INTERNET ACADEMY

Institute of Web Design & Software Services

システム開発基礎２

インターネット・アカデミー

システム開発手法

- 開発手法（ウォーターフォール）
- 機能要件と非機能要件
- 開発手法（アジャイル）
- 開発手法ごとのメリットとデメリット

ウォーターフォール開発

INTERNET ACADEMY
Institute of Web Design & Software Services

ウォーターフォール開発とは

ソフトウェア工学では初期に考案されたもっともポピュラーな開発モデル。

- 古いと揶揄されるが現在においてもよく使用される
- 慣れた案件でリスクが少ない場合は最も効率的な手法
- 1社で完結しない開発案件では全体としては大抵この形になる

- ✓ 原則として前工程が完了しないと次工程に進まない
- ✓ 前工程の成果物の品質をチェックして後工程に引き渡すことで、前工程への後戻り（手戻り）を最小限にする

ウォーターフォール開発

INTERNET ACADEMY
Institute of Web Design & Software Services

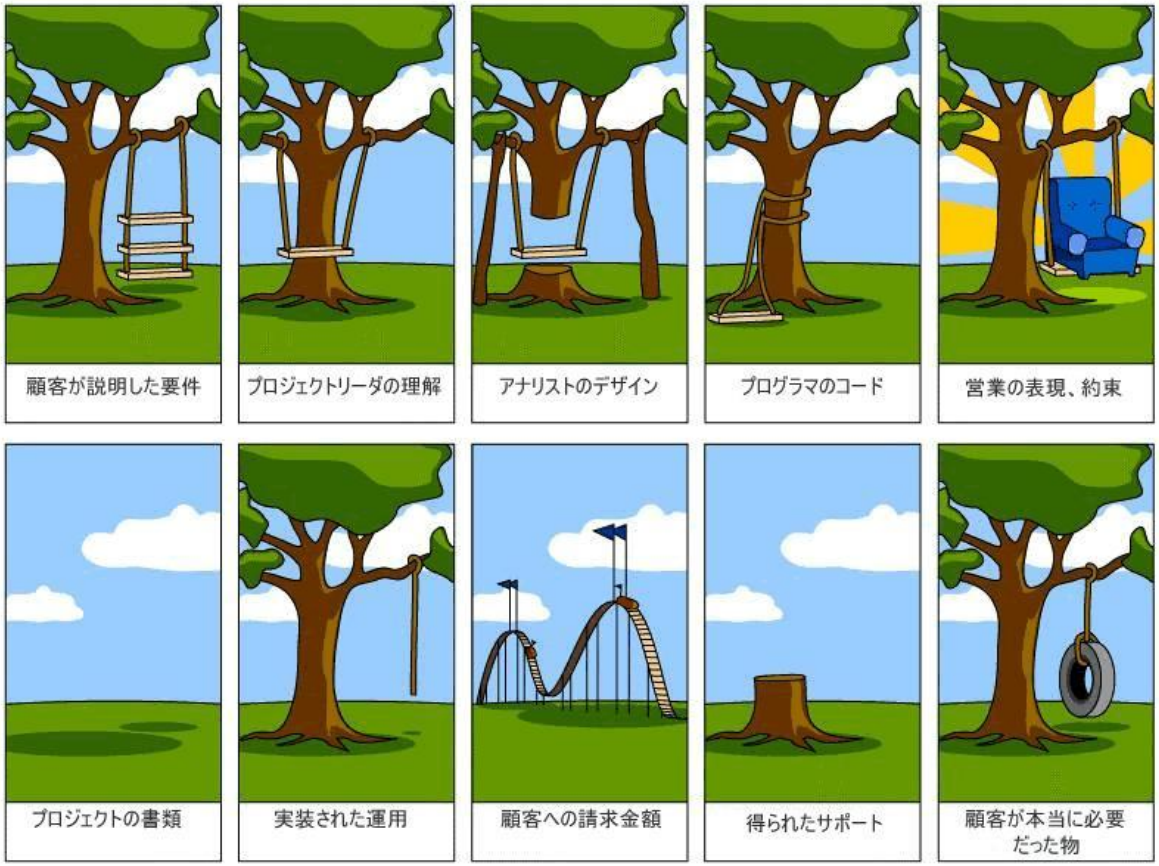
要件定義	顧客からの要望をヒアリングする
基本設計	どんなものを作るか(機能、画面、データベースなど)
詳細設計	具体的にどう作るか (プログラム書く直前)
開発	プログラミングする
単体テスト	ファイル単位、最小機能単位でのテスト
結合テスト	複数ファイル(複数機能)を連携させたテスト
運用テスト	本番環境と同じ状況でのテスト

要件定義のミス

INTERNET ACADEMY
Institute of Web Design & Software Services

要件定義　何のために、何をいつまでにいくらでつくるかなど

営業、PMのミスは後から挽回が難しい



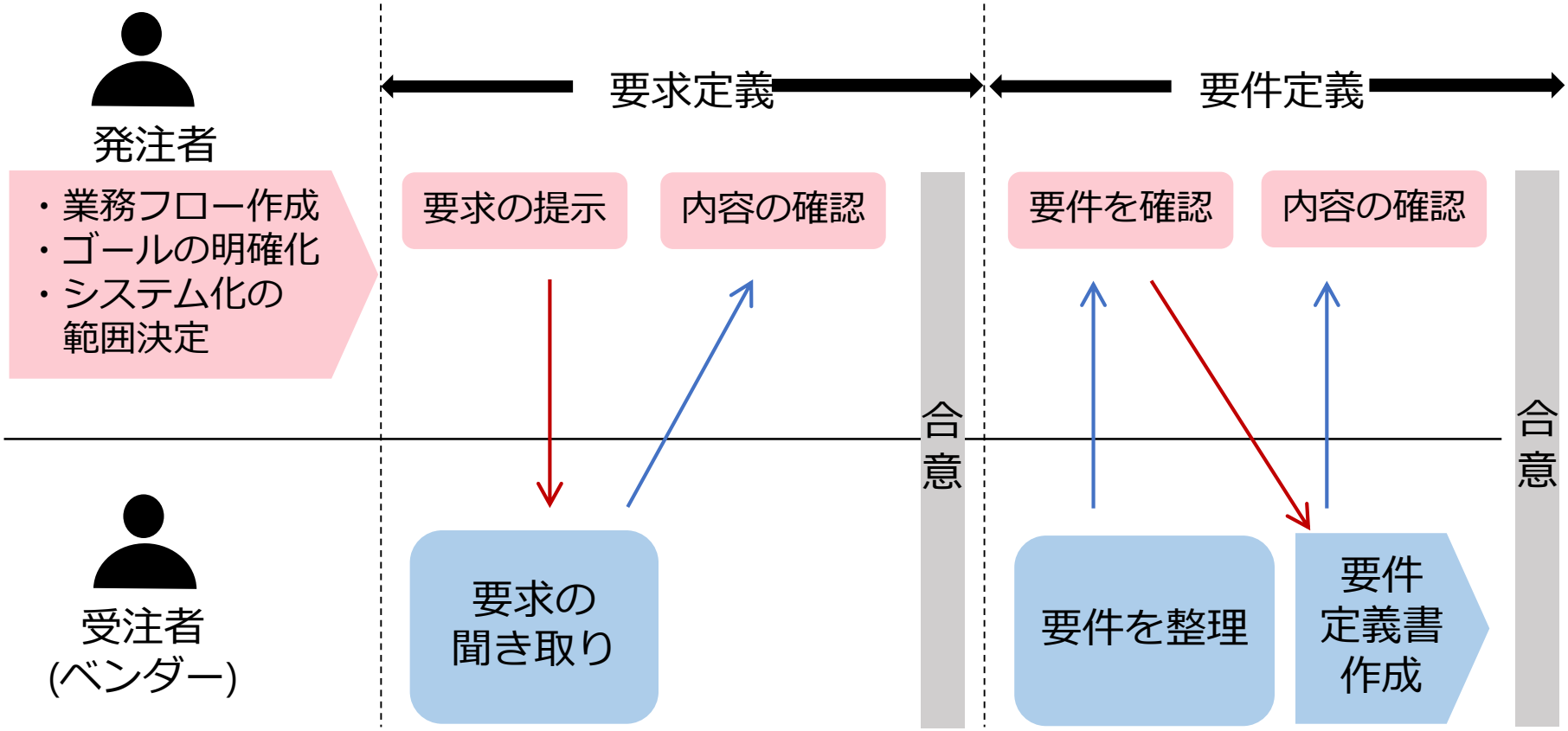
出典：
アンサイクロペディア

ウォーターフォール開発

要件定義　何のために、何をいつまでにいくらでつくるかetc

（1）**背景**：システム化に至った経緯。例）法改正のため、規模拡大のためなど
（2）**目的・方針**：背景を受けて導入する目的。例）業務効率化、一元管理など
（3）**概要**：システムの概要、構成する機器、動作環境（インフラ）など
（4）**機能**：欲しい機能、性能、満たして欲しい条件など
（5）**システム化する範囲**：予算と期間と対応能力から、出来る範囲を決める。
（6）**UIのイメージ**：見た目や使い勝手がシステムの価値を左右することも
（7）**運用・保守**：システムの稼働時間や、障害発生時の対応をどうするかなど
（8）**スケジュール**：納期、プロジェクト体制や連絡方法、打ち合わせ日程など
（9）**成果物**：作成する成果物、納品する内容や方法など

要求定義と要件定義



ウォーターフォール開発

INTERNET ACADEMY
Institute of Web Design & Software Services

基本設計　どんなものを作るか(機能、画面、データ)etc

（1）**業務フロー**：システムを利用する際のフロー。フロー図etc

（2）**機能一覧**：システムに実装する機能の一覧やグループ分け

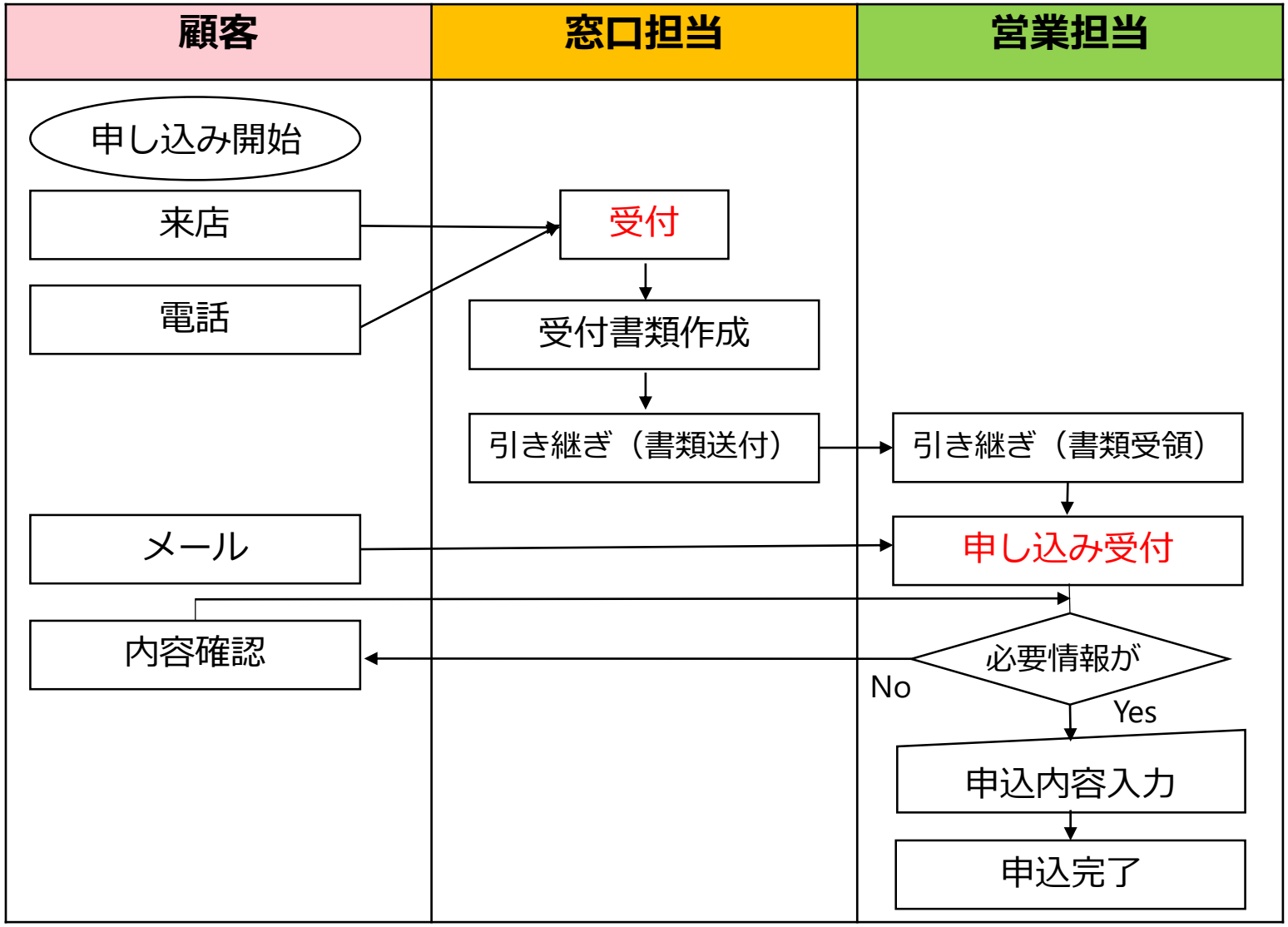
（3）**画面(帳票)レイアウト**：各種レイアウト定義。デザイナーが作成

（4）**画面遷移図**：各機能の画面の遷移図。ボタンラベルやメニュー名も記載

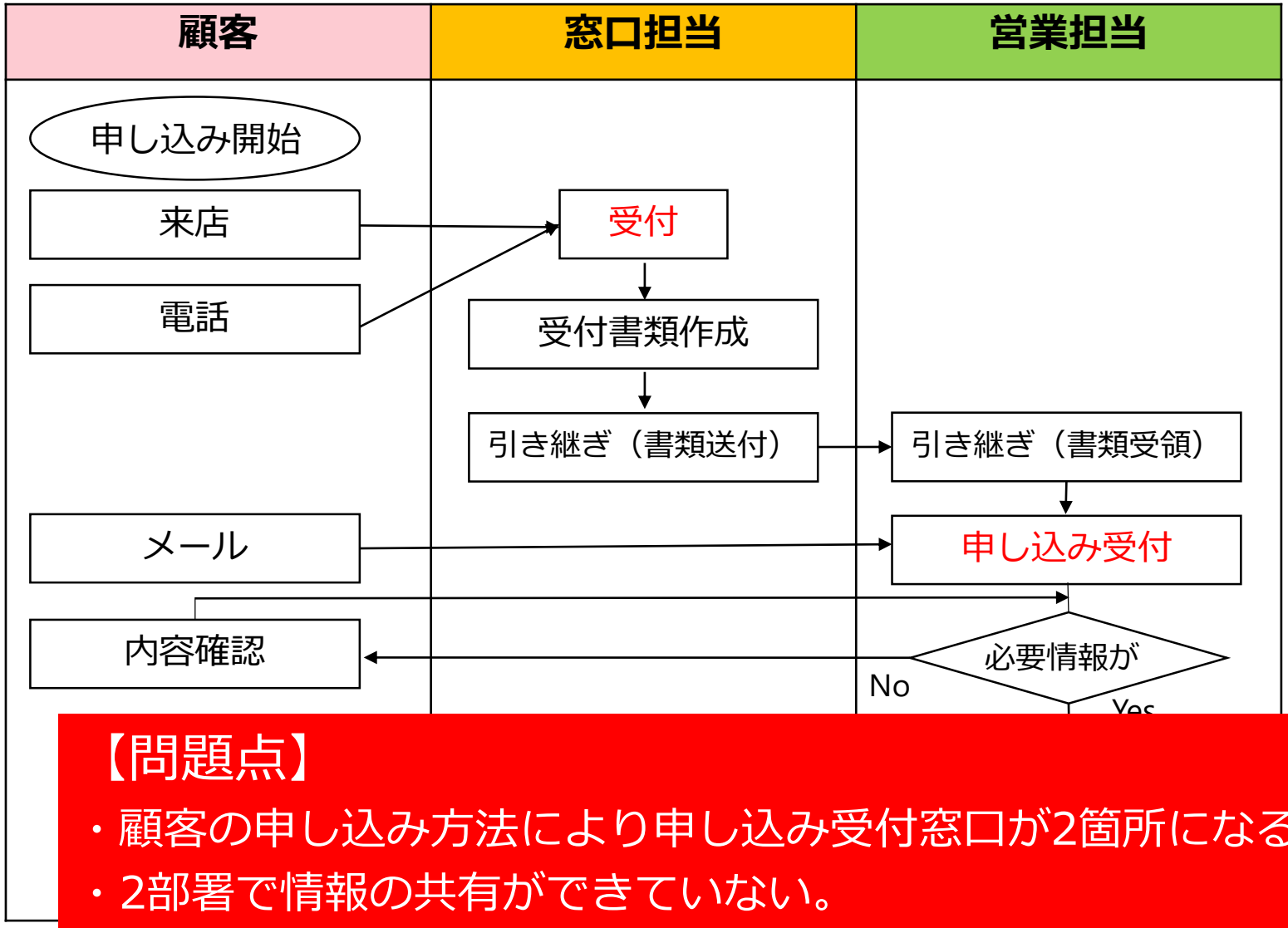
（5）**ER図**：Entity Relationshipの略。DBの要素と関係性を示す図

（6）**インタフェース一覧**：外部システムとの連携仕様など

アクティビティ図で問題を見つける



アクティビティ図で問題を見つける

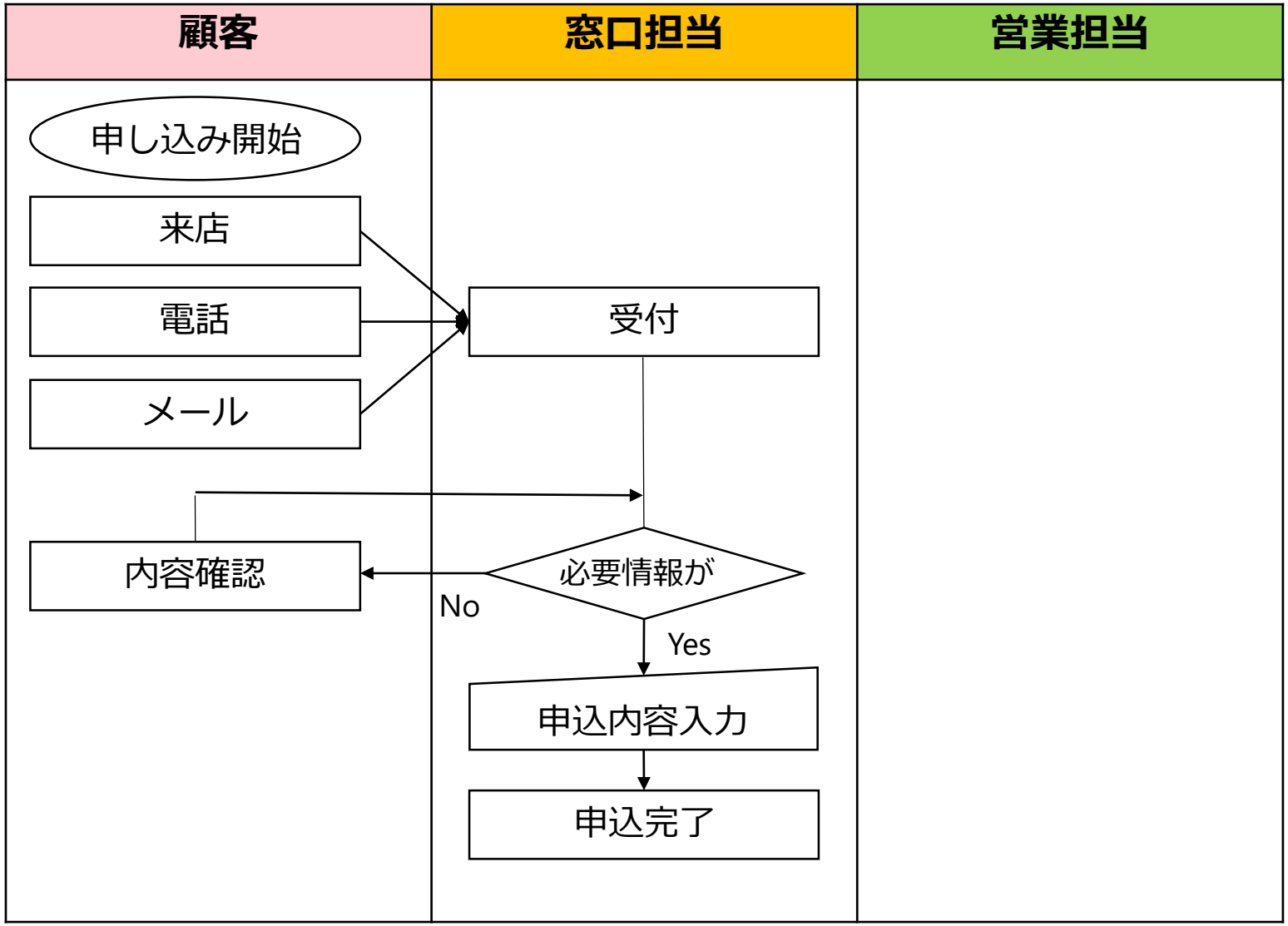


【問題点】

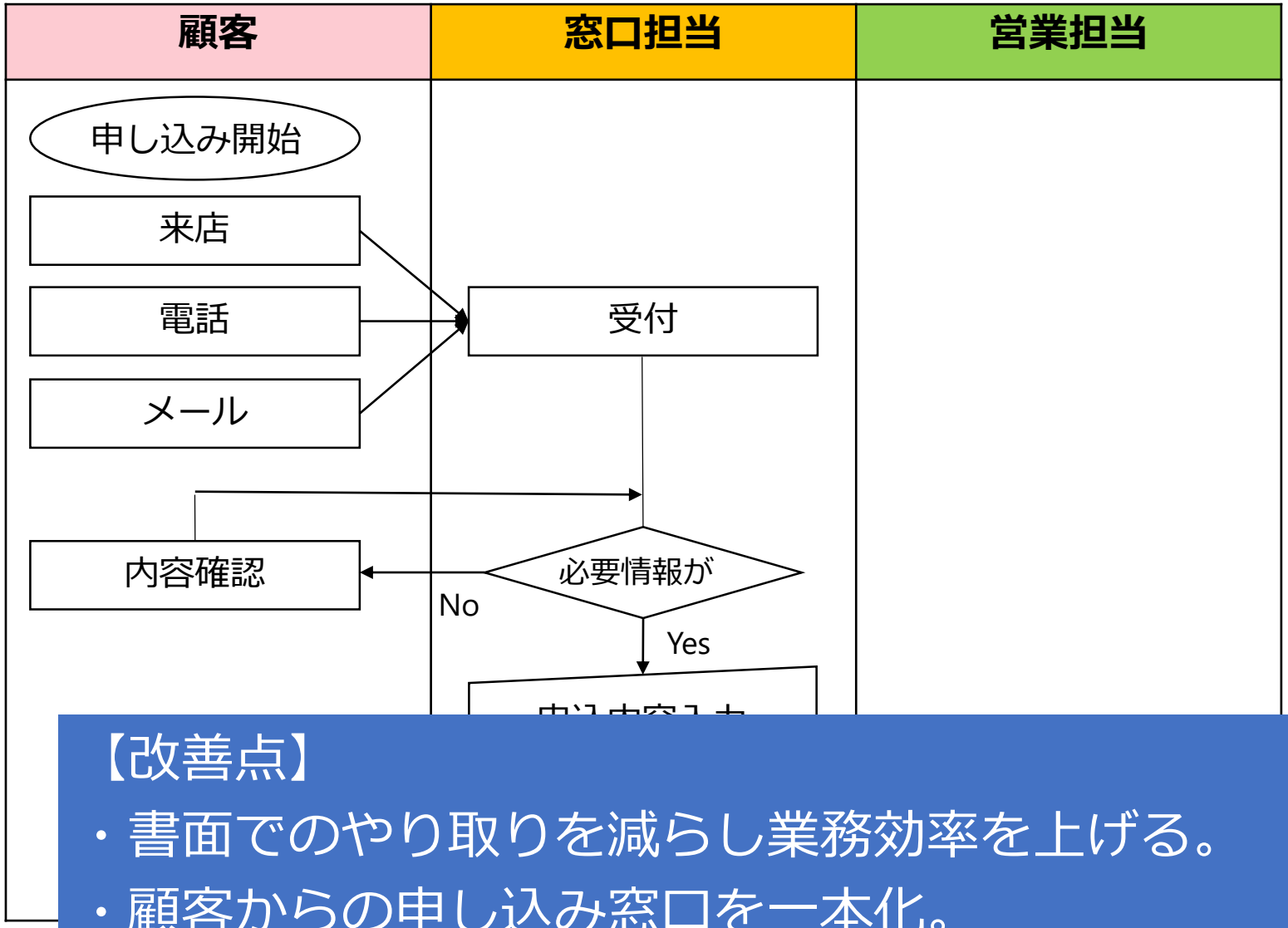
- 顧客の申し込み方法により申し込み受付窓口が2箇所になる。
- 2部署で情報の共有ができていない。
- 顧客からの申し込みに関する問い合わせ対応に時間がかかる。

アクティビティ図で問題を見つける（改善）

INTERNET ACADEMY
Institute of Web Design & Software Services



アクティビティ図で問題を見つける（改善）



【改善点】

・書面でのやり取りを減らし業務効率を上げる。

・顧客からの申し込み窓口を一本化。

ウォーターフォール開発

INTERNET ACADEMY
Institute of Web Design & Software Services

詳細設計　どのように作るか (プログラムを書くための計画)etc

（１）機能分割定義：クラス図

（２）データフロー：業務の流れをもとに、データの流れを図にする

（３）画面設計書／バッチ処理設計書（モジュール設計書）：

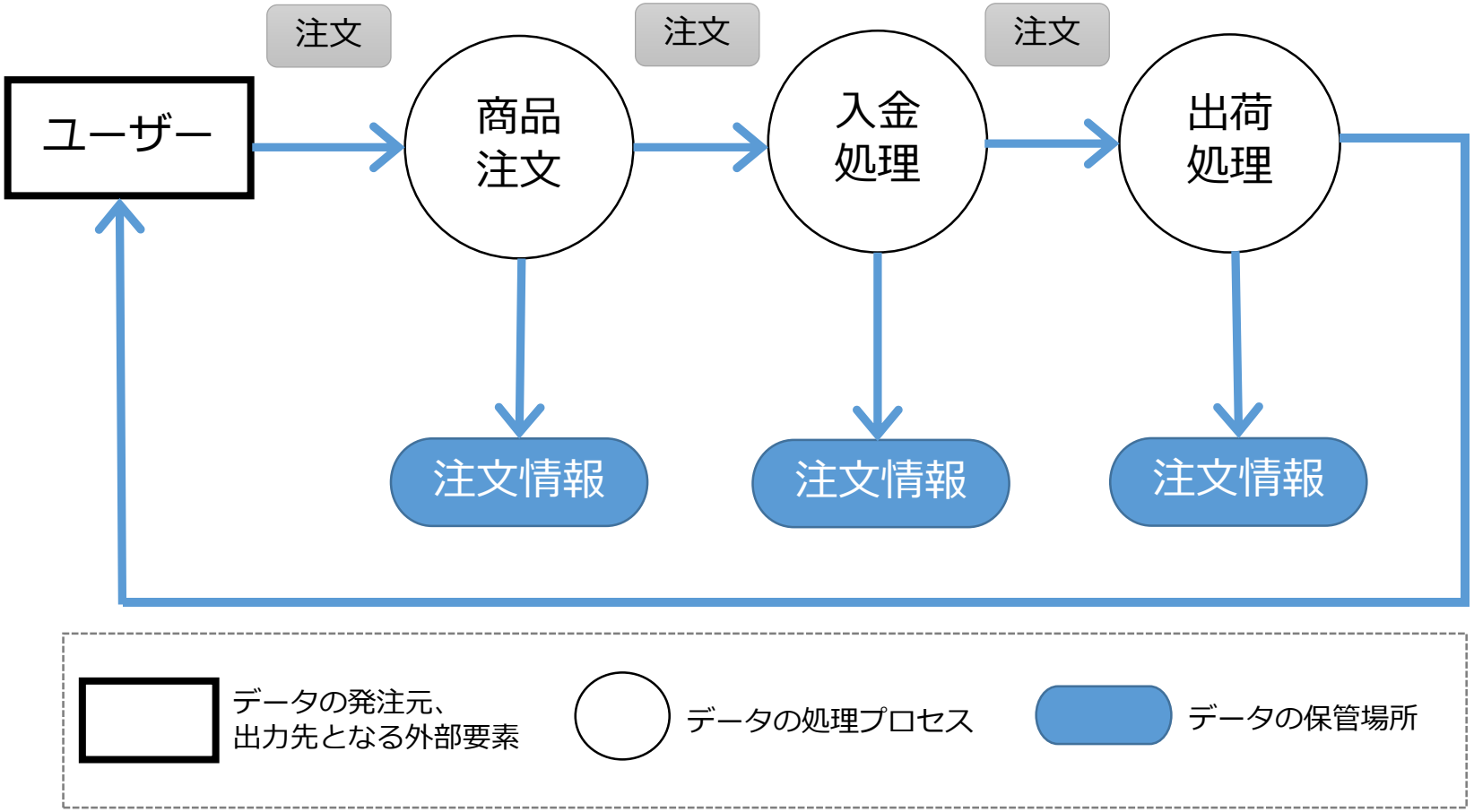
モジュール名称、役割、引数・戻り値、処理概要など

（４）DB関連：

テーブル定義書、モジュール名称、役割、引数・戻り値、処理概要など

データフロー図

■データフロー図の例



上流工程のよくある課題

INTERNET ACADEMY
Institute of Web Design & Software Services

- 見切り発車
  - 作れるかどうかわからない要件がある
  - 間に合うかどうかわからない要件がある
- リスクが洗い出せていない
  - 発注側が決めかねている要件がある
  - 発注側の変更可能性が高い要件がある
- 要件以外の情報の周知
  - 言語化、文書化できていない担当者の状況の共有

    例）上場、オフィス移転、担当者の退職、社名の変更、組織改編

上流工程のよくある課題

INTERNET ACADEMY
Institute of Web Design & Software Services

- 大規模な案件の問題
  - 全体像が俯瞰できない
  - 各モジュールの繋がりに不安
  - 成果物の完成形がイメージしづらい
  - 非機能要件を満たせるかどうか知見が無い

対策のためにどうするべきか

- 問題を早期に発見する/あぶりだす
- 不確定要素がいつまで不確定で良いか決めて順守/決断する

下流工程のよくある課題

INTERNET ACADEMY
Institute of Web Design & Software Services

特徴

- いざ開発したときに設計のまずい点や漏れが露呈される
- 問題に対応するのための時間と手段が限られている

対策のためにどうするべきか

- 問題を早期に発見する/あぶりだす
- 不確定要素がいつまで不確定で良いか決めて順守/決断する

システム開発手法

- 開発手法（ウォーターフォール）
- **機能要件と非機能要件**
- 開発手法（アジャイル）
- 開発手法ごとのメリットとデメリット

機能要件と非機能要件

いずれも、「要件定義」のフェーズで定める

・機能要件

顧客がシステムに求める機能のこと。作ってもらいたいシステムそのもの

例：ログイン機能がほしい。ユーザーがパスワードを忘れたら、
　　メールアドレスだけわかればリセットするようにしてほしい

・非機能要件

顧客がシステムに求める「できて当然」の度合い

例：２４時間、３６５日ノンストップで稼働し、
　　メンテナンスのために休止しない

機能要件

機能要件とは、**顧客がシステムに求める機能**のこと。

機能要件は「画面」「帳票」「処理」に分類される。

種類	説明・例
画面	表示される画面 例：販売製品や販売履歴が見れる画面が欲しい
帳票	PDFやCSVなどダウンロードするもの 例：製品納品の納品書を自動作成して欲しい
処理	自動化したい処理 例：月末の支払業務を自動化して欲しい

機能要件の書き方

要件定義の初期段階で洗い出す

- 顧客に完成形のイメージがあれば文章化してもらう
  - 提案依頼書（RFP [Request For Proposal] ）に記入してもらう

- 完成形のイメージがない場合はヒアリングしてまとめる
  - 多くの場合は担当者がシステムに詳しくないのでこのケース

機能要件の書き方

背景や目的

システム化する上で一番重要な項目。

例

- 法制度が変化したため
  - 顧客の同意取得機能、一定期間の保管とバックアップ、金融庁への報告データ出力
- 業務コスト削減したい
  - 入力作業の軽減、分析作業の自動化、コミュケーションの円滑化
  - 顧客管理の効率化、行動データの蓄積と売上データとの連携など
- 売上を増加させたい
  - 見込み顧客のリストアップ、連絡漏れの防止、定期メール送信
  - 追加契約のための需要予測機能、類似商品のレコメンドなど

機能要件の書き方

必要な機能を洗い出す

顧客が求める機能をヒアリングするところから。

1. ヒアリングしたら図や表にまとめる。
2. 画面や帳票のサンプルを作る。
3. よくある画面フォーマットに合わせてイメージのすり合わせをする

認識の齟齬を無くすため、業務フロー図にまとめる

機能要件の書き方

機能要件の選定

洗い出してまとめた機能要件を絞り込む。

実際には予算と納期の問題があるため。営業とＰＭの腕の見せどころ。

機能ごとの概算工数（費用）を提示し、機能削減した際には代替案をつけることもある。

代替案の例

・利用頻度の低い画面や帳票は、Excelやメールで作業する

・画面の一部の便利機能（例えばグラフ表示機能）を外す

・開発費用の少ない簡易機能を提供する

非機能要件とは

顧客が機能面以外にシステムに求める要件のこと。

例
稼働率、システム性能、セキュリティ、保守サービスの有無と品質など

非機能要件がおざなりなシステムの例

- タスクを実行するのに長い時間待たされる
- よく停止する
- 使い勝手が悪くいつもサポートにクレームがくる
- 機能追加やバグ修正が難しい
- セキュリティに問題がある

機能要件と大きく異なる点は、主に発注元に対してヒアリングして要件を洗い出していくものではないということ。

非機能要件の注意点

一般的に発注元はシステムはまず止まらないなど「非機能要件」に対して大きな期待をしていることが多い。

運用フェーズに入ったあと「非機能要件」が自分たちの考える水準に達していない場合は大抵トラブルになる。

トラブルを避けるためにも発注元、ベンダー双方で「非機能要件」について設計時に念入りに精査することが重要。

非機能要件の種類

INTERNET ACADEMY
Institute of Web Design & Software Services

IPA(独立行政法人情報処理推進機構)が公開している「非機能要求グレード」では以下の6つに非機能要求を分類している。

① **可用性**
運用時間やメンテナンス時間、システムの稼働率や災害時の対応など

② **性能/拡張性**
性能目標(動作速度や過負荷、高頻度、最大容量データ投入時の安定動作など)や機能各省の可否など

③ **運用/保守性**
バックアップ、マニュアルの整備、ログなどのシステム監視など

④ **移行性**
現システムからの移行のスケジュールやデータの統合、移行など

⑤ **セキュリティ**
ウイルス対策や不正アクセス対策などセキュリティ対策

⑥ **システム対策/エコロジー**
ハードウェアが環境に与える影響や耐震性など

他にもUX(ユーザーエクスペリエンス＝ユーザーがシステムを使ったときに得られる経験や満足の度合い)なども非機能要件に含まれる。

非機能要件の種類

IPA(独立行政法人情報処理推進機構)が公開している「非機能要求グレード」では以下の6つに非機能要求を分類している。

分類	概要	要件項目の例
①可用性	システムの継続利用	・障害や災害時における稼働目標
②性能・拡張性	システム性能、将来の拡張	・画面レスポンス ・データ増加
③運用・保守性	運用と保守のサービス	・稼働時間 ・データバックアップ ・システム監視 ・計画停止 ・サポート体制

非機能要件の種類

IPA(独立行政法人情報処理推進機構)が公開している「非機能要求グレード」では以下の6つに非機能要求を分類している。

分類	概要	要件項目の例
④移行性	現行システムからの移行	・移行スケジュール ・移行方法 ・移行データ
⑤セキュリティ	セキュリティの確保	・認証機能（ログインなど） ・機能制限 ・データの暗号化
⑥環境・エコロジー	設置環境や規格	耐震や温度、湿度、騒音など

非機能要件の書き方

機能要件と同じく、要件定義の初期段階で洗い出す

機能要件に比べると、検討する上での難易度が高い

・分かる範囲で提案依頼書（RFP ）に記入してもらう

・基本的にはヒアリングしてまとめる

非機能要件の書き方

１．まずは想定する非機能要件を挙げておく

非機能要件グレード(IPA)

＜モデルシステム＞
・社会的影響がほどんと無いシステム
・社会的影響が限定されるシステム
・社会的影響が極めて大きいシステム

＜可用性の継続性レベル＞
レベル１：定時内（9時～17時）
レベル２：夜間のみ停止（9時～21時）
レベル３：1時間程度の停止（9時～翌8時）
レベル４：若干の停止あり（9時～翌8:55）
レベル５：24時間無停止

非機能要件の書き方

２．非機能要件を顧客と設定する

顧客と話しながら、稼働時間や性能や災害時の対応などについて決める。

「稼働時間は、9時～21時ではなく、8時～翌5時にしてほしい」

注意点：
理由を聞いておくこと（代替案や変更可能性や変更頻度を知るため）

例） 稼働時間が8時～翌5時の理由
・勤務時間が9時からのため。
・夜勤スタッフがいるため。

非機能要件の書き方

３．非機能要件の選定

機能要件と同様に、実際には予算と納期の関係で満たせない要件もある。

例
以下のふたつの非機能要件では、どちらがシステム開発の負荷が大きいか？またどのような機能が必要になるか？

①稼働時間が6時～22時のシステム

②24時間フル稼働のシステム

非機能要件の書き方

３．非機能要件の選定

例

①稼働時間が6時～22時のシステム

→止めている間にメンテナンスができる

②24時間フル稼働のシステム

→システムを止めずにメンテナンスする必要がある

・サーバーをもう一つ用意して切り替える
・システムダウン時に自動的に切り替わる仕組みを用意必要性が高まる
・データバックアップを定期的に取る必要性が高まる
・運用保守の難易度が上がるための負担増加

→ インフラエンジニアの作業領域。クラウド活用が増えている。

システム開発手法

- 開発手法（ウォーターフォール）
- 機能要件と非機能要件
- **開発手法（アジャイル）**
- 開発手法ごとのメリットとデメリット

アジャイル開発

INTERNET ACADEMY
Institute of Web Design & Software Services

アジャイル開発とは

ソフトウェア工学において迅速かつ適応的に開発を行う軽量な開発手法群の総称。

- 非営利組織 Agile Alliance が開発手法を推進している
- 2000年あたりから採用する企業の増加している

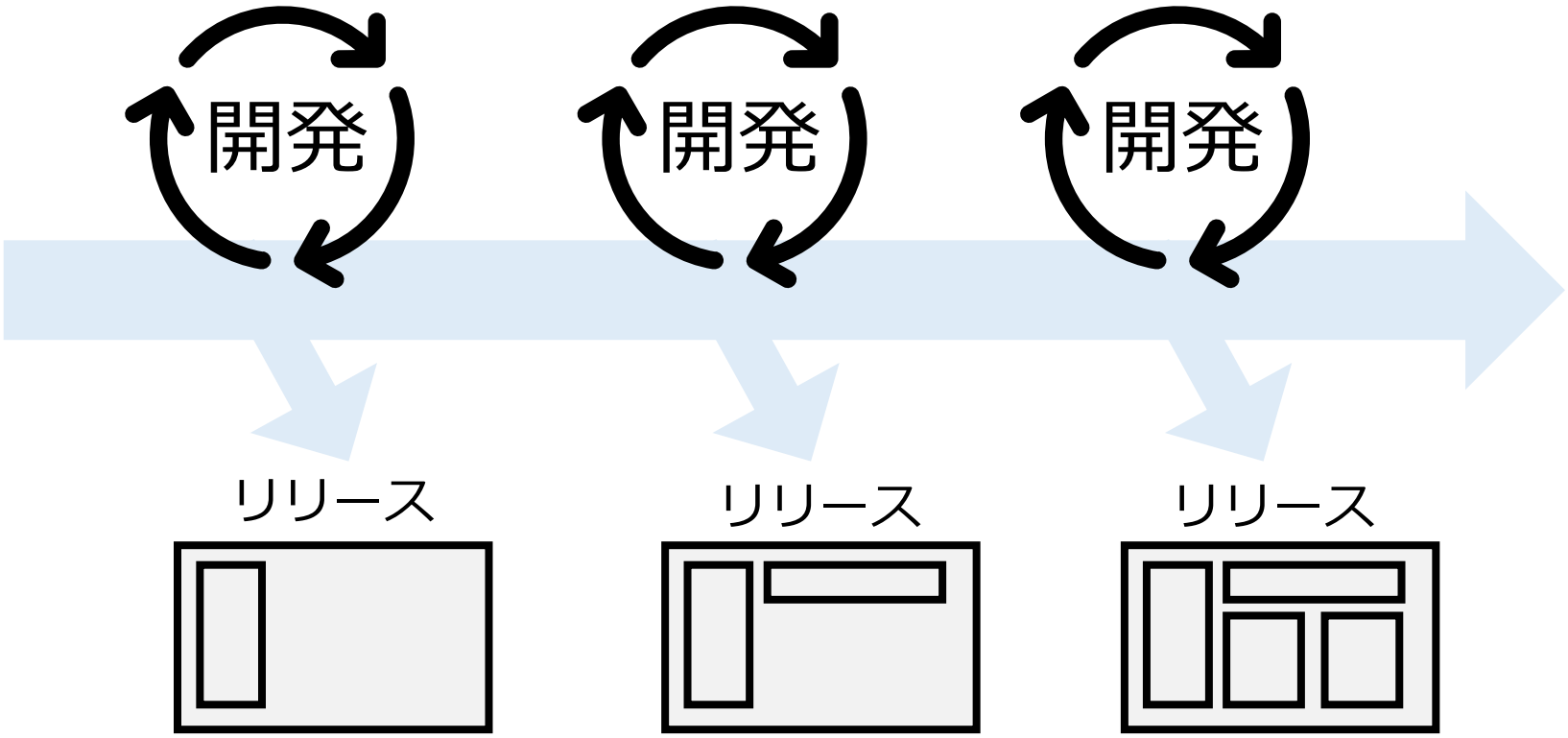
- ✓ ジャイルソフトウェア開発手法の多くは、反復 (イテレーション) と呼ばれる短い期間単位を採用することで、リスクを最小化する。

- ✓ 1つの反復の期間は、プロジェクトごとに異なるが、1週間から4週間くらいが多い。

INTERNET ACADEMY
Institute of Web Design & Software Services

アジャイル開発とは

反復 (イテレーション) は1週間から4週間が一般的。



アジャイル開発

INTERNET ACADEMY
Institute of Web Design & Software Services

アジャイル開発の特徴

- 定期的に計画を調整する
- 一度にまとめてすべての機能を作らない
- 1つの期間ごとにいくつかの機能を追加開発する
- 少人数チームで、一か所に集まって開発
- チームが目的の達成のために協力して進める
- 一度決定した計画を絶対のものにはしない
- 各メンバーや関係者からのフィードバックをプロジェクト進行に組み込むことで現場の意見を反映する

アジャイル開発

INTERNET ACADEMY
Institute of Web Design & Software Services

種類

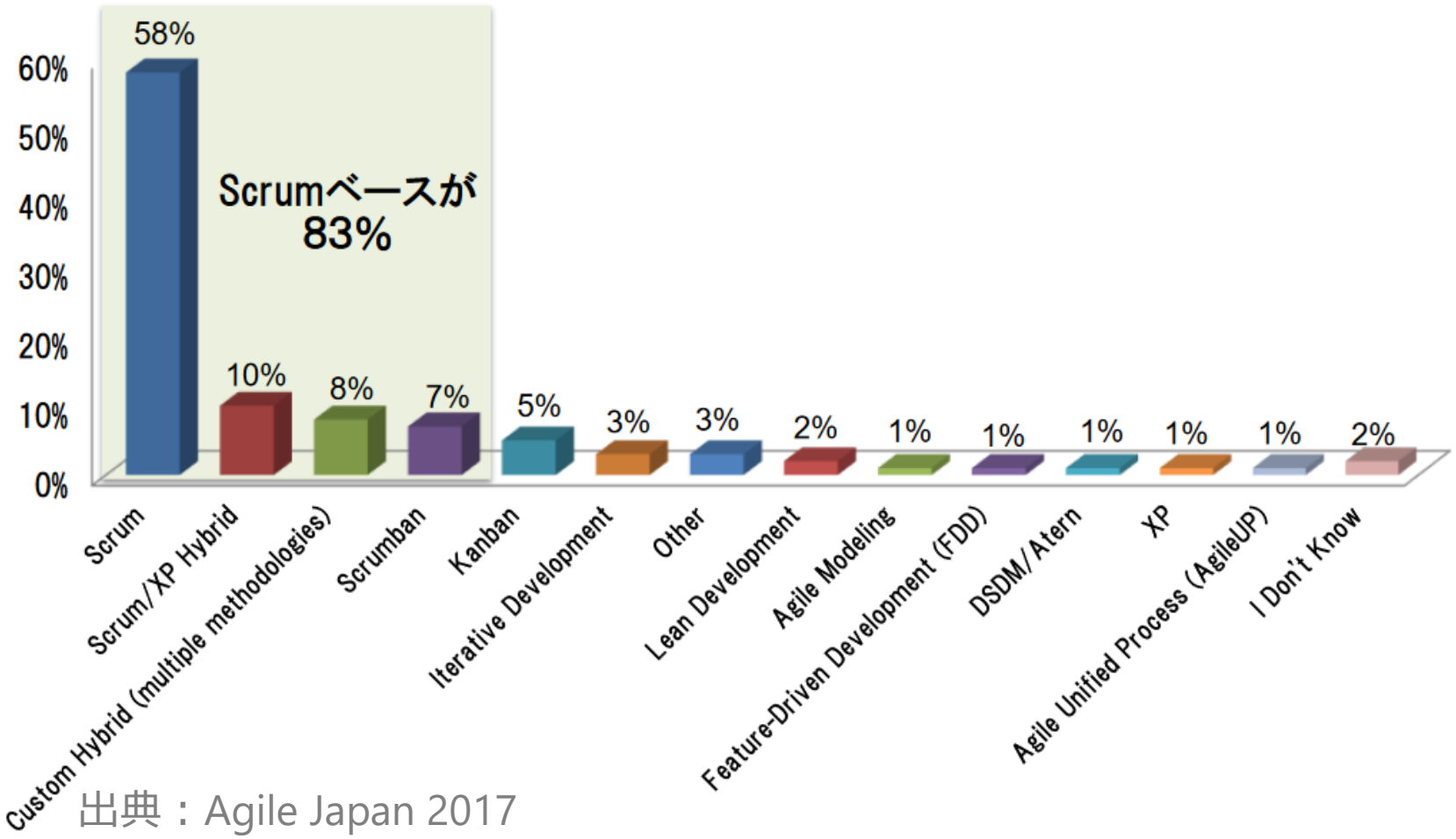
- **スクラム** (1986)
- リーン開発
- エクストリーム・プログラミング (**XP**) (1996)
- ユーザ機能駆動開発 (**FDD**; Feature Driven Development)
- Adaptive Software Development
- Dynamic Systems Development Method (DSDM) (1995)

他多数

アジャイル開発

INTERNET ACADEMY
Institute of Web Design & Software Services

アジャイルで最も利用されている手法はScrum



出典：Agile Japan 2017
https://www.agilejapan.org/2017/img/session/document/F-3_hybrid_agile_session_all_for_web.pdf

スクラム開発の語源

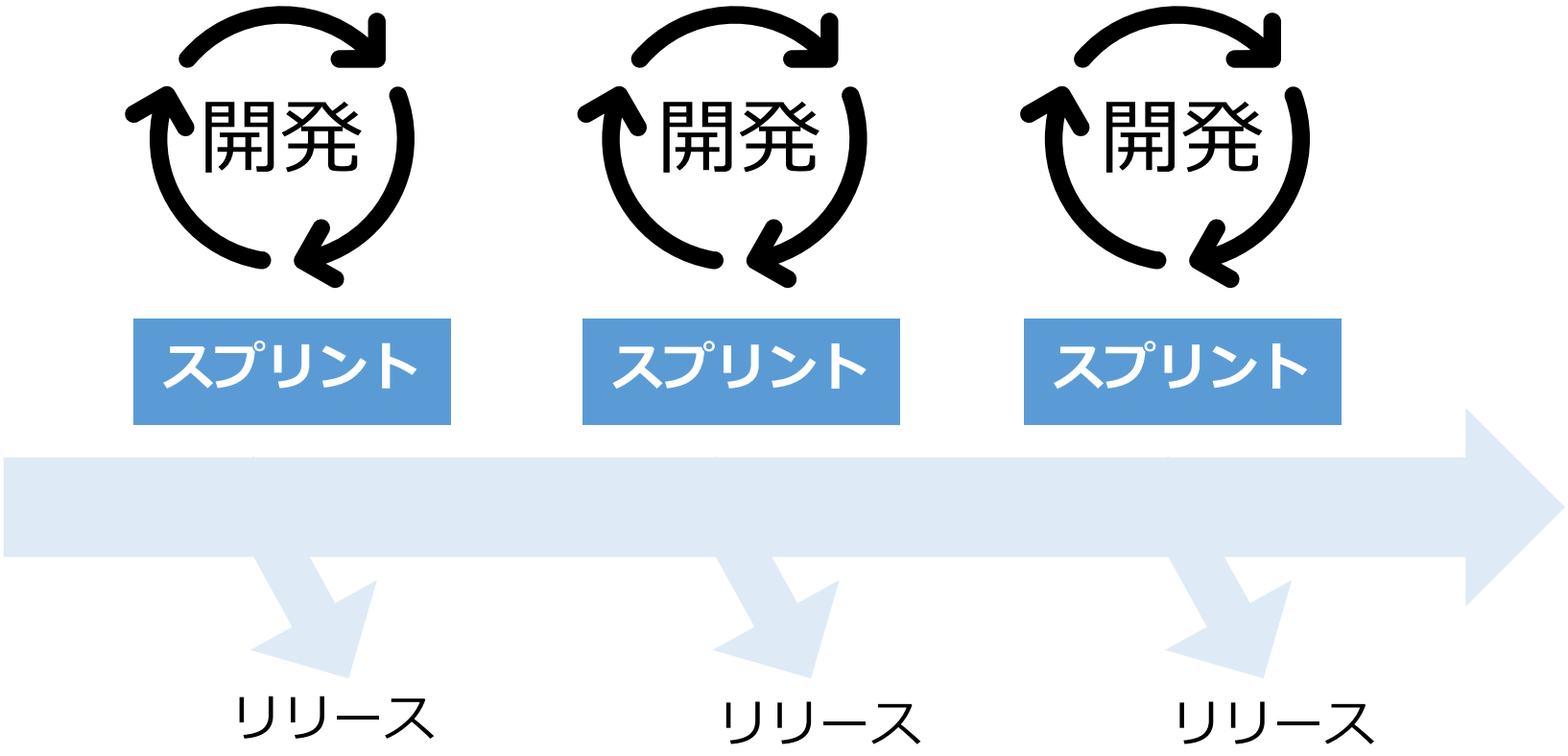
ラグビーで両陣が、8名ずつ肩を組んで一つの集団を作り、ぶつかり合う際のフォーメーション「スクラム」が語源。
その姿から集団が力を合わせることを”スクラムを組む”と表現することもある。

チームメンバーの成長やチーム自体の成長も目指すのが特徴。
チームマネジメント技法でもある。

アジャイル開発（スクラム開発）

スプリントの流れ

スクラム開発では、開発の1つの期間単位のことを「スプリント」と呼ぶ



スクラム開発の優れた点

■**短い期間で、最大限の成果をあげられる**

開発の際、優先順位の高い機能から順に開発を進める

⇒短い期間でも成果が上げやすい

■**作業の工数見積もりが正確になる**

開発では短い期間ごとに開発を区切って計画を立てるため、

作業の工数見積もりは比較的正確になる。

（数週間先なら予想しやすい）

ウォーターフォール開発のデメリットでよく挙がるのが「プロジェクトの工数見積もり」が　多すぎる/少なすぎる、というもの。
（プロジェクトの最序盤で数か月・数年先のことを見積もるため）

スクラム開発の優れた点

■自立的なチーム作りができる

ポイント

アジャイルでは各開発メンバーが自分で計画を立てて期限を切るため、セルフマネジメントができるようになっていく

- ✓ 自分のタスク以外も確認する場を設けるため、チームの成果のためにどうしたらいいかを全員で考えるようになる
- ✓ 個人視点からチーム視点になり、視野が広がり全体最適を考えられる
- ✓ 相互チェックによりチーム内で発生している問題の検知が早くなる
- ✓ 日々ミーティングの場を設けるため、「困っていること」や「分からないこと」を率直に話す文化が形成されやすい

スクラム開発の優れた点

■早めに軌道修正が出来る

ウォーターフォール開発では、プロジェクトの最終盤にならなければ、顧客は動いているプロダクトを見えなかった。
すると納品した後に「思っていたものと違う！」とトラブルになるケースもあった。

スクラム開発では、作っている機能が正しいか定期的に確認の場を設けるため、顧客の要望と違うものを作っていたら早めに気付ける

開発メンバーの役割

スクラム開発で必要となる各メンバーの役割は３つに分類される

- **プロダクトオーナー**
  作成するプロダクトの責任者
- **スクラムマスター**
  プロジェクトを進める“調整役”
- **開発メンバー**
  実際に開発を行うメンバー

開発メンバーの役割

プロダクトオーナー

- 作成するプロダクトの責任者である
- 機能の必要可否を判断する
- 機能の優先順位をつける
- プロダクトのビジョンを示す
- 他のメンバーに共有・説明する
- プロダクトオーナー自身は開発をしない
- プロジェクトのスケジュールや予算の管理をする

開発メンバーの役割

スクラムマスター

- プロジェクトを進める “調整役”である
- スクラムの進行ルールや進め方を全メンバーに説明する
- プロダクトオーナーや外部メンバーからの無理な要望をブロックする
- タスクを調整して特定のメンバーに負荷がかかりすぎないようする
- 専任の場合もあれば開発メンバーとの兼任の場合もある

開発メンバーの役割

開発メンバー

- 実際に開発を行うメンバーである
- 設計、ドキュメント作成、コーディング、テスト、運用まで一通りのスキルを持っていることが求められるのがスクラムの特徴
- 「特定分野しかやらない・出来ない」は基本的に避ける
- 苦手分野でも作業を融通し合って全メンバーが全ての仕事が出来ることを目指す
- メンバーはみな平等で、役職や年齢による上下関係はない

スクラム開発でのイベント

バックログを作る

- バックログとは、プロダクトへの要望をまとめたリストのこと
- 作成はプロダクトオーナーでも開発メンバーでも可(打ち合わせ)
- 各タスクは優先順位に基づいて定期的に柔軟に変更してよい

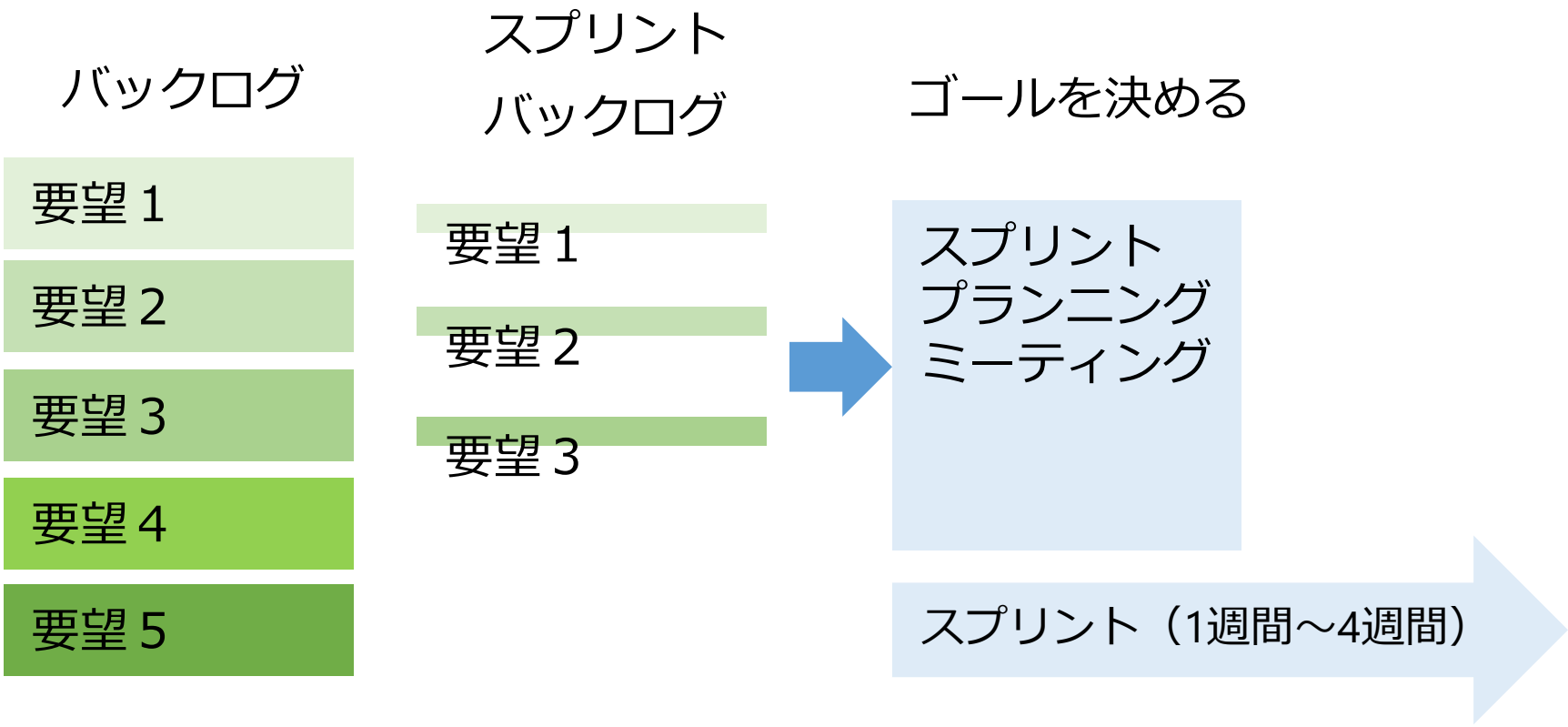
スプリントプランニングミーティング

- スプリント前に行う
- バックログの内容を元に計画を立てる
- スプリント内で「何を」「どれくらい作るのか」を決める
- バックログごとに工数見積もりをして各メンバーにタスクを割り振る

アジャイル開発（スクラム開発）

バックログ：プロダクトへの要望をまとめたリスト

スプリントバックログ：1～4週間で作成する要望リスト



アジャイル開発（スクラム開発）

スクラム開発でのイベント１

デイリースクラム

- チームの状況を共有するミーティング
- 毎朝行う（「朝会」とも呼ばれる）
- 5～15分程度の時間で振り返り、計画を立てる
- 昨日したこと、今日やること、起こっている問題などを報告

スプリントレビュー

- スプリントの最終日に行う
- プロダクトオーナーが成果物を確認する場
- 必ず動くアプリケーションを使って確認する（重要）
  ※画面キャプチャや文書での代用はしない

アジャイル開発（スクラム開発）

スクラム開発でのイベント２

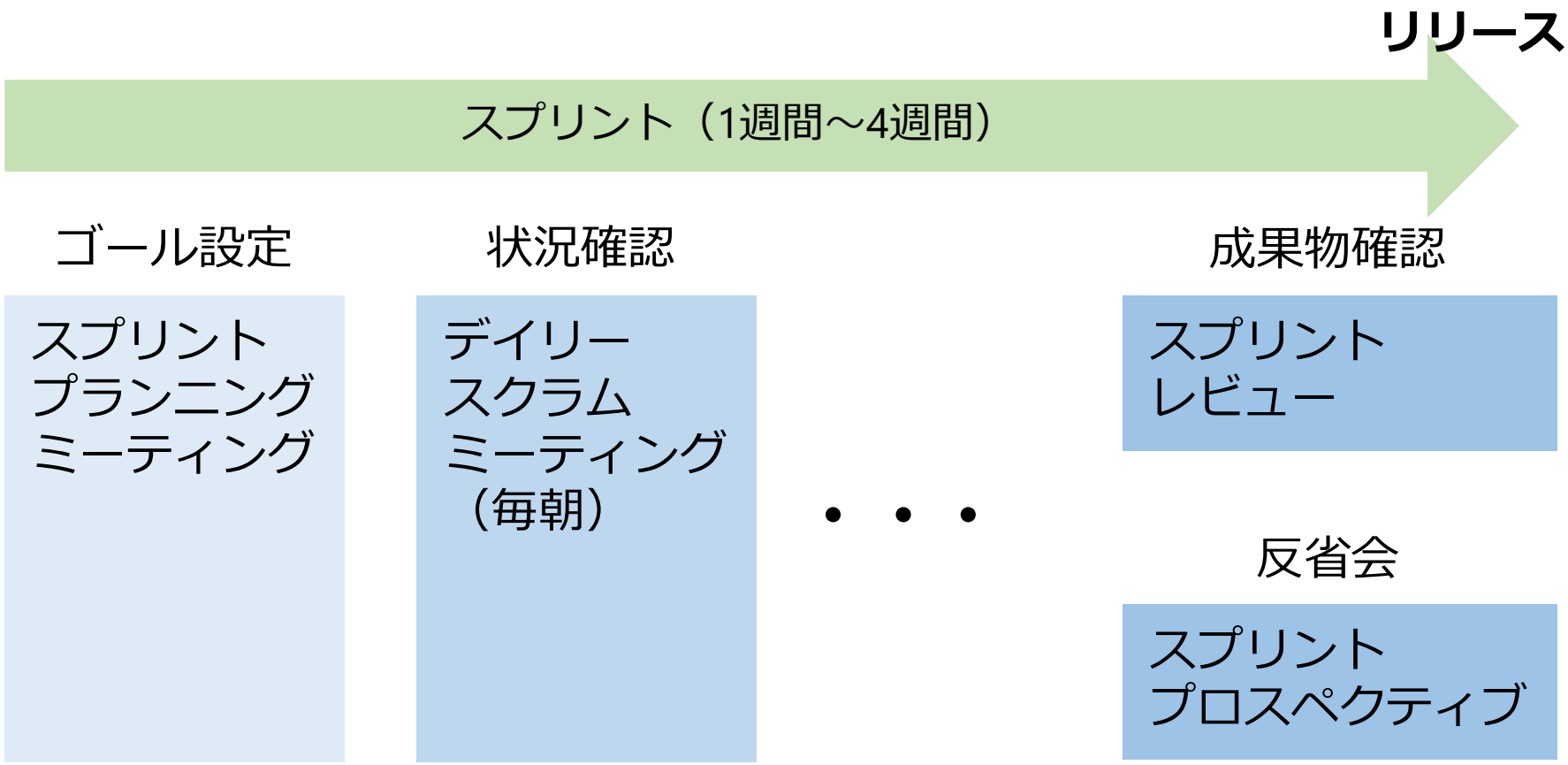
スプリントレトロスペクティブ

- これもスプリントの最終日に行う
- 目的はスプリントを振り返ること
- 各メンバーで議論し、次回のスプリントに活かす(以下を話す)
  - スプリントで良かった点
  - 改善すべき点
  - その要因と改善策

※レトロスペクティブ（retrospective）...振り返り

スプリントの流れ

スクラム開発では、開発の1つの期間単位のことを「スプリント」と呼ぶ



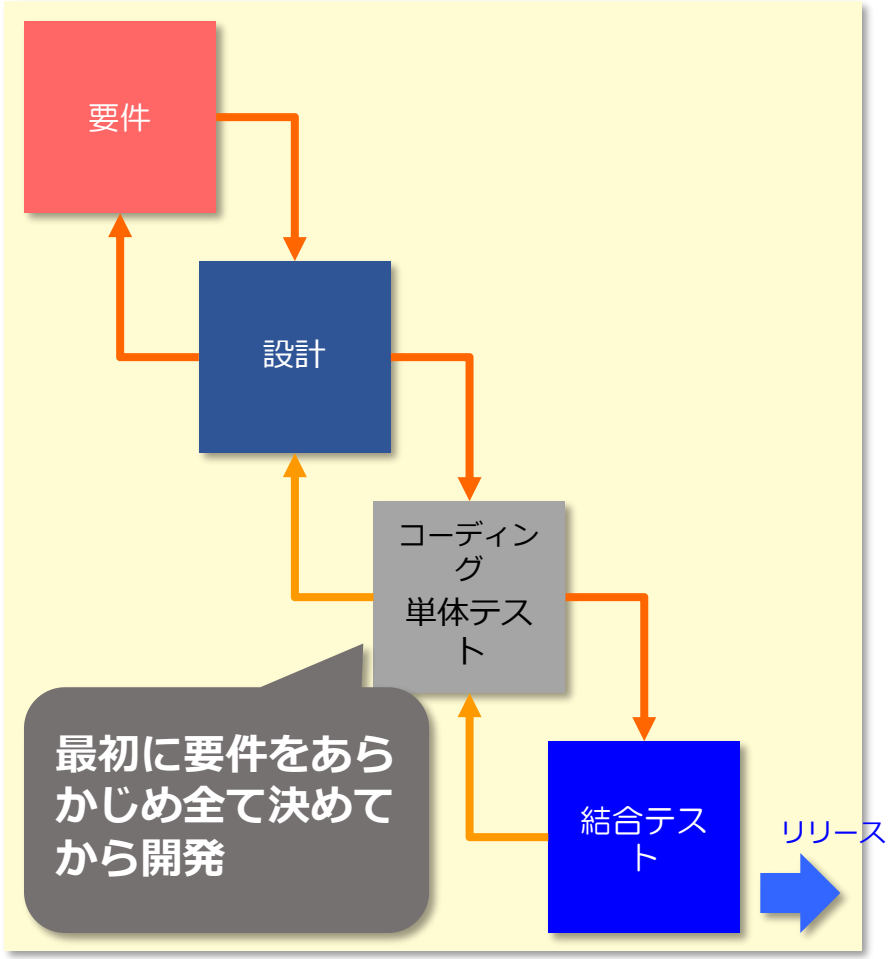
システム開発手法

- 開発手法（ウォーターフォール）
- 機能要件と非機能要件
- 開発手法（アジャイル）
- **開発手法ごとのメリットとデメリット**

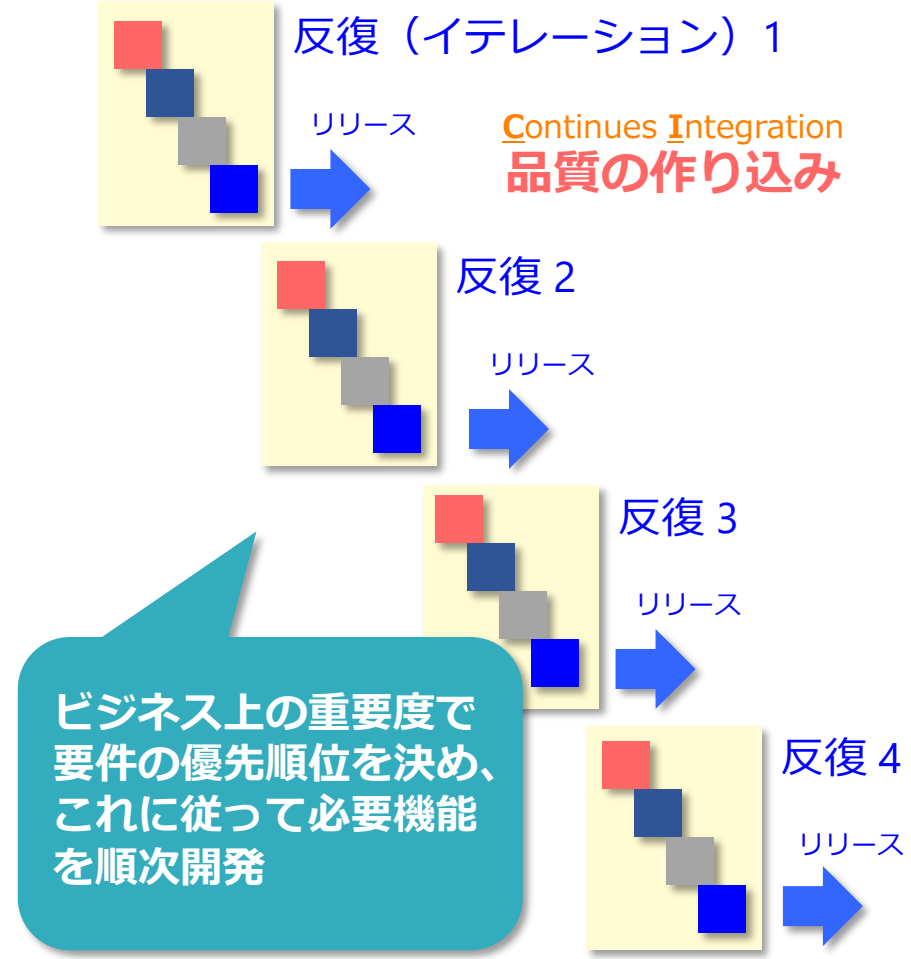
ウォーターフォールとアジャイルの比較

INTERNET ACADEMY
Institute of Web Design & Software Services

ウォーターフォール開発



アジャイル開発

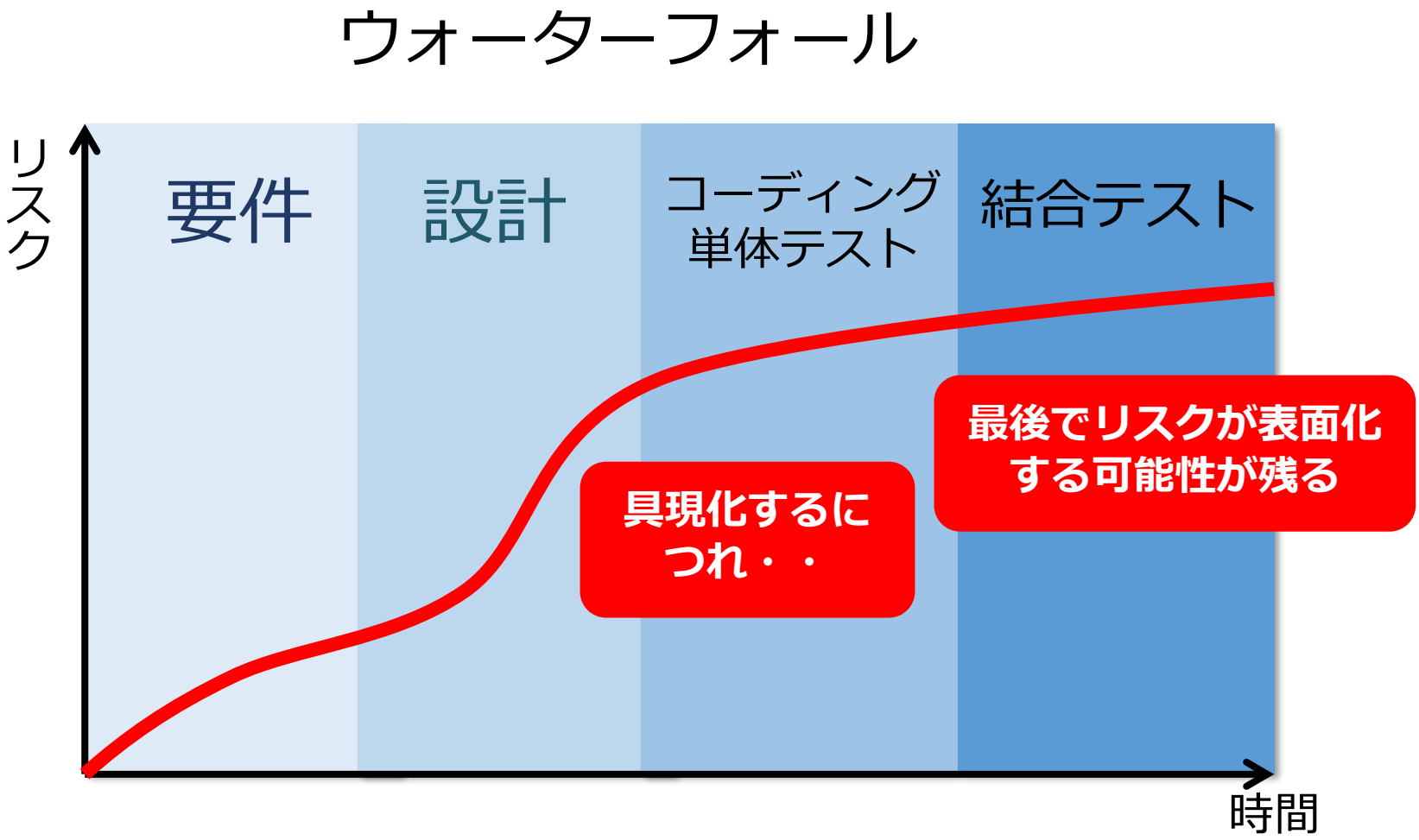


ウォーターフォールとアジャイルの比較

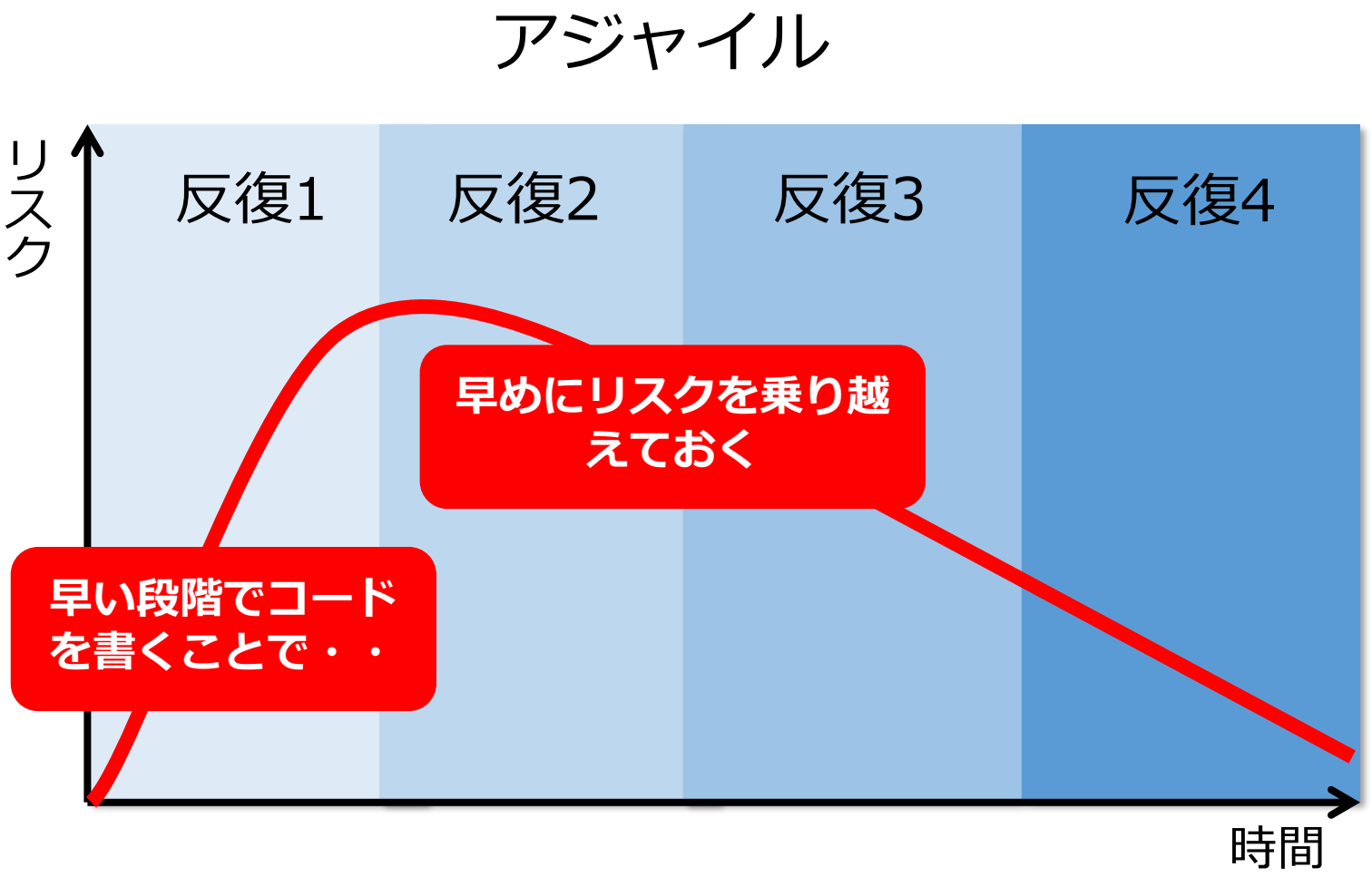
INTERNET ACADEMY
Institute of Web Design & Software Services

	ウォーターフォール	アジャイル
前提条件	要求仕様	工数と納期
ゴール	工数と納期	実現仕様
	コストと納期を守ること	機能と品質を実現すること

リスク対応のタイミング



リスク対応のタイミング

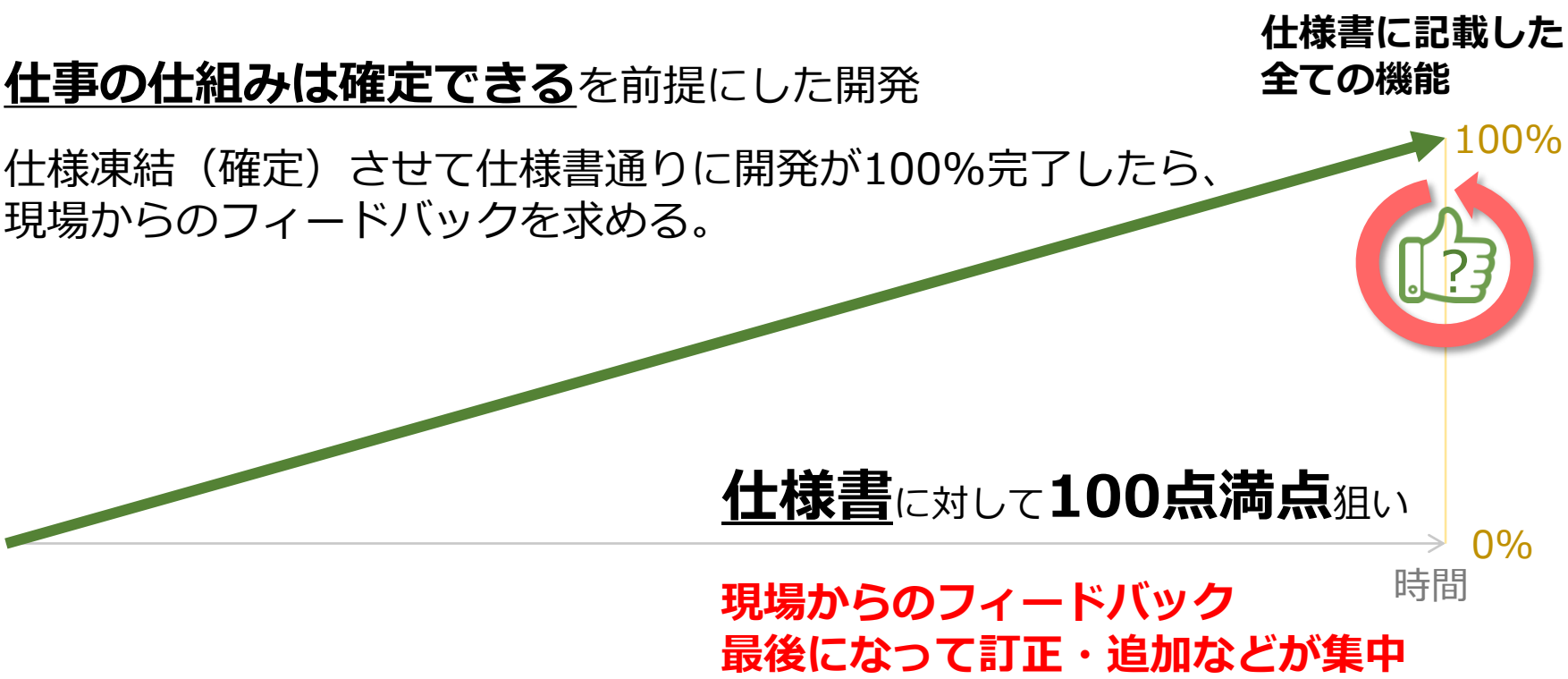


INTERNET ACADEMY
Institute of Web Design & Software Services

ウォーターフォール開発の考え方

仕事の仕組みは確定できるを前提にした開発

仕様凍結（確定）させて仕様書通りに開発が100%完了したら、
現場からのフィードバックを求める。



INTERNET ACADEMY
Institute of Web Design & Software Services

アジャイル開発の考え方

仕事の仕組みは変化するを前提にした開発

途中の成果からフィードバックを得て、
仕様や優先順位の変更を許容する。

ビジネスの成果に対して**合格点**狙い

DevOpsとは

「DevOps（デブオプス）」とは、 開発手法やツールを使って、開発者(Development)と、運用者(Operations)が密接に連携することで、 より柔軟かつスピーディーにシステムを開発すること。

【メリット】

・テスト、ビルド、デプロイといった今まで人間が行なっていた作業を自動化することで、ヒューマンエラーを防げる

・Excelで管理していた煩わしい事務作業を減らすことができる

開発と運用の協調・連携を実現するDevOps

情報システムに求められること

- システムによってビジネスの成功に貢献すること
- ビジネスの成功のための貢献を確実、迅速にユーザーに届けること
- ユーザーの求める貢献の変化に迅速・柔軟に対応すること