早い段階から確認できること、短期的であるため計画の見通しが立ちやすいこと、検査も繰り返し行うため問題の検知が早いこと、などが挙げられます。

そもそもアジャイル開発は、従来のウォーターフォール開発のような開発手法と比べて、柔軟性に富んで一線を描くという発想から成り立っています。この発想を現実化するためには、あえて短期的な計画しか明確にせず変化を受け入れる余地を残しておかなければいけません。そうすると、小分けに開発を進めていく手法が最も効率的なのです。

アジャイルの価値観

従来の開発アプローチ

DSDM, Atern(アターン)の開発アプローチ



アジャイル開発の基本的な考え方を紹介します。従来の考え方、アプローチからは大きなパラダイムシフトになります。

従来は、まず開発すべき機能（要求）が大前提で、これだけの機能を実装（開発）する為には、これだけの期間（工期）と何人の体制（コスト）が掛かりますというプロジェクト計画の作成方法です。

アジャイルでは、逆の発想で、現在与えられているリソース、すなわち与えられた時間（工期）と予算（コスト）の範囲内で実現できる機能（要求）はどれだけか？というプロジェクト計画になります。

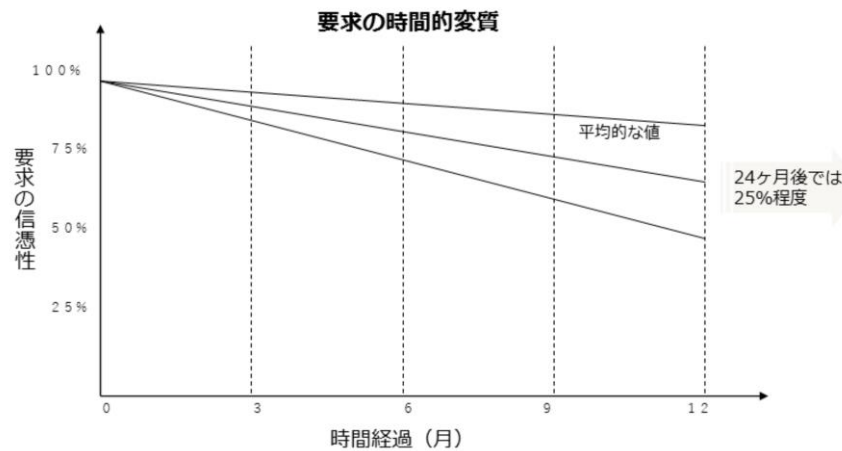
したがって従来の方法では、決められた機能を100%実装するために、工期、工数が不足してくると、品質を犠牲にするアプローチではないでしょうか？

アジャイルでは、工期、工数、品質を満たす事が前提で、実現される機能で調整を行う事になります。

要求の変質

迷信：要求を明確に定義できれば良い

現実：要求は時間と共に変化



仮に要求仕様をしっかりと定義できてもその要求自体が変質しています。

開発初期における要求定義

従来の、開発初期に明確な要求定義をすれば、良いシステムが

できあがるという考え方は、データから見ても迷信である。要求の

信憑性は時間の経過とともに下がるため、初期の要求を変更せ

ず開発を進めることは品質の向上に繋がりにくいことを説明する。