アジャイル開発

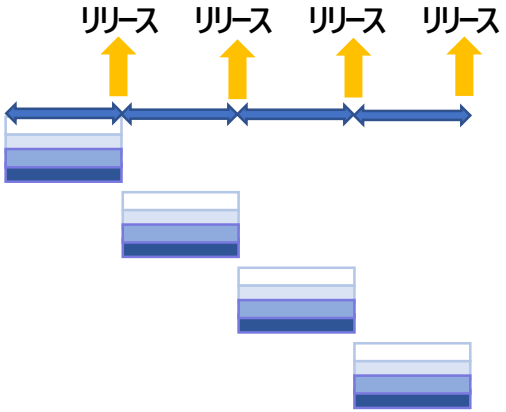
目　次

第１章 アジャイル開発の概要

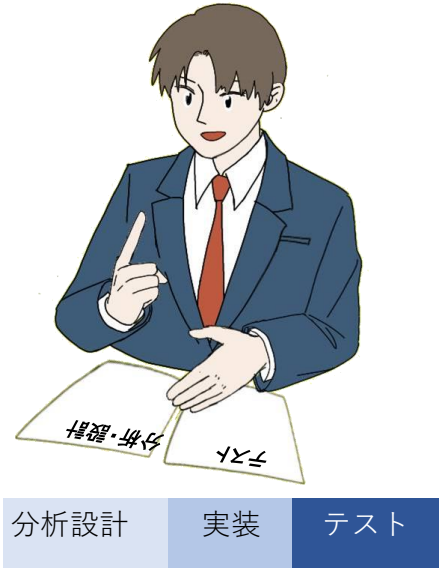
- アジャイル開発の概要
- アジャイル宣言とアジャイル原則
- アジャイルの価値観
- パラダイム・シフト

アジャイル開発の概要（特徴）1

◆ 反復型（繰り返し）の開発

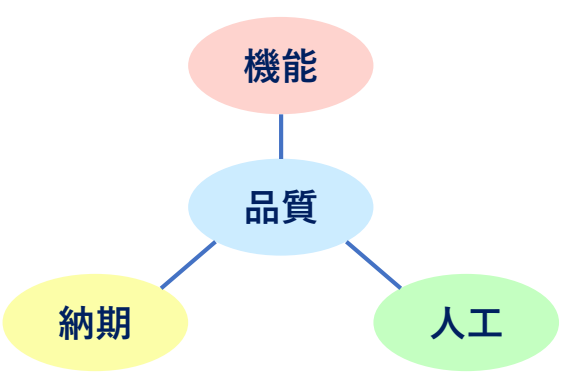


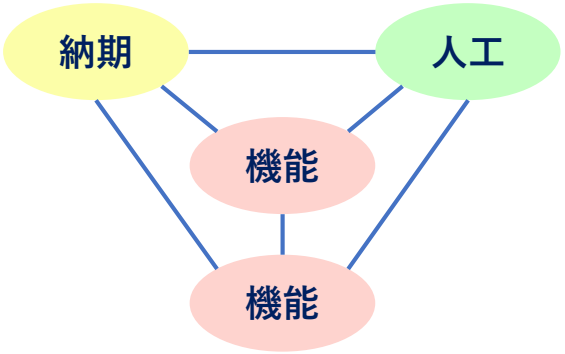
◆ 一人多役（多能）



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エア開発宣言

2001年2月　（於　米国ユタ州スノーバード）　制定

私たちは、ソフトウエ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エアを、
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エア開発の原則（１）

私たちは、以下の原則に従う。

1. 顧客満足を最優先し、
   価値あるソフトウエ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. 動くソフトウエアを、2～3週間から2～3ケ月という
   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エア開発の原則（２）

7. 動くソフトウエ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エ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！

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エ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

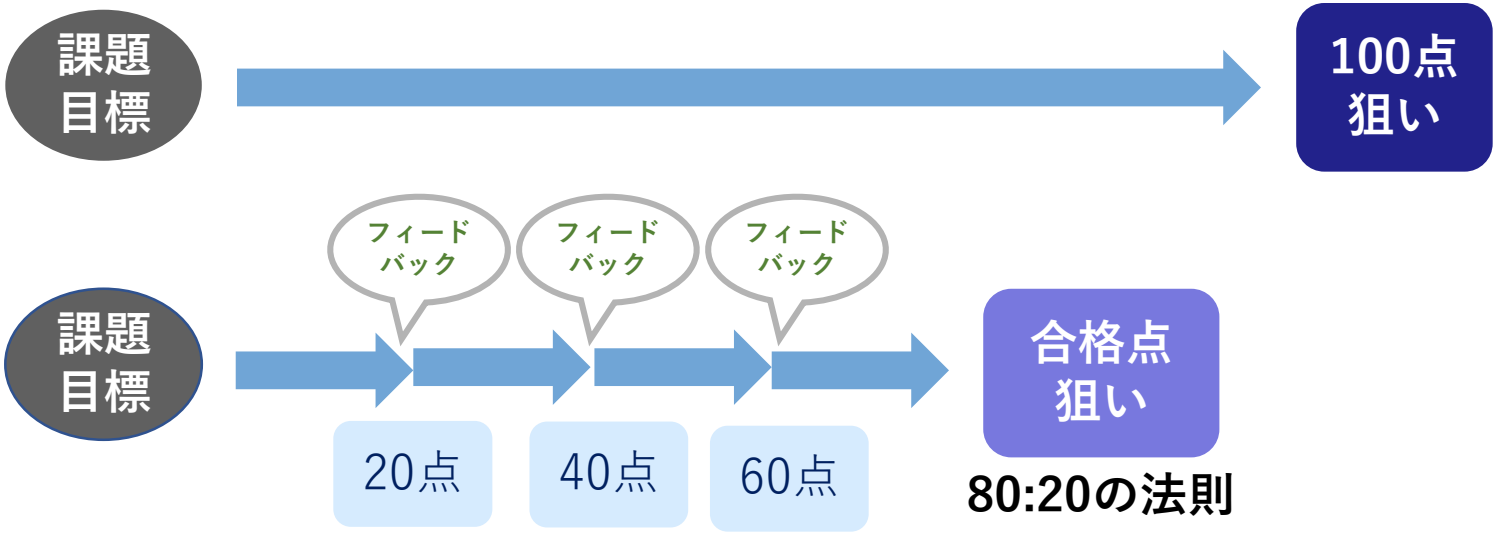
アジャイル開発の仕事の基本構造

イテレーション（スプリント）

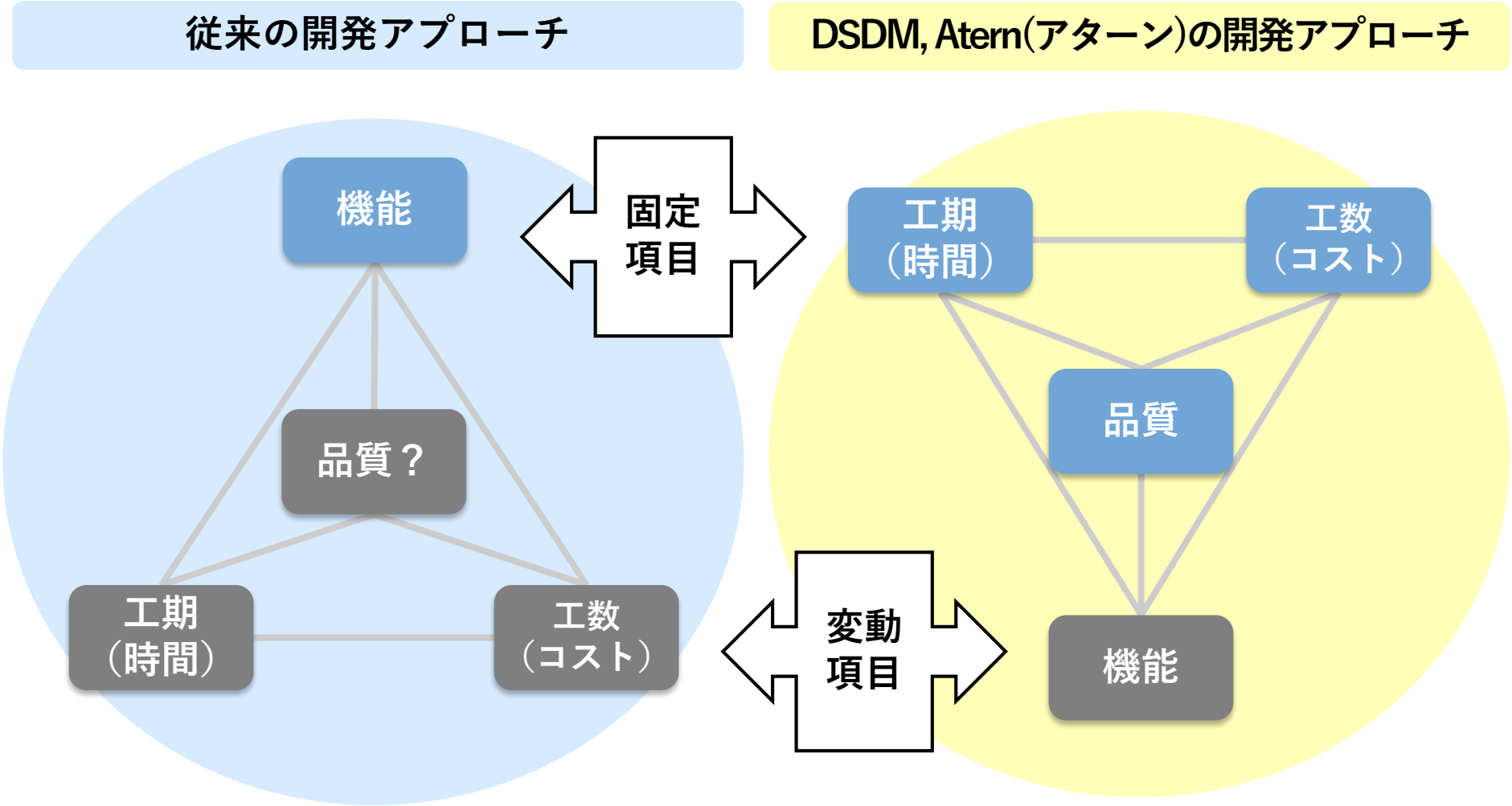
- 時間の使い方
- 透明性
- 品質

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？



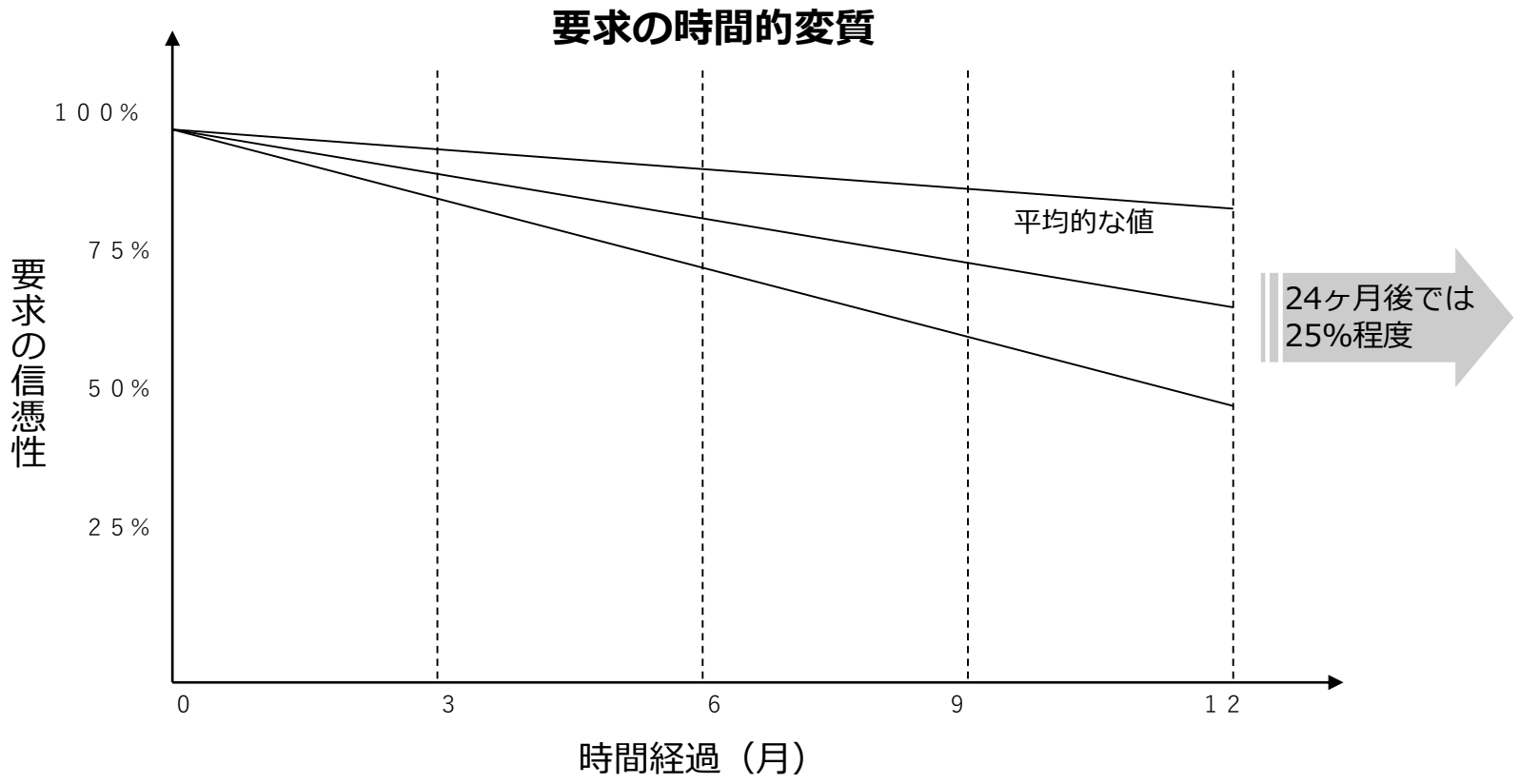
アジャイルの価値観


従来の開発アプローチ
DSDM, Atern(アターン)の開発アプローチ
機能
固定
項目
品質？
工期
（時間）
工数
（コスト）
工期
（時間）
工数
（コスト）
品質
変動
項目
機能

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）

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

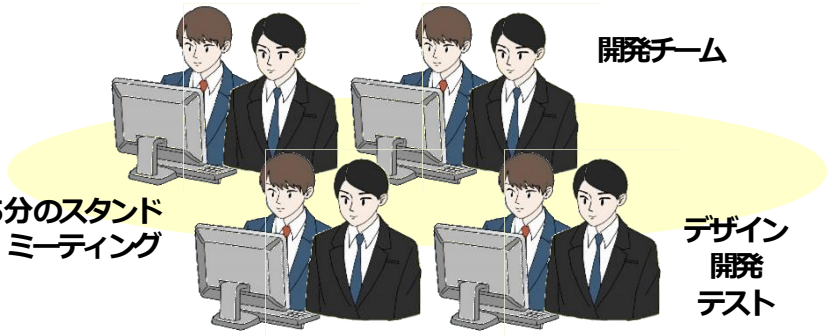
タスクボード

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

4週間 | 4週間 | 4週間 | 4週間 | 4週間

イタレーション1 | イタレーション2 | イタレーション3 | イタレーション4 | イタレーション5

納品 | 納品 | 納品 | 納品



開発チーム

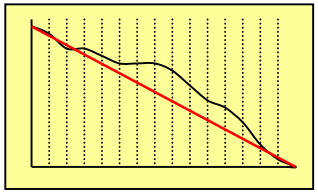
毎朝15分のスタンド
アップ・ミーティング

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）

スプリントレビュー
1時間

バーンダウン・チャート



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

CSAJ

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

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

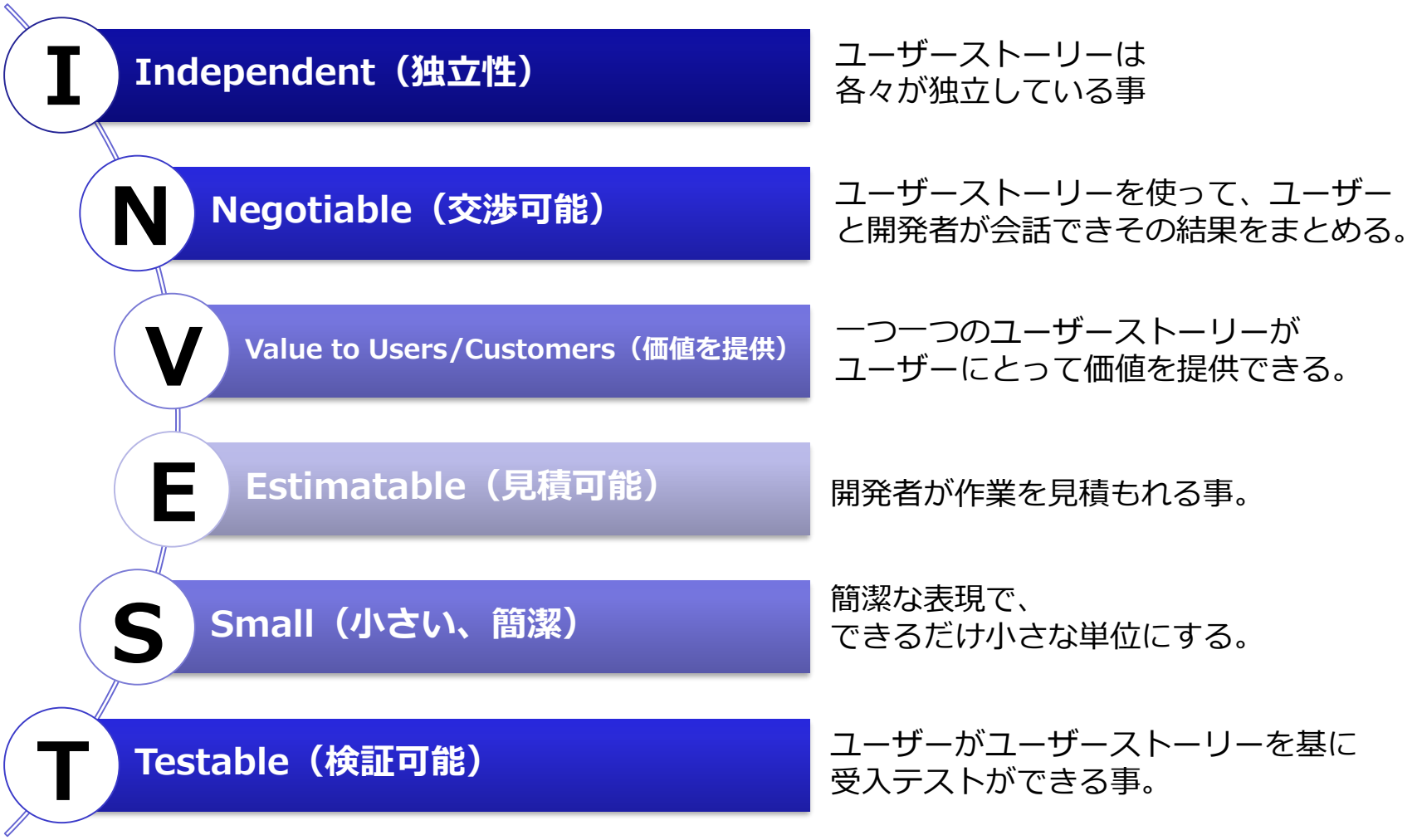
「役割・利用者 (As a role)」

「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。

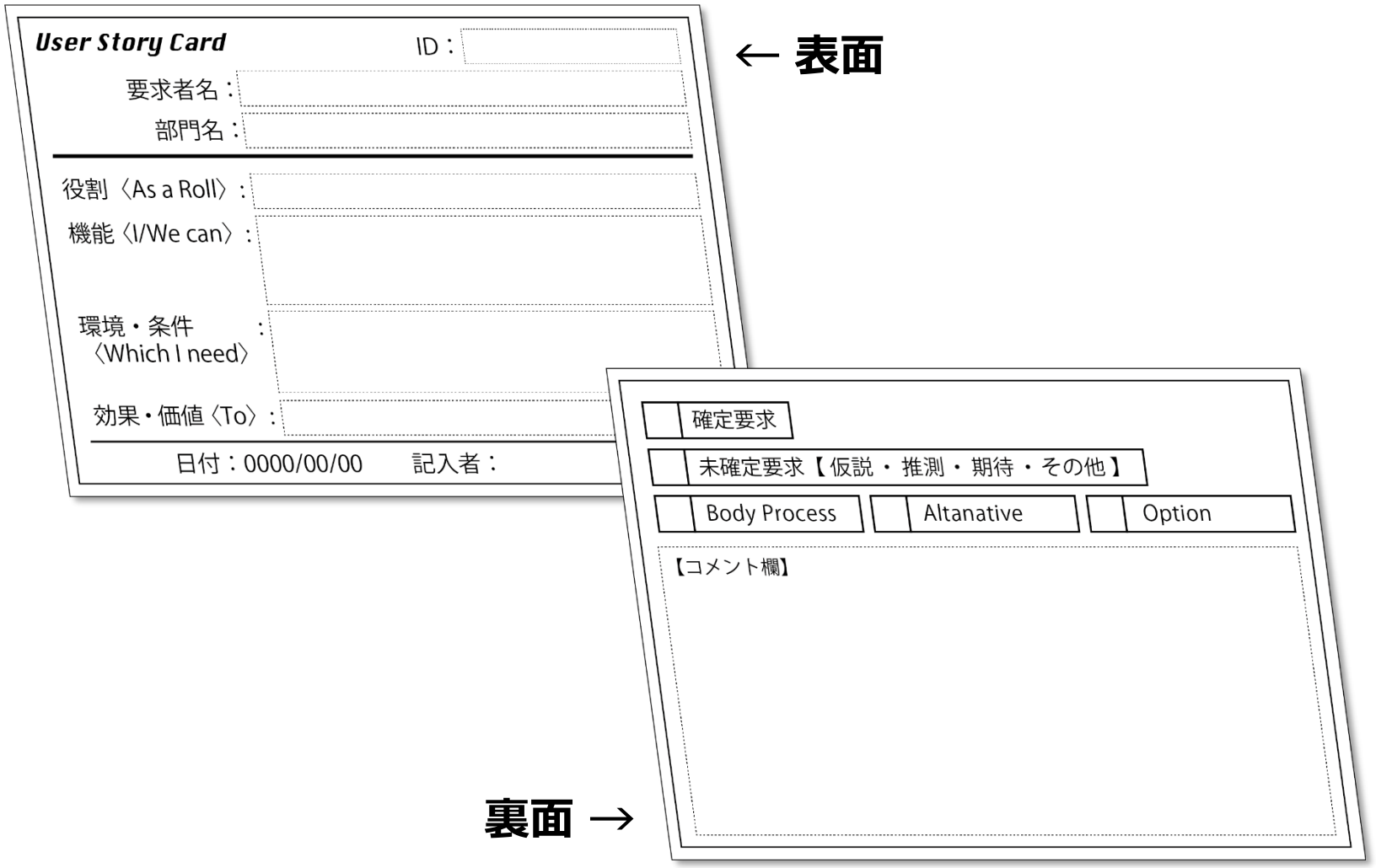
(例)

＜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）



← 表面

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
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

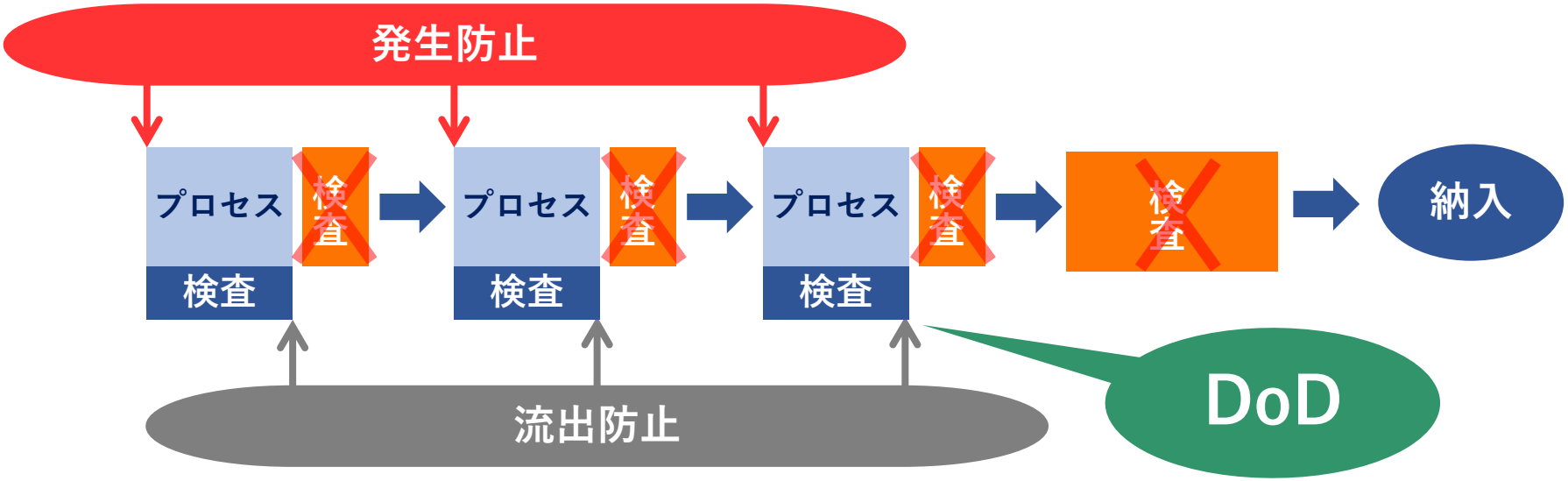
- 毎日チーム全員で行う短時間（15分程度）の状況確認ミーティング
- 継続と透明化を重要視する
- 「前日の作業結果」「当日の作業予定」「課題」を全員が同じ順に報告
- 必要に応じてフォローアップ・ミーティングを行う

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



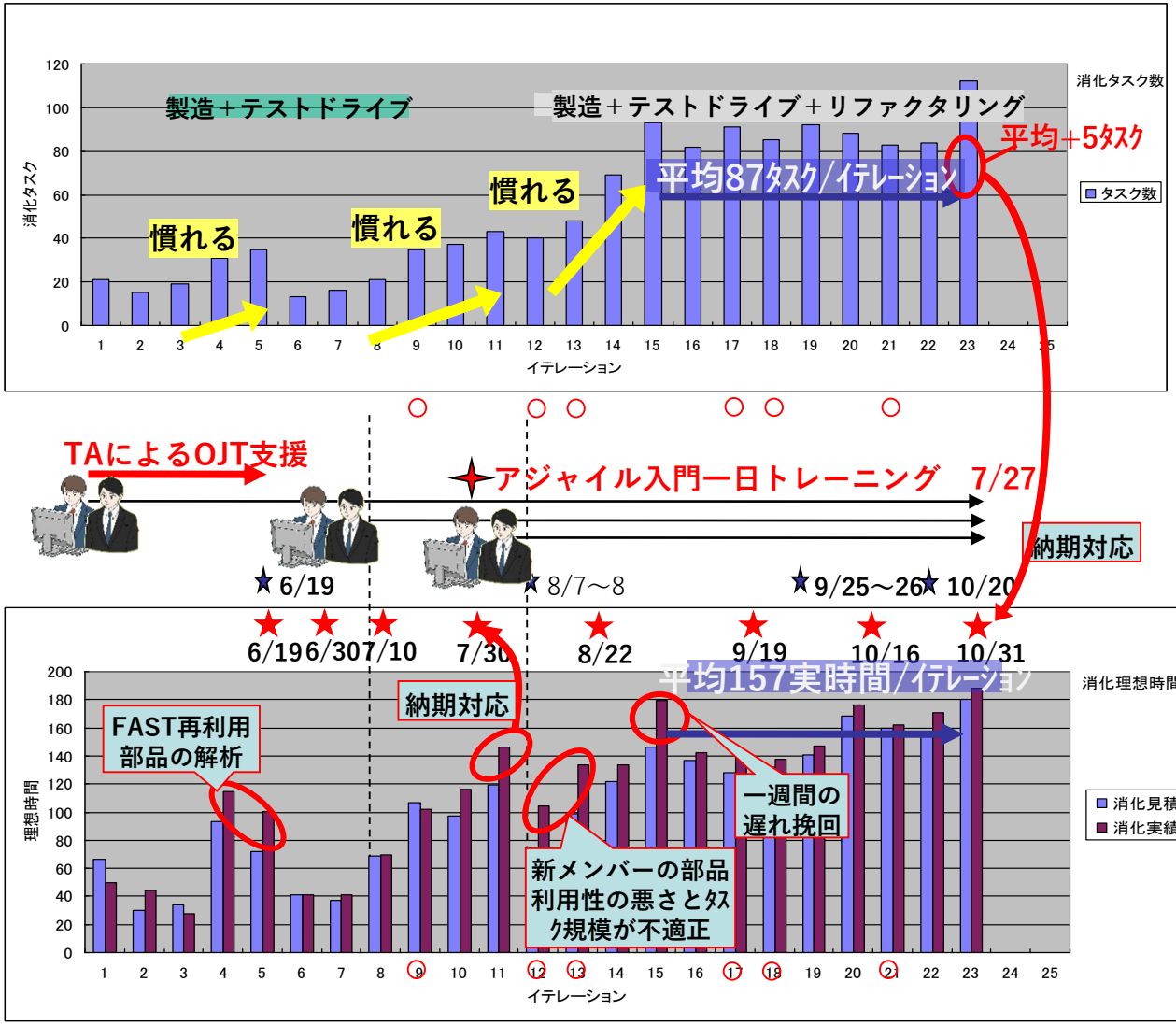
バーンダウン・チャートとベロシティ

［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

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

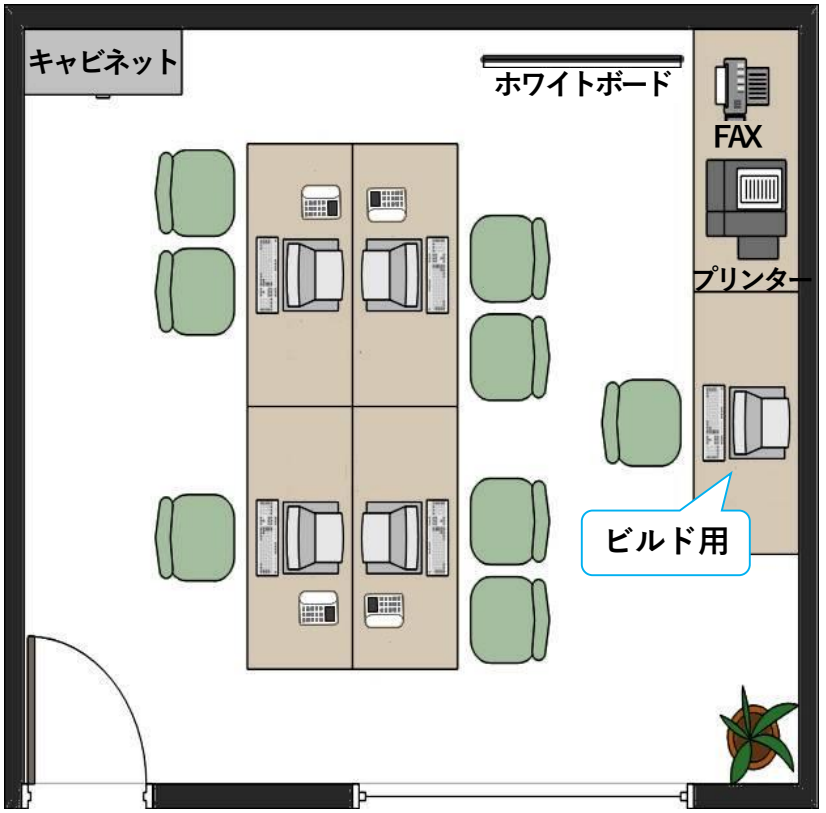
開発チーム

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

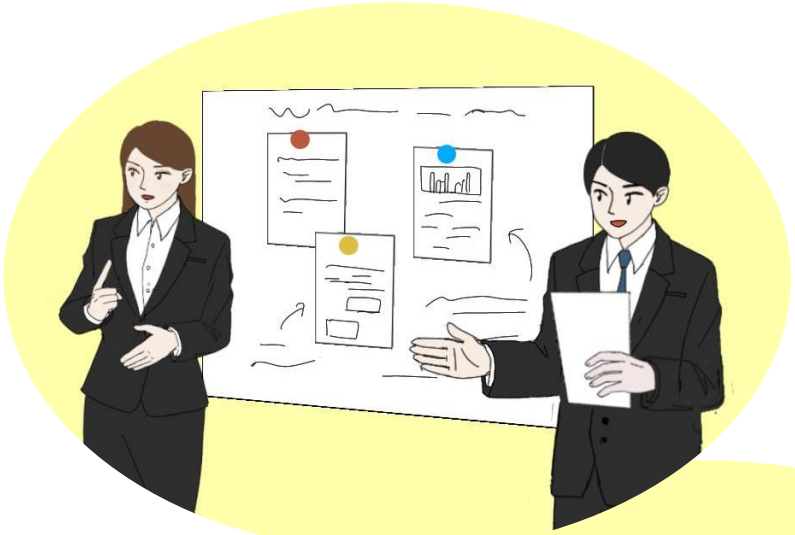
スクラムの環境

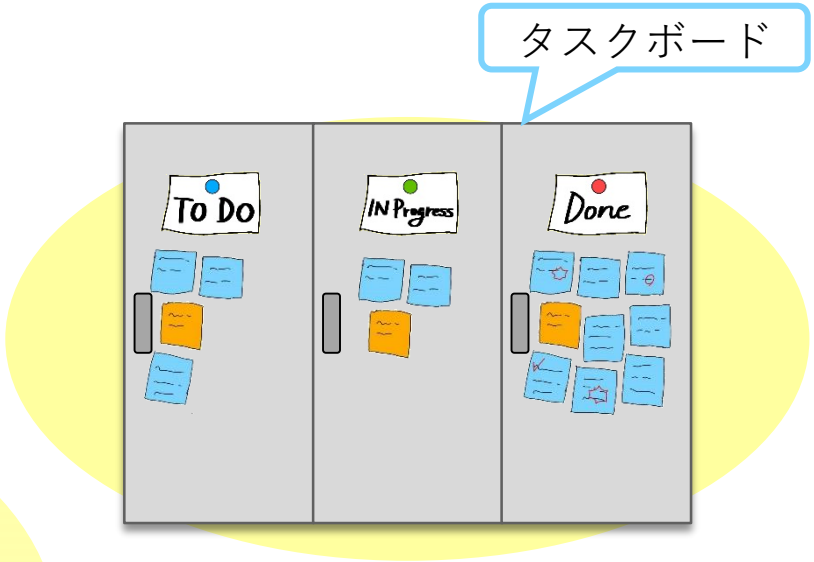
- 透明性の高い作業環境
  区画の無い共有作業スペース
  →コミュニケーションのとりやすさを意識



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。

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
・陣嘗、トラブル
・進抄
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
・ペアブログラミング
・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。
- 今日実装した複維な厳能は、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？
- ユーザとの直掻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

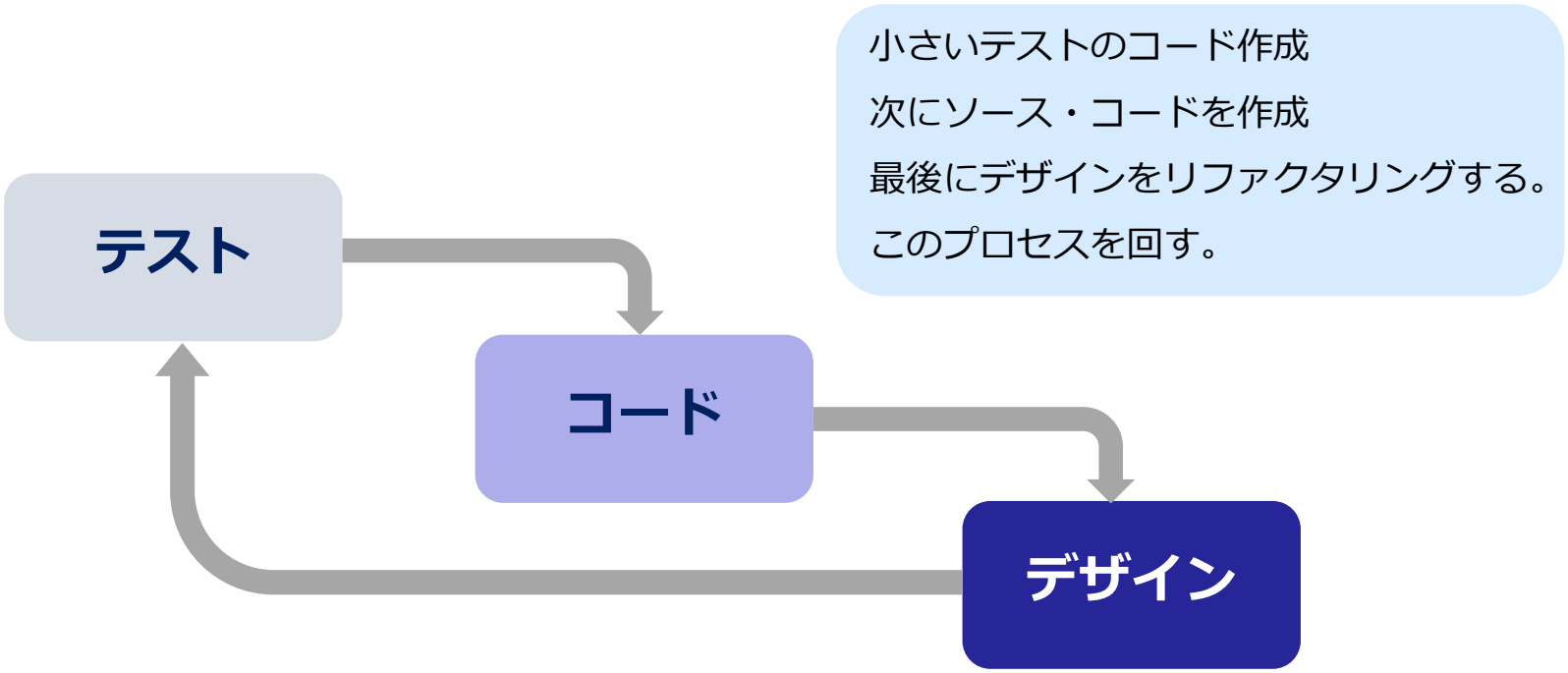
株動するシステムがもっとも情報量が多いフイードバック
・要求：ユーザーにシステムを操作してもらいながら
・段計：実際のコードを見たり、機能追加をしてみる
・品質：線助しているシステムに対してテストを行う
・進捗：助作している機能を見る／実績をみる

4.勇気

コミュニケーション、シンプル、フイ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棄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

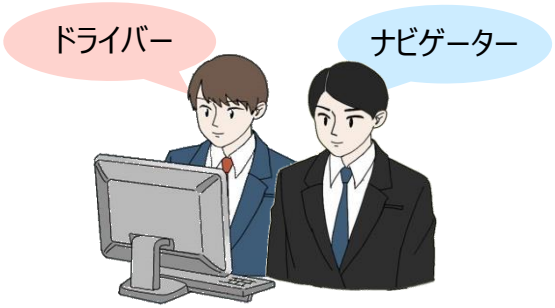
- コードをできるだけ小さな単位に分割する
- 処理のかたまりに名前をつける
- 処理を重複して記述しない
- コードをできるだけ小さな単位に分割する（部品が小さいほうが理解しやすく、バグも少なくなる）

- 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。

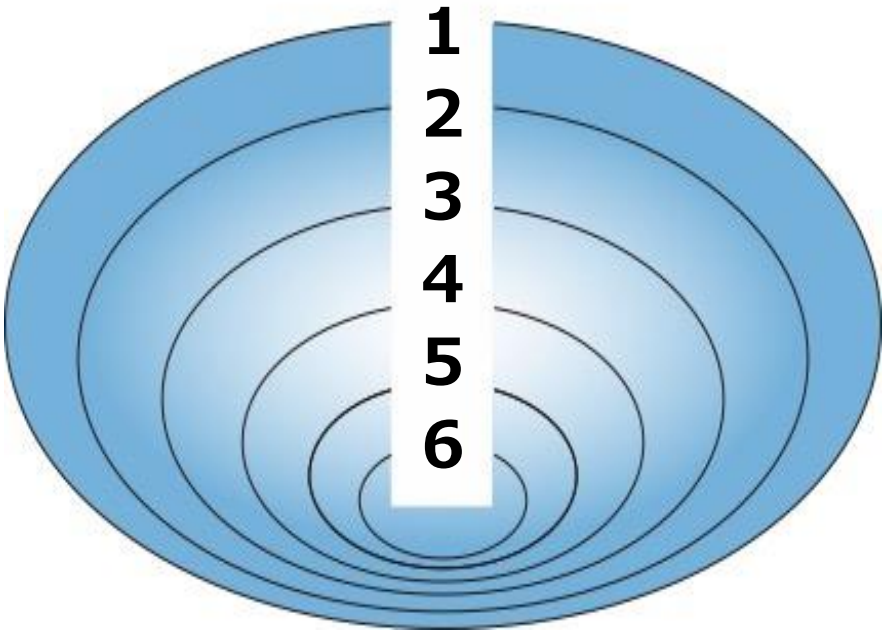
第５章 アジャイルな計画

- 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

リリース計画とスプリント計画

アジャイルプロジェクトはリーリスの繰り返しから、
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

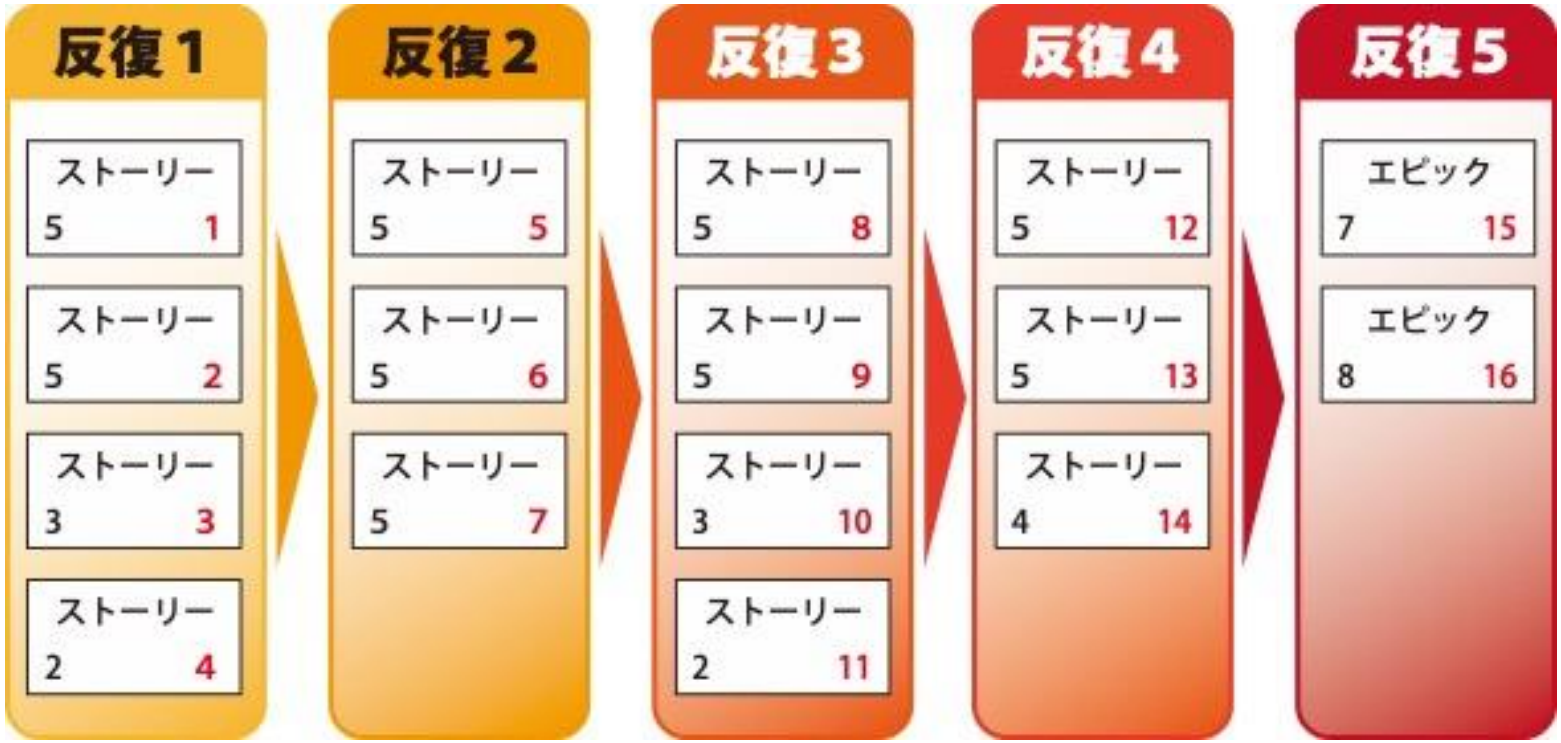
全体リリース計画の立案

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
- 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）

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

開発チーム

毎朝15分のスタンド
アップ・ミーティング

デザイン
開発
テスト

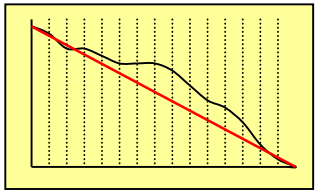
タスクボード

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

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

バーンダウン・チャート



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

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

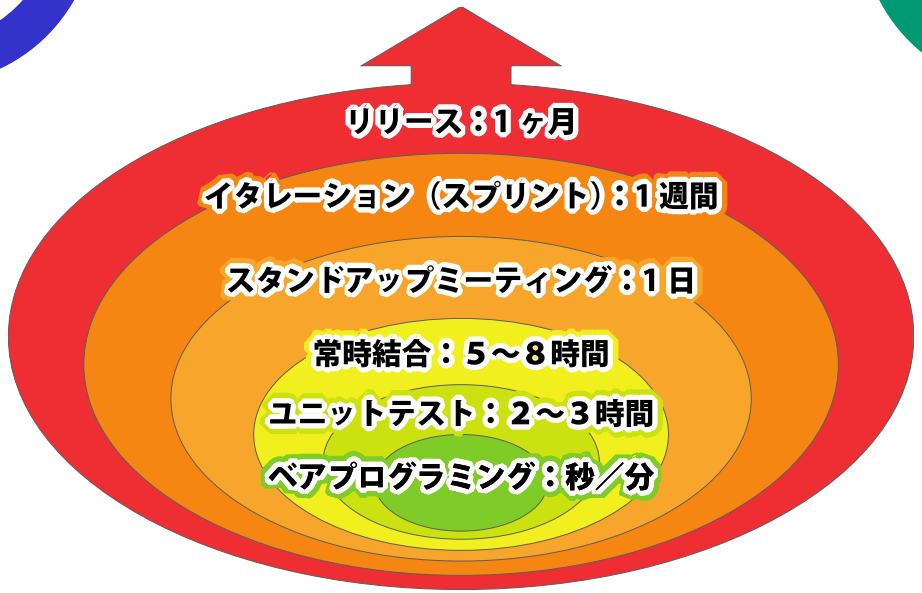
発表時間：　各チーム5分

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

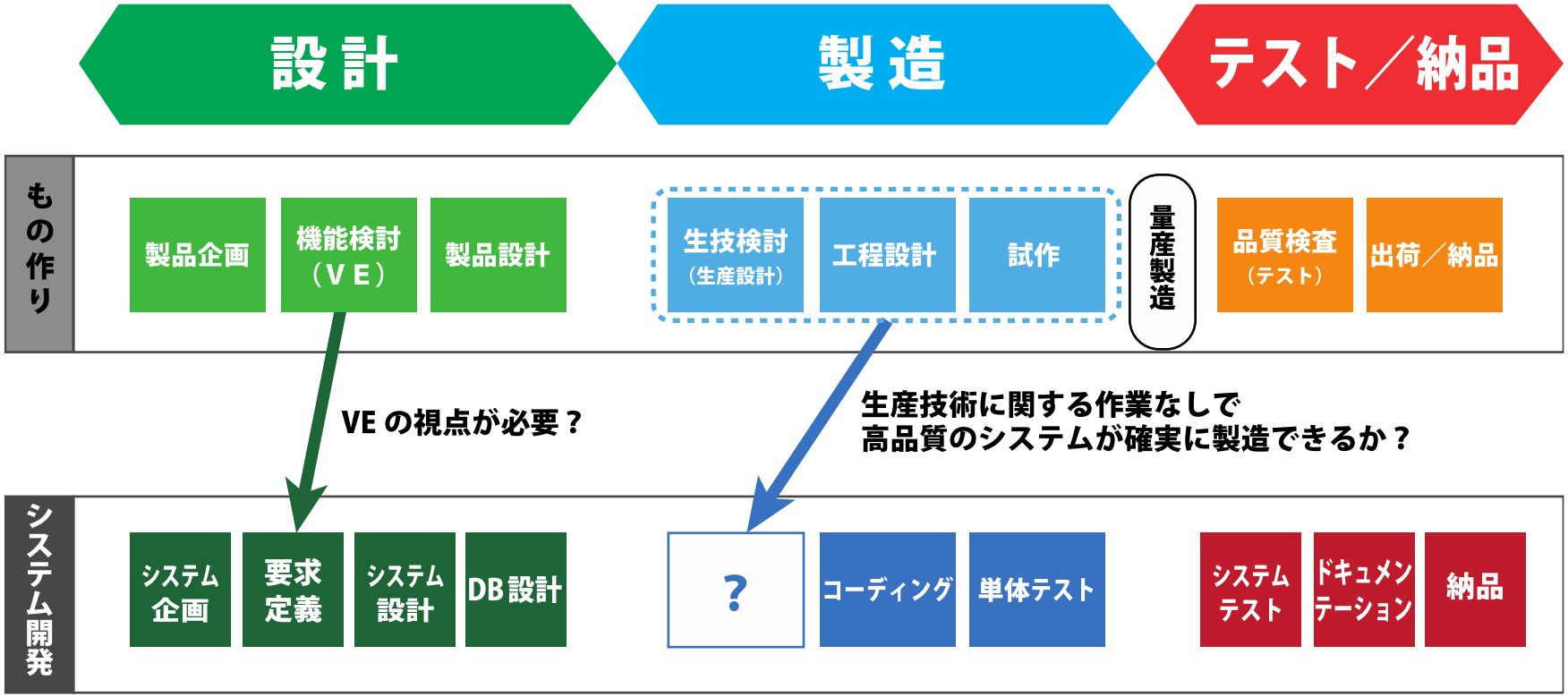
  自工程完結で周囲への影響が減少

  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

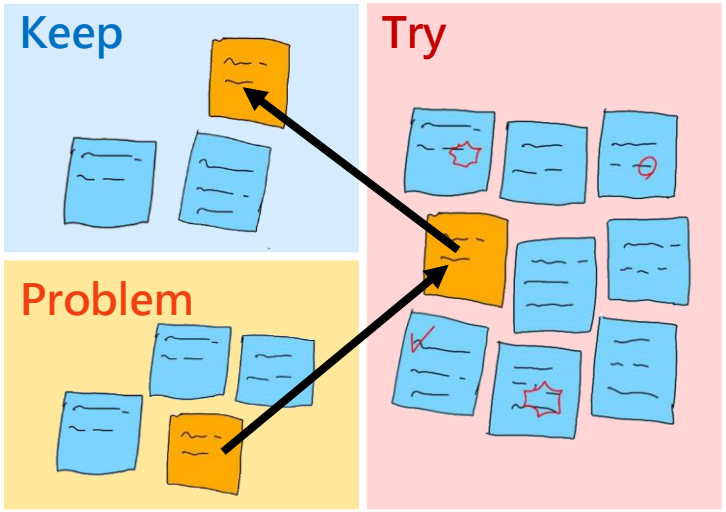
『もの作り』と『システム作り』の相違



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。

②Problem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。

④Problem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、躾）
- 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

ソフトウエ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ケ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　ｘｘｘ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

チーム

デイリースクラム

スクラムマスター

チーム

デイリースクラム

スクラムマスター

チーム

デイリースクラム

スクラムマスター

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

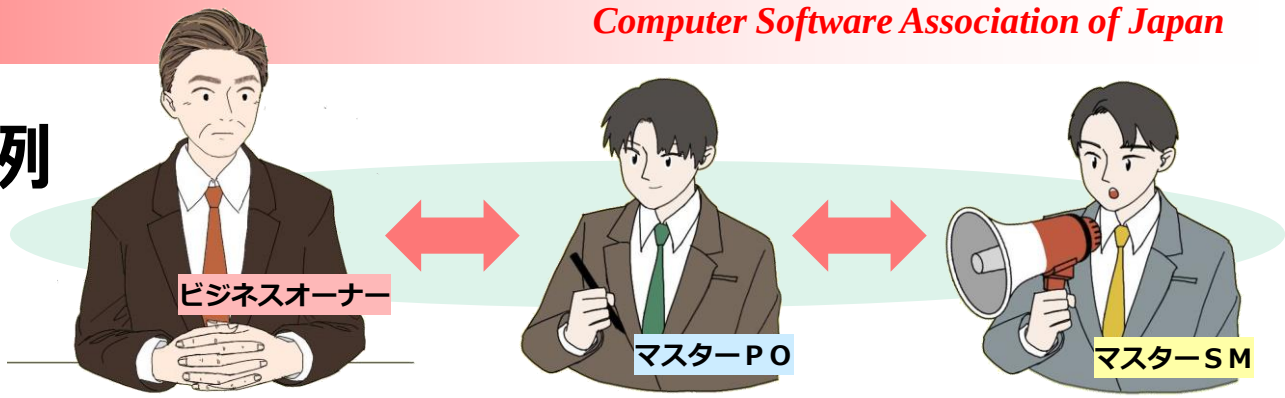
KANBANシステム

- 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	■山■雄	X A B C D	X-1 X-2 X-3 X-4 A-1 A-2 A-3	100 / 50 / 0 10 8 6 4 2 計画 実績	X B	C	D
GGGG HHHH IIIIIIIII JJJJJJ KKKK LLLLL MMMM	○川○子	G H J K M	G-1 G-2 G-3 G-4 J-1 J-2 J-3	100 / 50 / 0 10 8 6 4 2 計画 実績	G	H	K

プロダクトオーナー　スクラム・チーム　ＱＡ

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ーを使用する。
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. 実施中のイテ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

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