ソフトウェア開発プロセス

内容

- 開発プロセス
- ソフトウェア開発モデル
- アジャイル開発

開発プロセス

開発プロセスとは

- ソフトウェアを計画的・体系的に開発するための手順や流れ
- 要件定義から設計、実装、テスト、運用・保守までの一連の工程
- 各工程で「何を」「どのように」進めるかを明確にすることで、品質や納期、コストを管理しやすくする
- 開発プロセスを定めることで、チーム全体で共通認識を持ち、効率的な開発が可能になる

開発プロセスの必要性

- 無計画な開発
  - 要求が曖昧なまま実装を始めると、後から仕様変更が頻発し、混乱や手戻りが発生
  - 結果として「動くけど使いにくい」システムになる可能性も
- スケジュールの遅延
  - 各工程の見積もりが不十分だと、後工程にしわ寄せが発生
  - 「テストする時間が足りない」「無理な残業でリリース」などの問題につながる
- 低品質なソフトウェア
  - バグや設計ミスが多いと、運用開始後に障害対応が頻発
  - 保守コストが膨らみ、新機能追加にも影響
  - 例：リリース後に「本番環境でしか起きない不具合」が多発し、信頼を損なう

- ソフトウェアは目に見えない資産であり、試作とやり直しのコストが高い
- 「何を作るか」「どう作るか」「いつまでに」「どのような品質で」を明確にする必要がある
- 再現性と改善性のある開発を可能にするために、標準化されたプロセスが求められる
- 開発チームの経験や規模にかかわらず、共通の枠組みを持つことで、属人化の防止や品質の均一化が図れる

ソフトウェアのライフサイクル

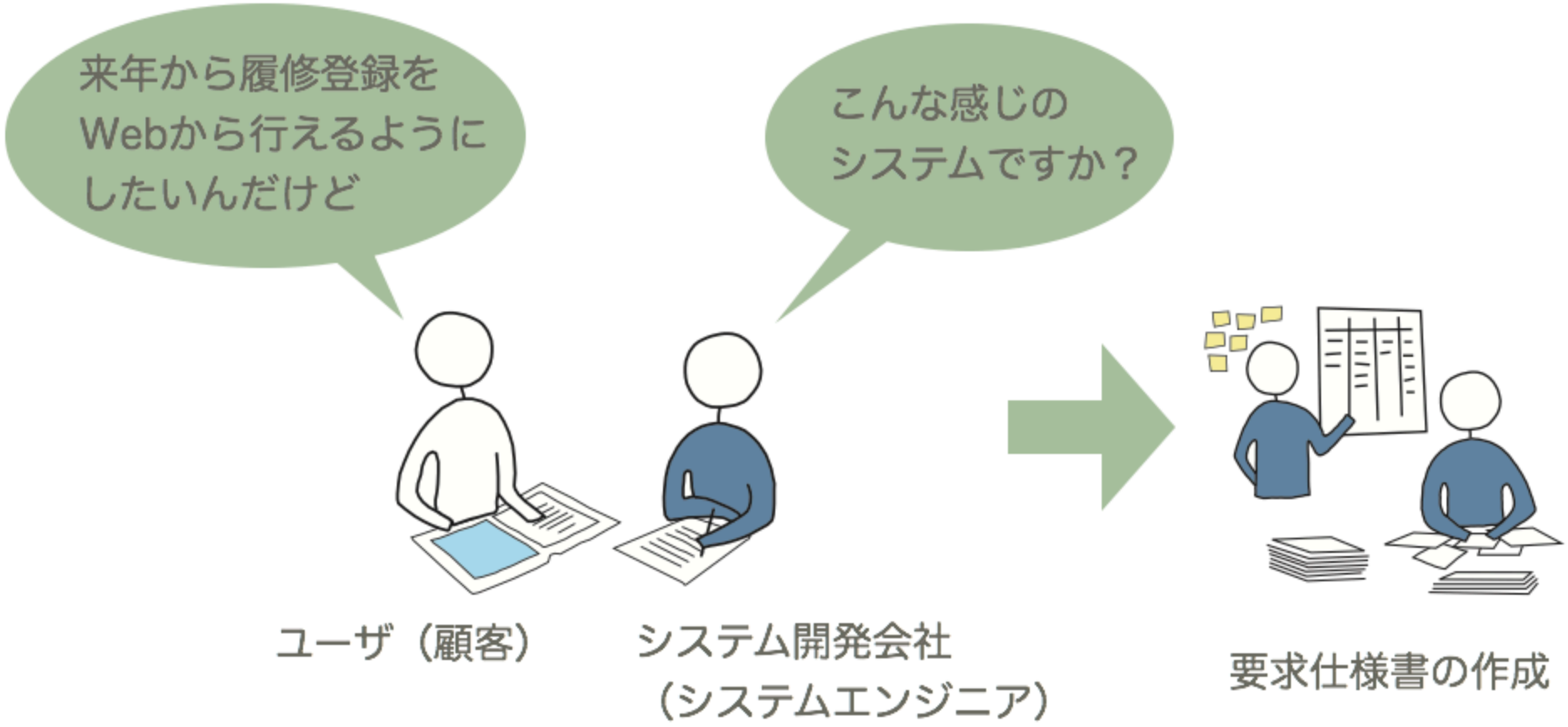
1. 要求定義（何を作るか決める）
2. 設計（システム構造を決める）
3. 構築（コードを書く）
4. テスト（バグを検出し修正）
5. 運用・保守（システムを継続的に改善）

要求定義

目的

- ユーザの要求を満足するような仕様（specification）を決める．
- ユーザの要求を明確にし，システムが何をするかを定義する．
- ユーザの要求を文書化し，開発チーム全体で共有する．
- 要求定義を行うことで、開発の方向性を明確にし、後工程での手戻りを減らす．

要求定義のイメージ

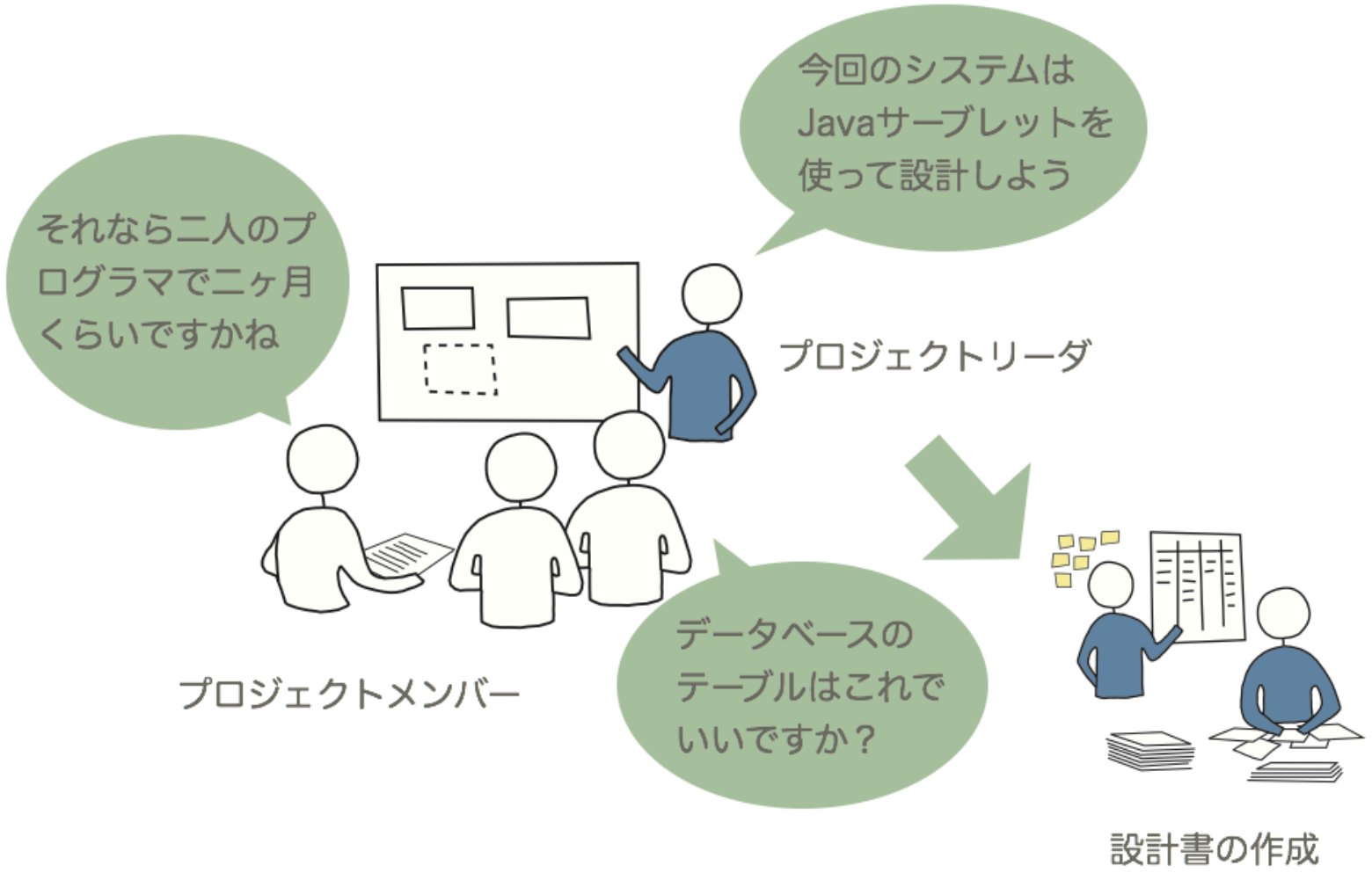

来年から履修登録を
Webから行えるように
したいんだけど
こんな感じの
システムですか？
ユーザ（顧客）
システム開発会社
（システムエンジニア）
要求仕様書の作成

設計（Design）

目的

- 要求定義に基づいて、システムの構造や動作を具体的に設計する.
- ソフトウェアアーキテクチャ（architecture）や実装方法（implementation）を決める.
- システムの各コンポーネントの役割やインタフェースを定義する.
- 設計を行うことで、実装の効率化や保守性の向上を図る.

設計のイメージ


今回のシステムは
Javaサーブレットを
使って設計しよう
それなら二人のプ
ログラマで二ヶ月
くらいですかね
プロジェクトリーダ
プロジェクトメンバー
データベースの
テーブルはこれで
いいですか？
設計書の作成

構築（Construction）

目的

- 設計に沿ってコーディングを行う.
- ソフトウェアの実装を行い、動作するプログラムを作成する.
- コーディング規約に従い、可読性や保守性の高いコードを書く.
- コードレビューを行い、品質を確保する.

構築のイメージ

テスト（Testing）

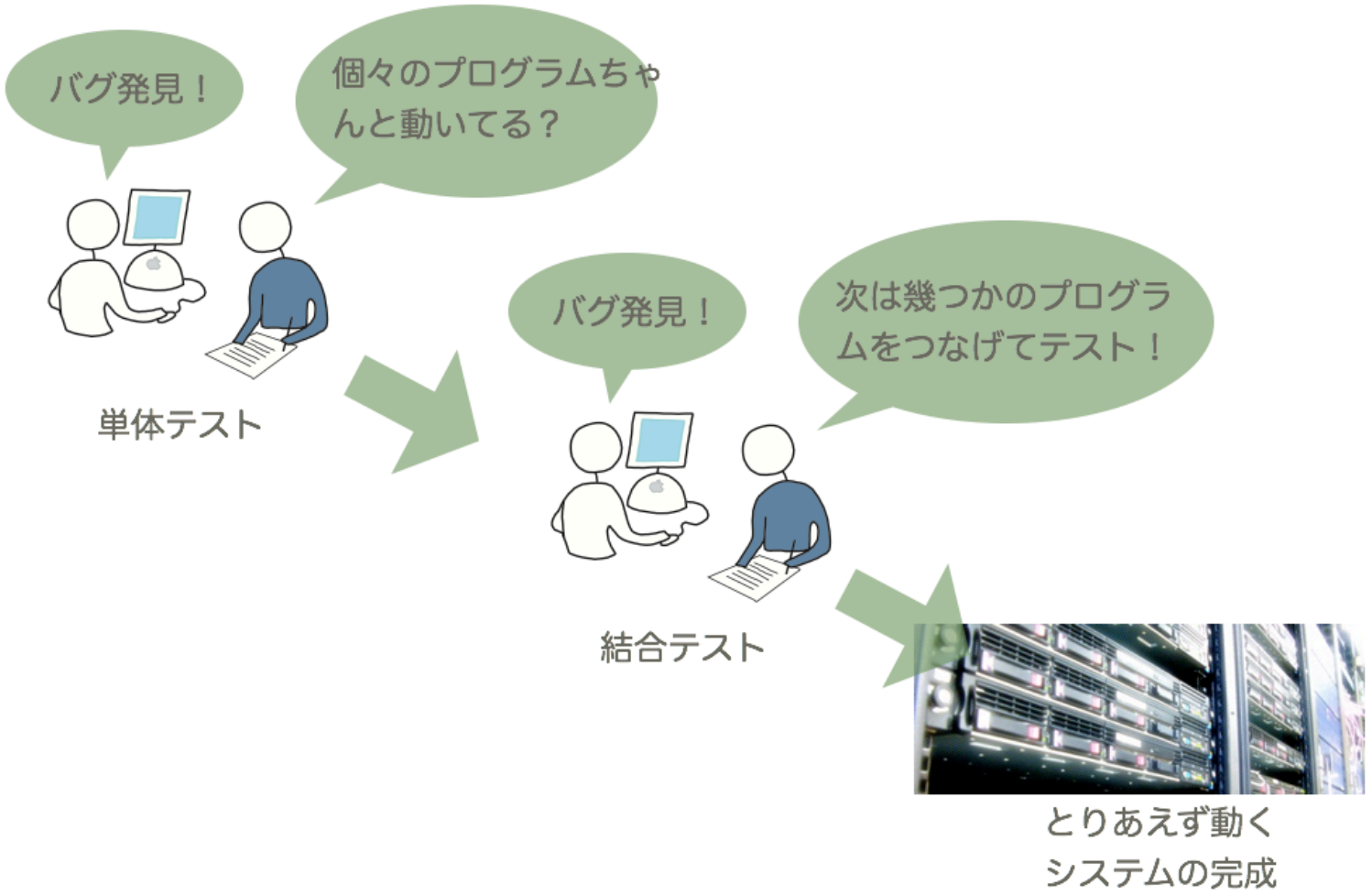
目的

- プログラム／システムが要求通り動作するかどうか確認する
- バグや不具合を検出し、修正する．
- テストを通じて、システムの品質を確保する．
- テスト工程は、単体テスト、結合テスト、システムテスト、受入テストなどに分かれる．

テスト工程

- 単体テスト（Unit Test）：単一モジュールの動作を確認する
- 結合テスト（Integration Test）：複数のモジュールを結合して，モジュール間のインタフェースなどが要求通り動作するかどうか確認する
- システムテスト（system Test）：実際の運用をシミュレートした環境で動作を確認する
- 受入テスト（Acceptance Test）：システムを受け入れるかどうか決定するためのテスト

テストのイメージ


バグ発見！
個々のプログラムちゃんと動いてる？
単体テスト
バグ発見！
次は幾つかのプログラムをつなげてテスト！
結合テスト
とりあえず動く
システムの完成

テストのイメージ

次は全体で仕様を満たしてるかどうかテスト！

ここまずいです！

システムテスト

受け入れテスト

いいよ

出来上がりました

最初の要求通りかテストお願いします

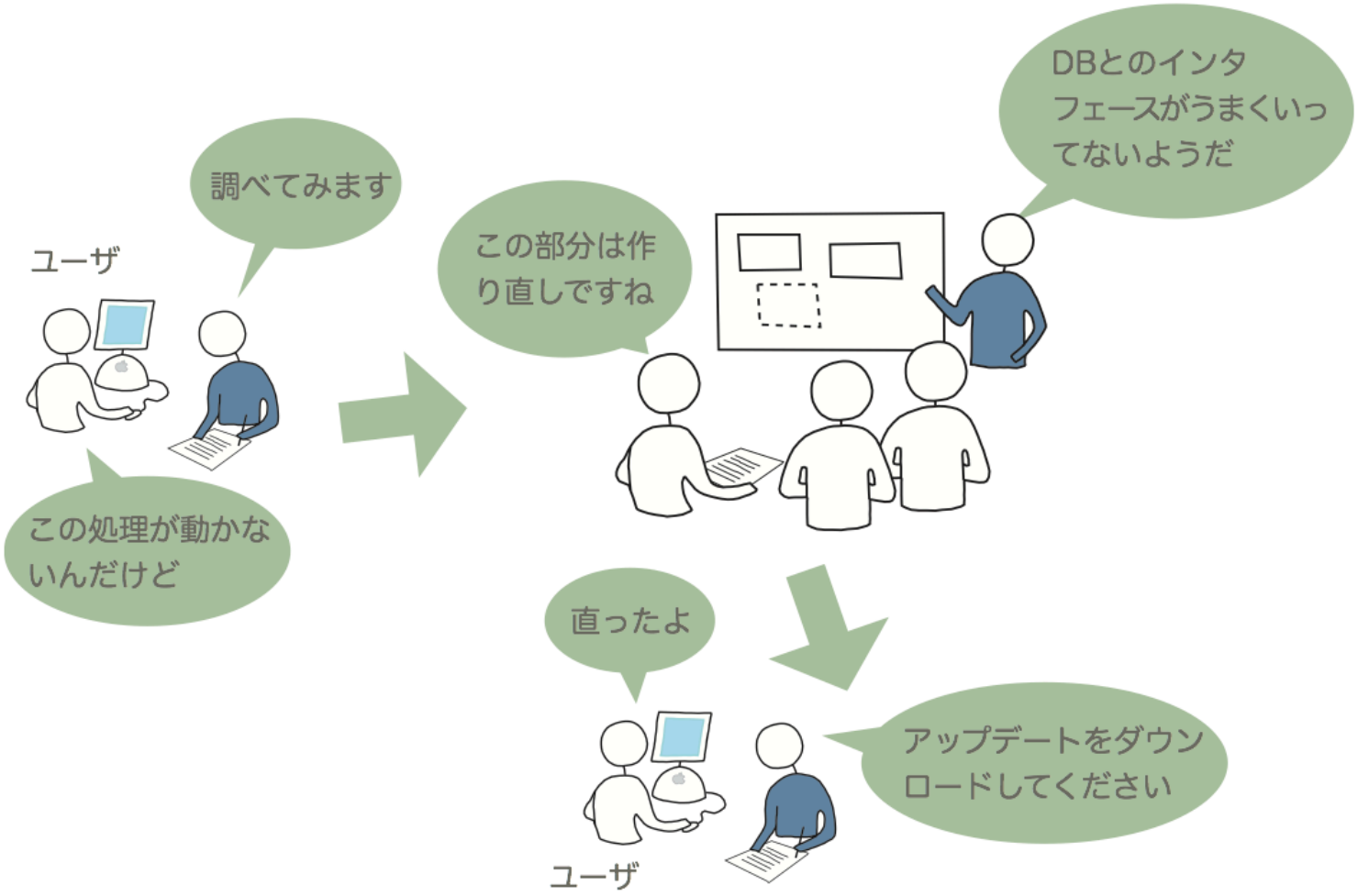
ユーザ

運用・保守

目的

- システムを実際に運用し、ユーザに提供する.
- 運用中の問題を監視し、必要に応じて修正や改善を行う.
- ユーザからのフィードバックを受けて、システムを継続的に改善する.
- 運用・保守を行うことで、システムの信頼性や安定性を確保する.

保守のイメージ


ユーザ
調べてみます
この処理が動かないんだけど
この部分は作り直しですね
DBとのインタフェースがうまくいってないようだ
直ったよ
アップデートをダウンロードしてください
ユーザ

まとめ

- ソフトウェア開発は、要求定義から運用・保守までの一連のプロセスを経て行われる
- 各工程での計画的な進行が、品質や納期、コストの管理に寄与する
- 開発プロセスを定めることで、チーム全体での共通認識が得られ、効率的な開発が可能になる

ソフトウェア開発モデル

ソフトウェア開発モデルとは

- 開発プロセスをどのように実行するかを定めた進め方のスタイル
- 各工程の順序や繰り返し、重点の置き方を体系的に示したもの
- 開発モデルを選ぶことで、プロジェクトの特性（例：変更の多さ、納期の厳しさ、規模の大きさ）に応じた最適な手法が適用できる
- 代表的なモデル：
  - ウォーターフォールモデル
  - アジャイル開発モデル（Scrum, XP など）
  - スパイラルモデル
  - V字モデル
  - etc.

開発プロセスと開発モデルの違い

観点	開発プロセス	開発モデル
定義	ソフトウェア開発の基本的な工程の流れ	そのプロセスをどのように進めるかの手法
役割	要求定義・設計・実装・テスト・運用などの工程	工程の順序や反復、柔軟性に着目
対象	要求定義 → 設計 → 実装 → テスト → 運用	ウォーターフォール、アジャイル、スパイラルなど

ソフトウェア開発モデルの必要性

- ソフトウェア開発は複雑であり、計画的かつ体系的な進行が求められる
- 開発モデルを導入することで、以下のメリットが得られる：
  - 効率的な進行：各工程の目的や手順が明確になり、無駄な作業や手戻りを減らせる
  - 品質の向上：テストやレビューの実施タイミングが明確になり、バグや設計ミスを早期に発見できる
  - リスク管理：仕様変更やトラブルが発生した際の対応方法（例：イテレーションの再実行）が事前に定義されている
- プロジェクトの特性（規模、変更頻度、納期の厳しさなど）に応じた**最適な開発モデルを選ぶ**ことで、プロジェクトの成功率を高められる

ソフトウェアの分類

- 組み込みソフトウェア
  - 自動車、家電製品、IoTデバイスなどに搭載されるソフトウェア
  - ハードウェアと密接に連携し、リアルタイム性が重要
- 業務用ソフトウェア
  - 企業の業務を支援するシステム（ERP、銀行システム、会計管理など）
  - 高い信頼性と保守性が求められる
- Web／モバイルアプリ
  - SNS、ECサイト、ニュースアプリなどのアプリケーション
  - ユーザーインタフェースや頻繁な更新に対応
- AI／機械学習システム
  - チャットボット、画像認識、予測分析などのシステム
  - データ駆動型であり、継続的な学習・改良が必要

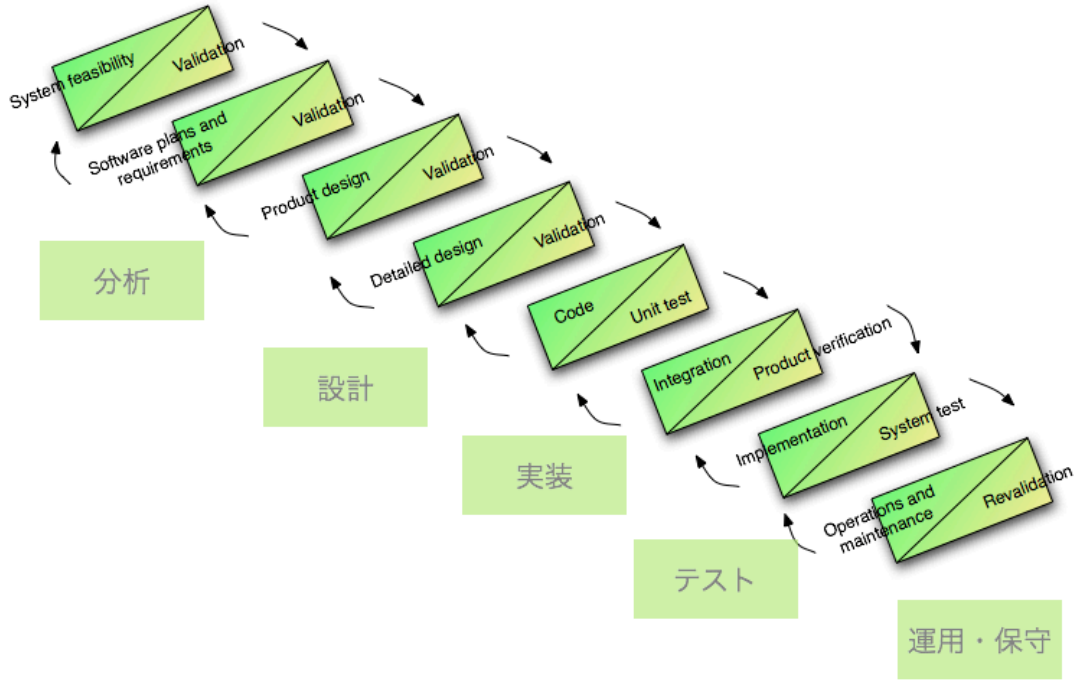
ソフトウェアの種類と開発モデルの対応

ソフトウェアの種類	よく使われる開発モデル	理由・背景
組み込みソフトウェア	ウォーターフォールモデル	ハード依存・仕様凍結が早い
業務用ソフトウェア	ウォーターフォール／スパイラル	大規模で信頼性・文書性が重視される
Web／モバイルアプリ	アジャイル開発	変化が早く、短期リリースが必要
AI／機械学習システム	アジャイル／スパイラル	試行錯誤・継続的学習が前提

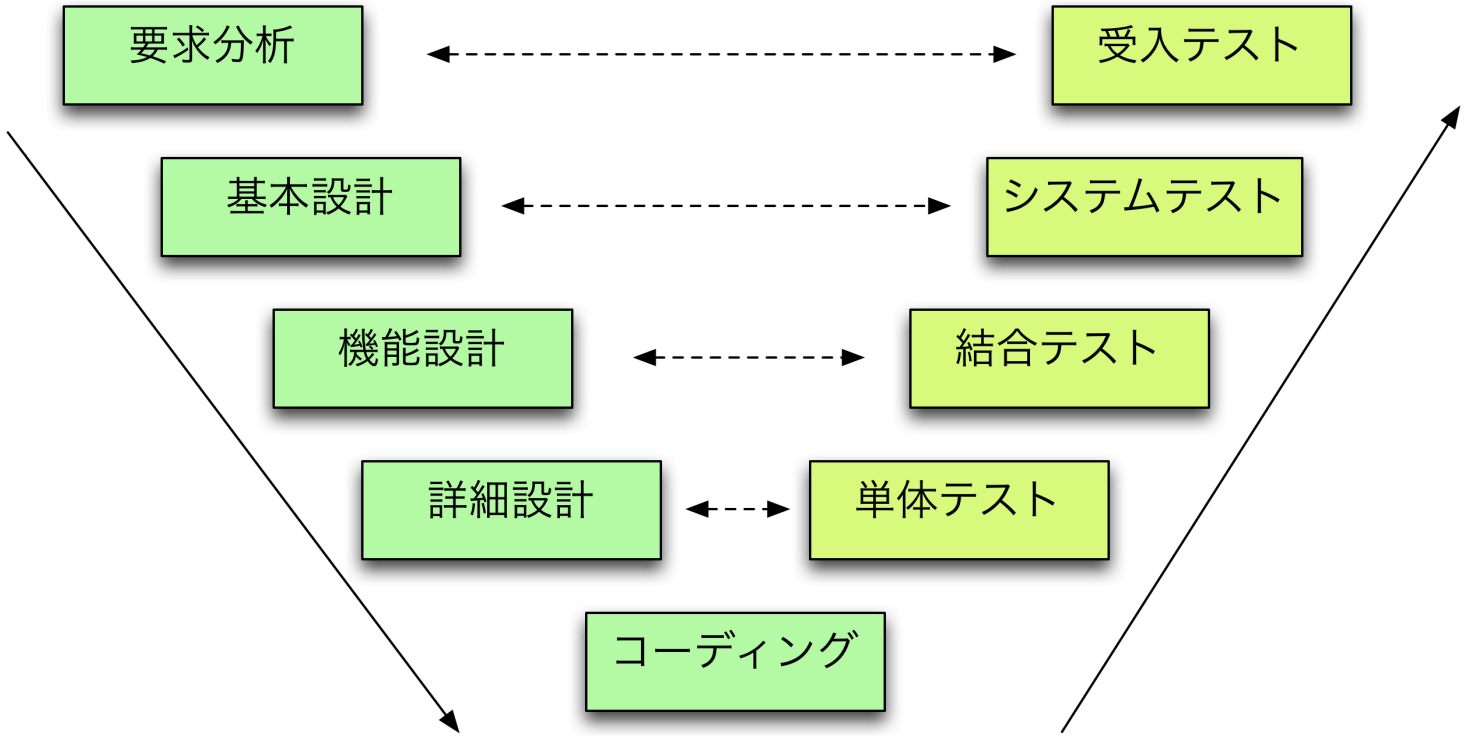
ウォーターフォールモデル（Waterfall Model）

- 1970 年代に誕生
- 特徴
  - 各工程を順に進める「段階的開発」
  - 工程間のフィードバックは基本的に少なく、前工程に戻りにくい
- 工程
  - 要求分析：ユーザ要求からシステムの仕様を決める
  - 設計：仕様を実現する構造や実装方法を決める
  - 構築：プログラムを作成する
  - テスト：設計や仕様通りに動作するかを確認する
  - 運用・保守：バグ修正や機能追加などを行い、システムを維持・改善する

- ドキュメント駆動プロセス（Document-Driven Process）
  - 各工程の成果物（設計書・仕様書など）をドキュメントとして残し、次の工程へ引き継ぐ
  - ドキュメントのレビューを通じて品質を保証する



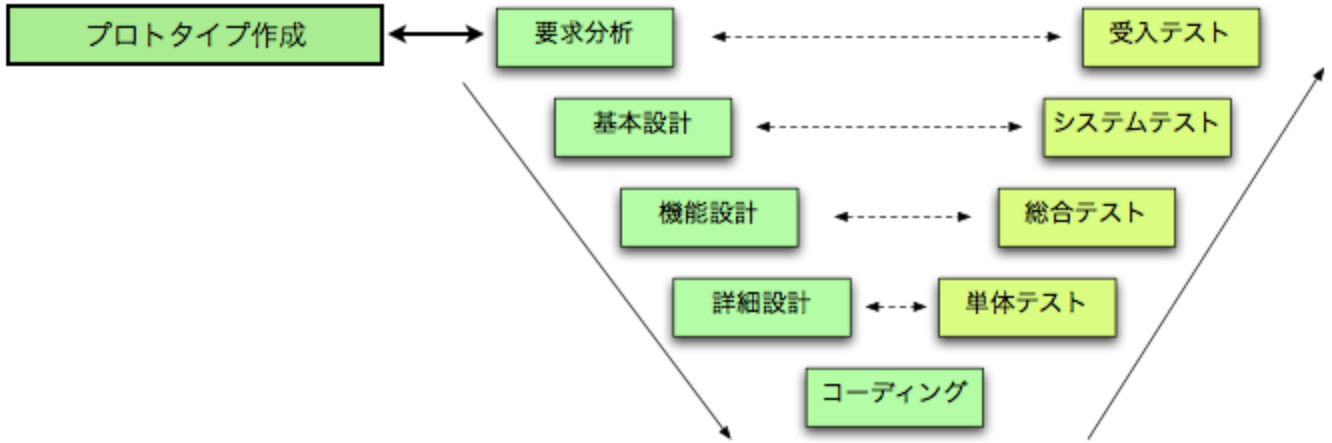
- V モデル（ウォータフォールモデルの変化版）
  - 各開発工程に対応するテスト工程が明示されており、要求仕様に対する受入テスト、設計に対する統合テストなどの関係が図で表される



- 長所
  - 各工程で，マイルストーン（達成目標）と成果物（この場合は成果物＝ドキュメント）があるのでスケジュール管理が行いやすい
  - 分業が容易
- 短所
  - リスク管理が難しい
  - ソフトウェア開発には必ず「戻り工程」が存在する
  - ユーザの本当の要求が完成まで見えにくく、リリース後の修正コストが高くなりやすい
- 対象例
  - 大規模で高い信頼性が要求されるシステム

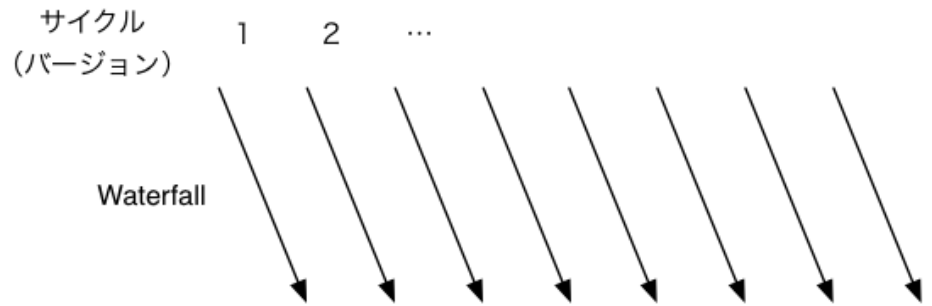
プロトタイピングモデル（Prototyping Model）

- ウォーターフォールにおける「手戻り」リスクを低減
- 最終的に運用するソフトウェアシステムを作る前に，実験的なシステム（プロトタイプ）を素早く作り、ユーザと仕様のすり合わせを行う
- 特に「要求分析」段階（フェーズ）で行うことが多い
- どの程度の「プロトタイプ」を作成するかが難しい（短所）



インクリメンタルモデル（Incremental Model）

- 小さな機能範囲のシステムから作成
- 改良（バージョンアップ）によるシステム構築
- 典型的には「確実に必要な機能」から作成していく
  - プロトタイピング：**不確実な要求を明確化する**
  - インクリメンタル：**確実な機能から順に開発し、段階的に完成に近づける**
- 設計全体を初期段階で明確にしないと、後の段階で統合が困難になり、コードが複雑化する（短所）のでリファクタリング（再構築）が必要になる場合がある



モデルの比較（まとめ）

モデル	特徴	利点	課題・注意点
ウォーターフォール	段階的に進む（文書重視）	管理しやすい、文書整備される	柔軟性が低い、戻りに弱い
プロトタイピング	試作品による要求確認	要求の明確化が早期にできる	プロトタイプの扱いが難しい
インクリメンタル	機能単位で段階的に開発	優先度に応じて進められる	設計の一貫性維持が難しい

アジャイル開発

アジャイル開発とは

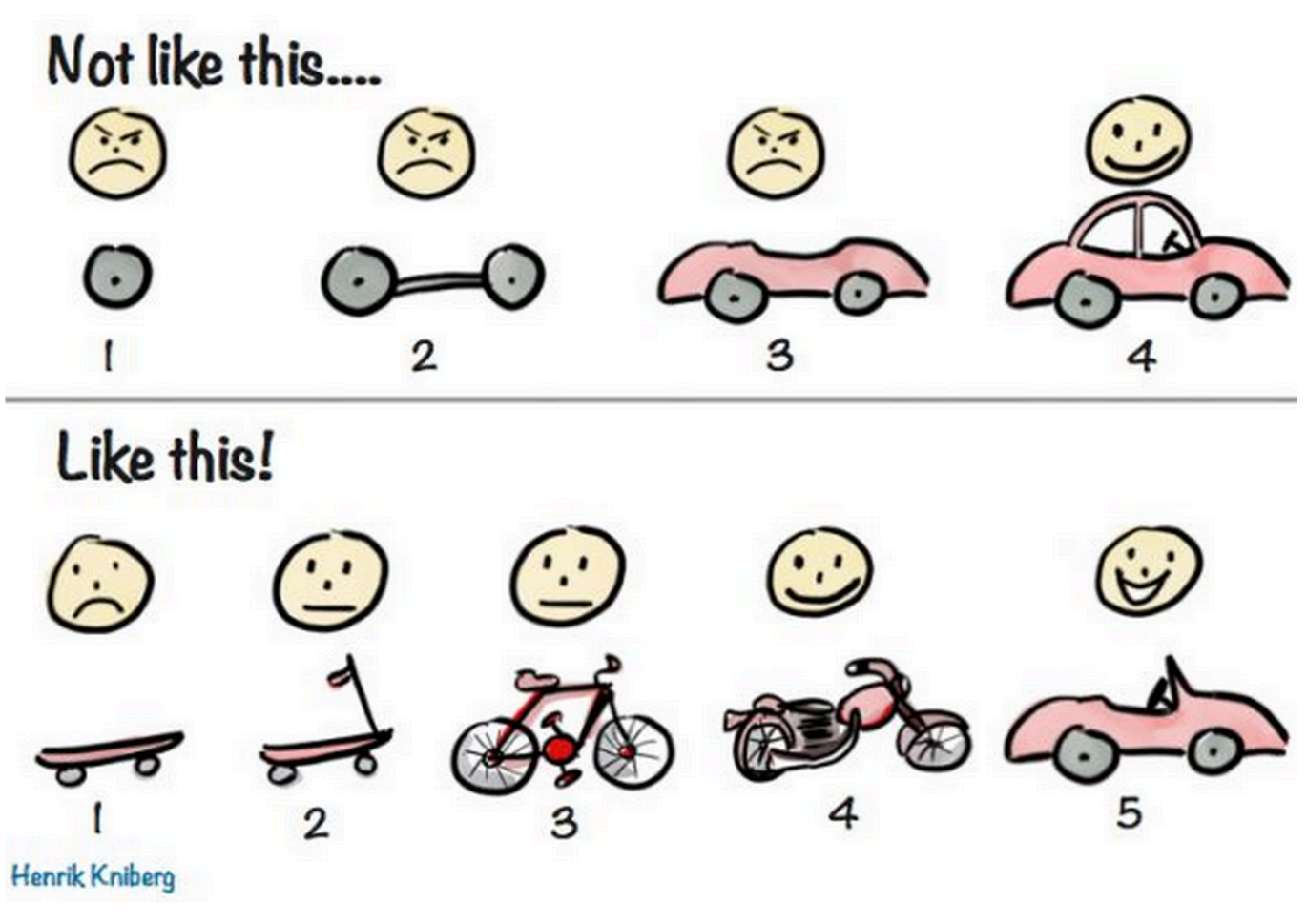
- アジャイル（agile）：すばやい，俊敏な
- 短い開発期間（イテレーション）を単位として反復的に開発を行う
- 顧客要求の変化や不確実性に柔軟に対応し、リスクを最小化することが目的

特徴

- 顧客とエンジニアで1つのチームを構成し、常に要求をすり合わせながら開発を進める
- 優先度の高い要求から順に実装を進める
- 開発期間単位毎に実装・テスト・リリースを行う→実際に動くソフトウェアを通じて顧客と確認し、次の開発に反映する

ウォーターフォールとの比較

	ウォーターフォール	アジャイル
要求定義	最初にすべての要求を文書化して確定	要求は変化する前提で開発を進める
チーム	工程毎やモジュール毎に分業する.	顧客を巻き込んだチームで 一つのプロダクトを作成する.
検証	完成後に一括で確認する	短期間ごとに動作確認を繰り返す



The Myth of Incremental Development by Herding Cats

アジャイル開発に向いているケース

- 小規模ソフトウェア・機能が分離できるもの（プラグインなど）
  - 1チームで巨大なものを作成するのが難しい
- 品質よりもマーケットシェアが重要なソフトウェア（オンラインアップデートできるものなど）
  - 信頼性の作り込みが難しい

➡️ Webサービス，スマホアプリ

※現時点では組み込みなどにはあまり向いていない（と言われている）

Scrum

- アジャイル開発手法の代表例の一つ
- 作業，会議，成果物を定めたもの

特徴

- 要求の高いものから実現する（プロダクトバックログ）
- 短い期間で目標・実施・検査・改善を行う（スプリント）
- 動作するもので要求の確認を行う（スプリントレビュー）
- プロダクト：ユーザ要求を満たす「完成物」
- プロセス：チームの作業を円滑に進めるための進行方法

役割

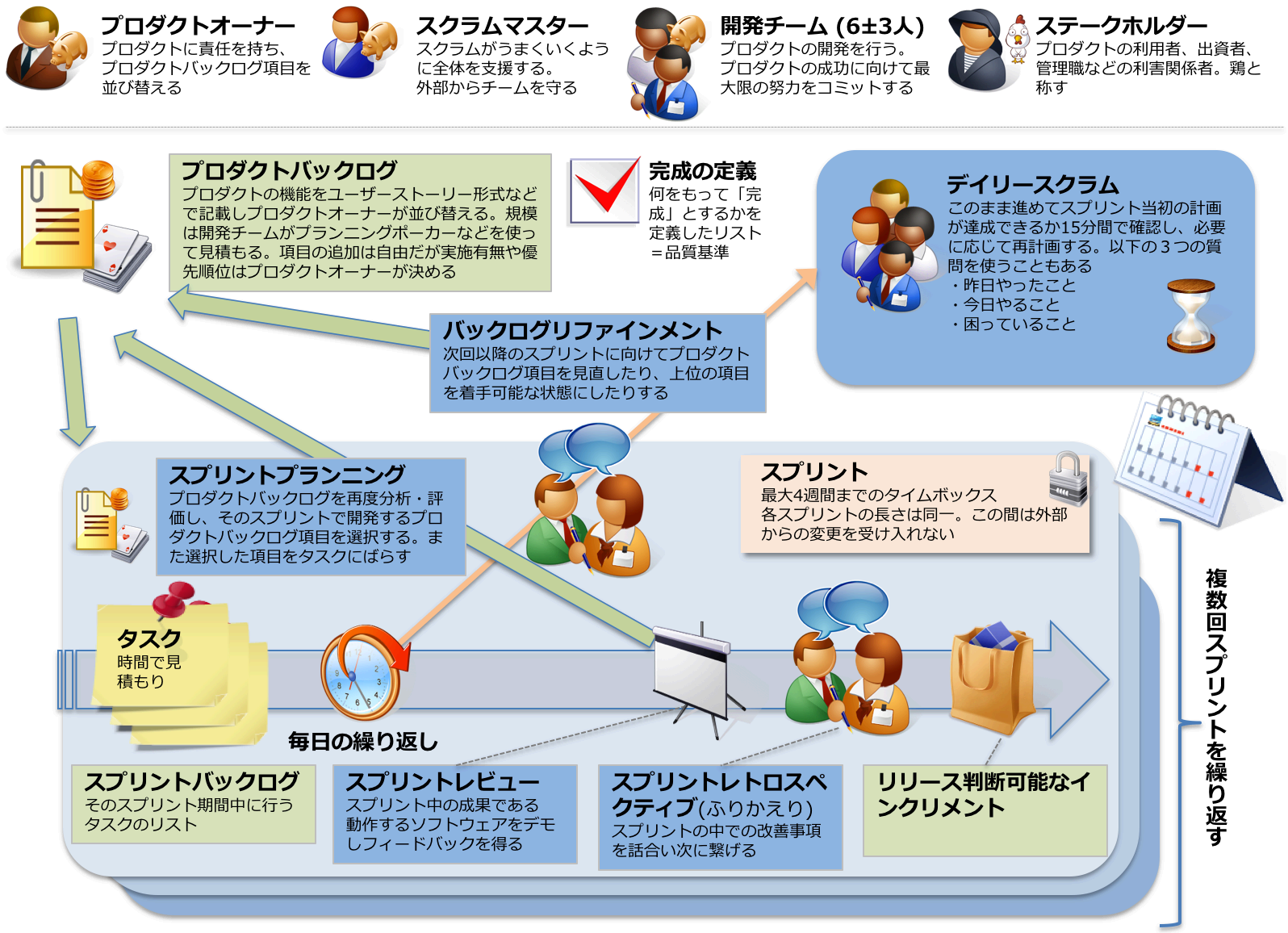
- プロダクトオーナー：プロダクトの責任者．プロダクトバックログの管理．
- 開発チーム：開発をする人．上下関係はなし．
- スクラムマスター：教育・ファシリテータ

成果物

- プロダクトバックログ：プロダクトへの要求一覧．開発中も変化する．
- スプリントバックログ：スプリントで行うタスクリスト．
- リリース判定可能なプロダクト：「動く」ソフトウェアプロダクト．

会議

会議名	内容
スプリント計画	今回のスプリントで何をするか決定．スプリントバックログの作成
デイリースクラム	毎日の短い進捗確認（15分）．リスク（スプリントの目標を達成できなくする要因）の発見
スプリントレビュー	完成物を共有・フィードバックを得る．リリース判定可能なプロダクトのデモ．プロダクトバックログの確認・更新．
スプリントレトロスペクティブ	ふりかえり．プロセス観点で KPT (Keep, Problem, Try)/YWT（わかったこと，やったこと，つぎにやること）．



Scrumのメリット

- 変化に強い：顧客の要求変更に柔軟に対応できる
- コミュニケーションの活性化：デイリースクラムやレビューでチーム内の情報共有が進む
- 早期リリースとフィードバック：スプリントごとに動作する成果物を提供
- 継続的な改善：スプリントごとに振り返り、プロセスを改善
- 自己組織化チーム：チームが自律的に判断し、主体的に開発を進められる
- 進捗の見える化：成果物とバックログにより、状況が把握しやすい

Scrumのデメリット

- 導入ハードル：既存の開発プロセスからの移行には意識改革が必要
- 運用コスト：定期的なミーティングに時間とリソースが必要
- 習熟が必要：フレームワークの理解不足では効果が出にくい
- 全員の積極的な参加が前提：関与が弱いとスクラムの利点が活かせない
- プロジェクトの適性が必要：組み込み系や長期的な仕様固定型には不向きな場合もある
- チームの特性に依存：文化・価値観・スキルが合わないと機能しにくい

全体のまとめ

- ソフトウェア開発プロセスは、**計画的・体系的な開発**を実現するための手順
- 開発モデルは、プロセスを**どのように進めるかを定めたスタイル**
- ウォーターフォールモデル、プロトタイピングモデル、インクリメンタルモデルなどが代表的な開発モデル
- アジャイル開発は、**変化に柔軟に対応**し、**顧客とのコミュニケーション**を重視する開発手法
- スクラムはアジャイル開発の一つで、**短い開発期間で反復的に開発**を進める手法
- 開発モデルの選択は、プロジェクトの特性に応じて**最適なものを選ぶ**ことが重要