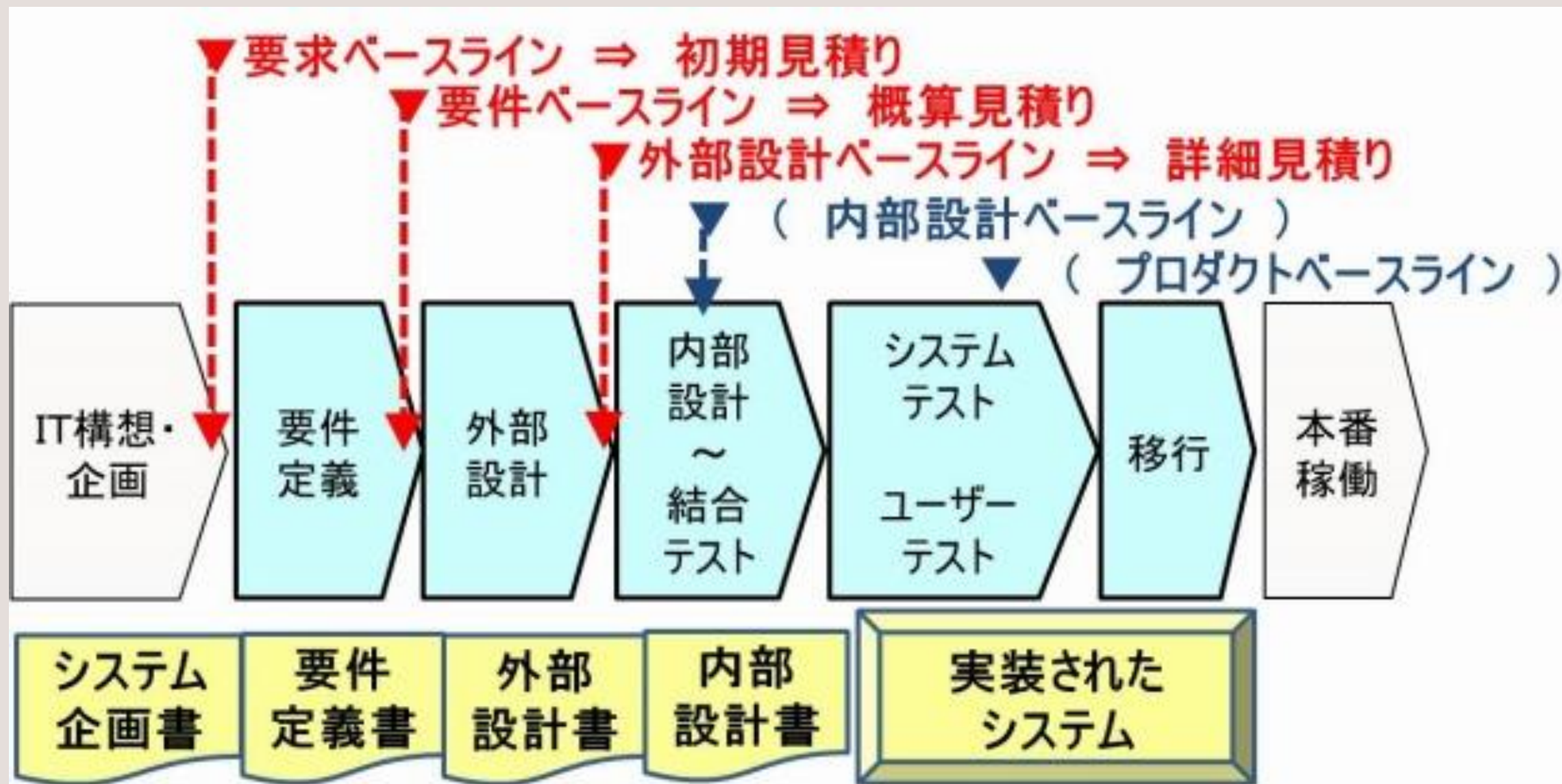
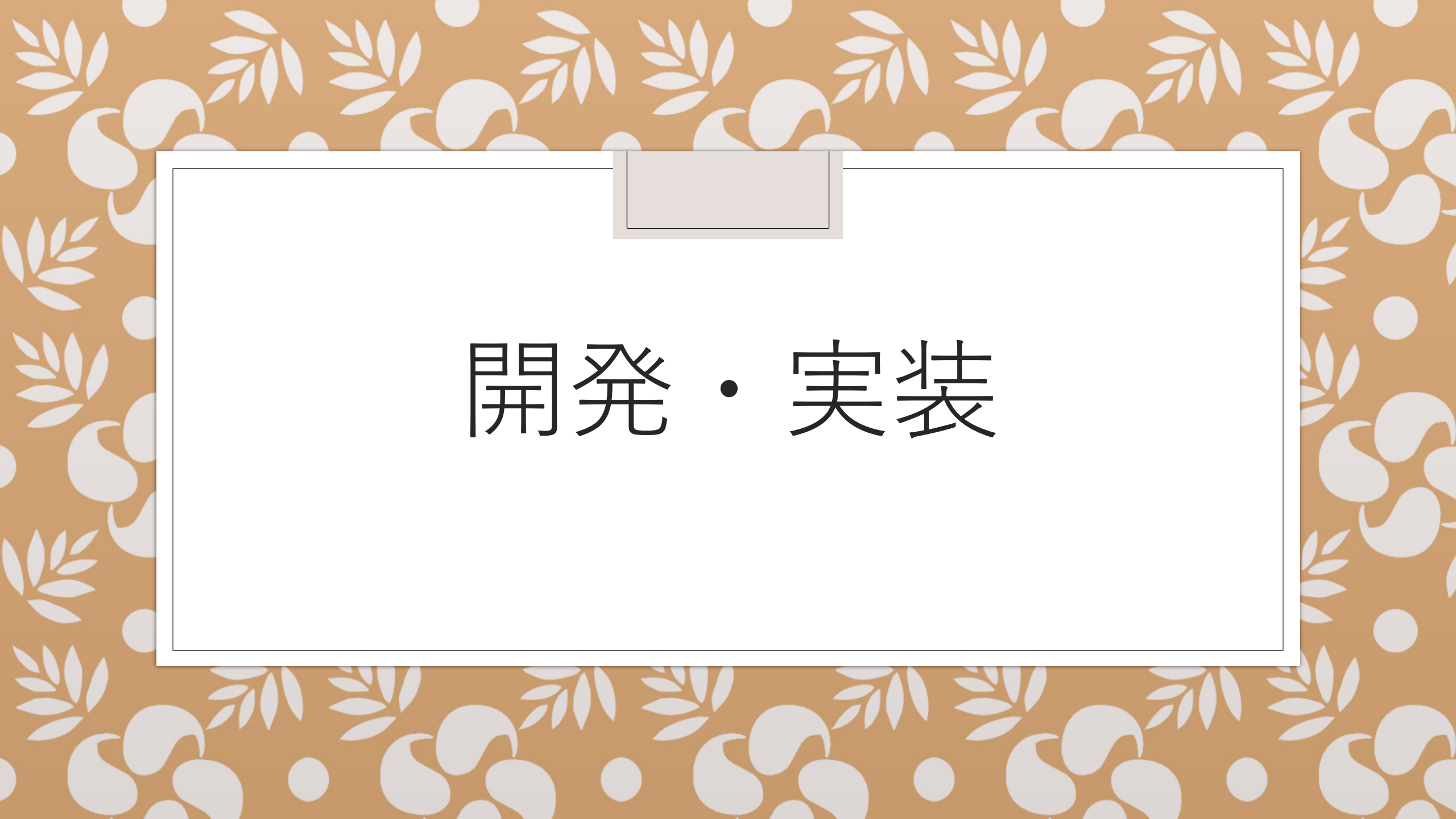




システム設計実践演習 第9回

仕事の流れについて





開発・実装

開発・実装の全体像

内部設計が完了したら、その内容をもとにプログラムを作成します。これを**開発**や**実装**、または**製造**と呼びます。

ソースコードを作成するには、設計工程で作成した設計書が必要でこの工程での成果物はプログラムとなります。

※厳密には、設計書で定められたとおりに動作しているかを確認するテストも含まれますが解説は別途行います。

プログラム言語の選択

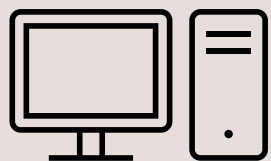
ソースコードはプログラム言語で記述しますが、どのプログラム言語を使えばいいのか、よく問題になります。一般的に使われるものでも20種類ぐらい存在します。

このとき、最初に考えることは「**何をつくるか**」です。

iOS用のアプリとWebアプリでは向いている言語が異なります。

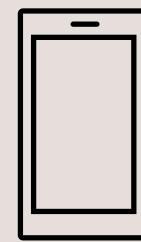
プログラミング言語は目的に合わせて選ぶ

デスクトップアプリ

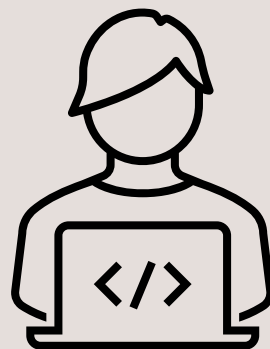


C#、VB.NET

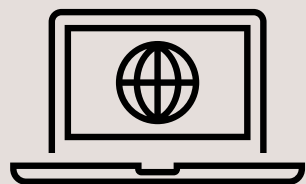
モバイルアプリ



Kotlin、Swift

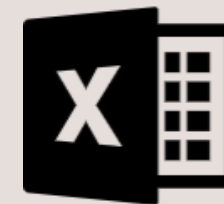


PHP、JavaScript



Webアプリ

VBA

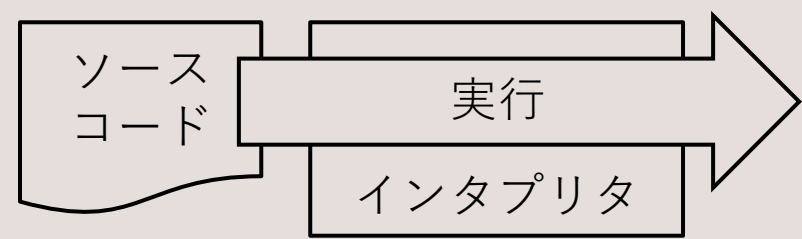


Office製品

インタプリタ言語とコンパイラ言語

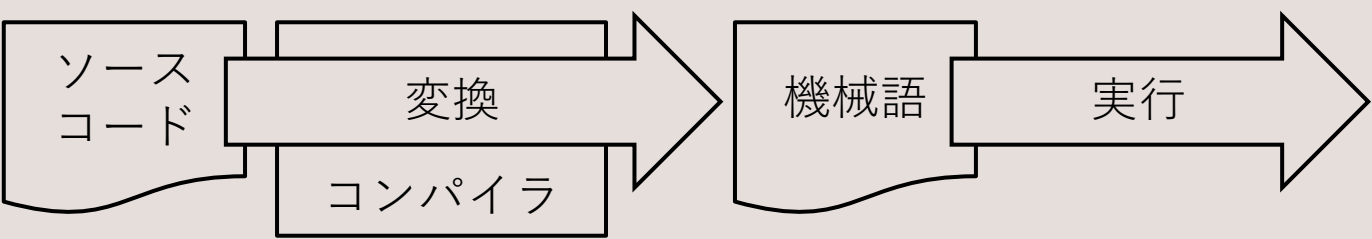
種類	説明	メリット	デメリット	言語
インタプリタ	コードを実行する際に 1 行ずつ機械語に翻訳していく言語	プログラムを即実行できる	実行速度が遅い	PHP Python など
コンパイラ	まずすべてのコードを機械語に翻訳してからまとめて実行する言語	プログラム処理速度が速い	修正のたびにコンパイルが必要	Java Go など

インタプリタ型



機械語に翻訳しながら実行

コンパイラ型



事前に機械語に翻訳 実行するときは機械語から

チームでの開発を意識する

複数人でシステムを開発する場合、チームメンバーが理解できる言語を選定する必要があります。また、**開発したシステムは数年単位で使い続けられる可能性があるため、多くの開発者が対応できる言語選択がベター**となります。

しかし、技術進歩もあり過去に経験のない**新しい言語選択が迫られた場合は、部署で勉強会を開催したり他のメンバーを巻き込んで開発できる体制**を作る必要があります。

フレームワークの選択

現代のアプリの多くは効率よく開発をするために何らかのフレームワークを使っています。

Windowsデスクトップアプリであれば「.NET Framework」
PHPであれば「Laravel」などが有名です。

フレームワークは開発の土台として提供されるもので、用意された**フレームワーク沿って開発をしていれば、ある程度の機能を実装できます。**

フレームワーク選択の注意点

フレームワークは便利な仕組みですが使用する場合、そのフレームワークに沿ってプログラムを作成する必要があります。

最初の段階で、**採用するフレームワークが出来ること出来ないことを把握しておかないと、開発の後半で大幅な修正が発生します。**

また、後からフレームワークを変えたいと思っても、全面的なソースコードの書き換えが必要になるため、現実的ではありません。

ライブラリの使用

フレームワークと似た役割としてライブラリがあります。

多くのプログラムで使われる便利な機能をまとめたもので、
例を挙げると次のような機能があります。

- ファイルの読み込み、保存
- 画像処理（拡大、縮小、ファイル形式の変換など）
- HTTP通信のリクエスト・レスポンス処理

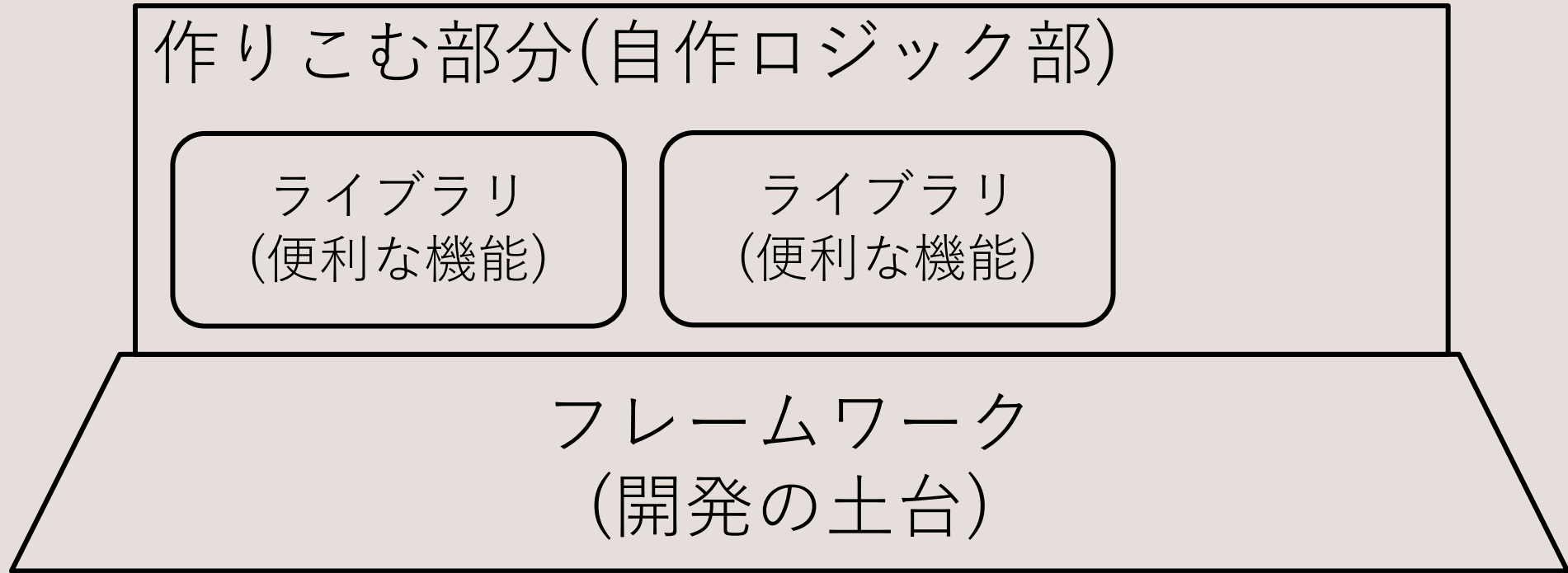
フレームワークとライブラリの違い

フレームワークはソフトウェアの土台であり、多くのソフトウェアで必要とされる機能があらかじめ用意されています。

そのため、**フレームワークを使うことで必要な機能がある程度実装された状態から開発をスタートすることが出来ます。**

一方、ライブラリは開発したい機能に合わせて選択できるパーツで、**自分が欲しい機能に必要なライブラリを組み込むことで効率よく開発を進めることが出来ます。**

フレームワークとライブラリのイメージ



開発者は、フレームワークに合わせてカスタマイズしたい部分を記述し、ライブラリを呼び出す処理を実装することで新たなソフトウェアを短期間で開発することが出来る。

開発環境と本番環境

。開発環境

システム開発を行うときに、開発者が使用するPC上で動くソフトウェア。プログラミングを行うソフト以外にも**バージョン管理やテストを実行するためのソフトウェアも含まれます。**

。本番環境

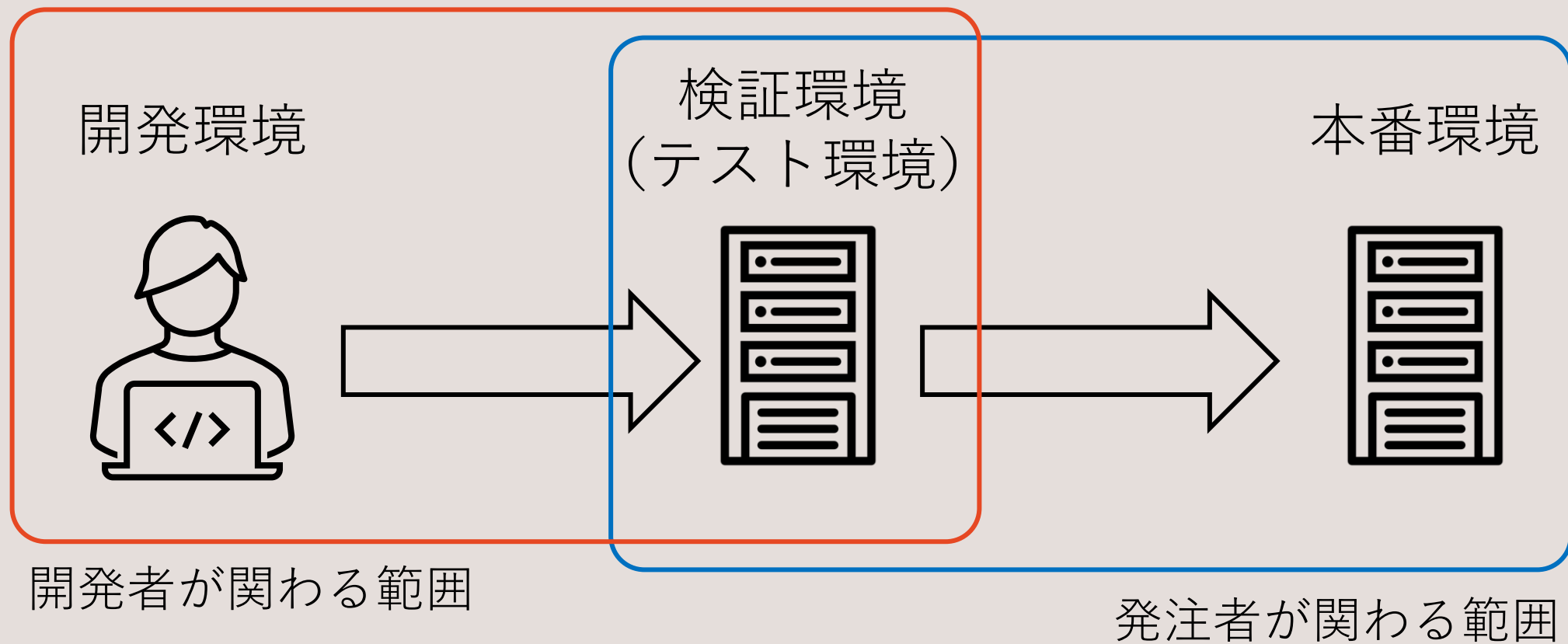
開発したソフトウェアを利用者が使う環境。Webアプリであれば、それらを公開するためのWebサーバーなど**一般の利用者が実際にアクセスするときに使う環境**を指します。

検証環境(テスト環境)

既存のWebアプリを改修する場合など、すでに利用者がいる状況で、**開発環境から本番環境に直接プログラムを配置すると、トラブルになる可能性があります**。例えば、変更した内容を発注者が事前に確認できなかったり、変更した内容に不具合があって利用者に影響が出たりといった事象が考えられます。

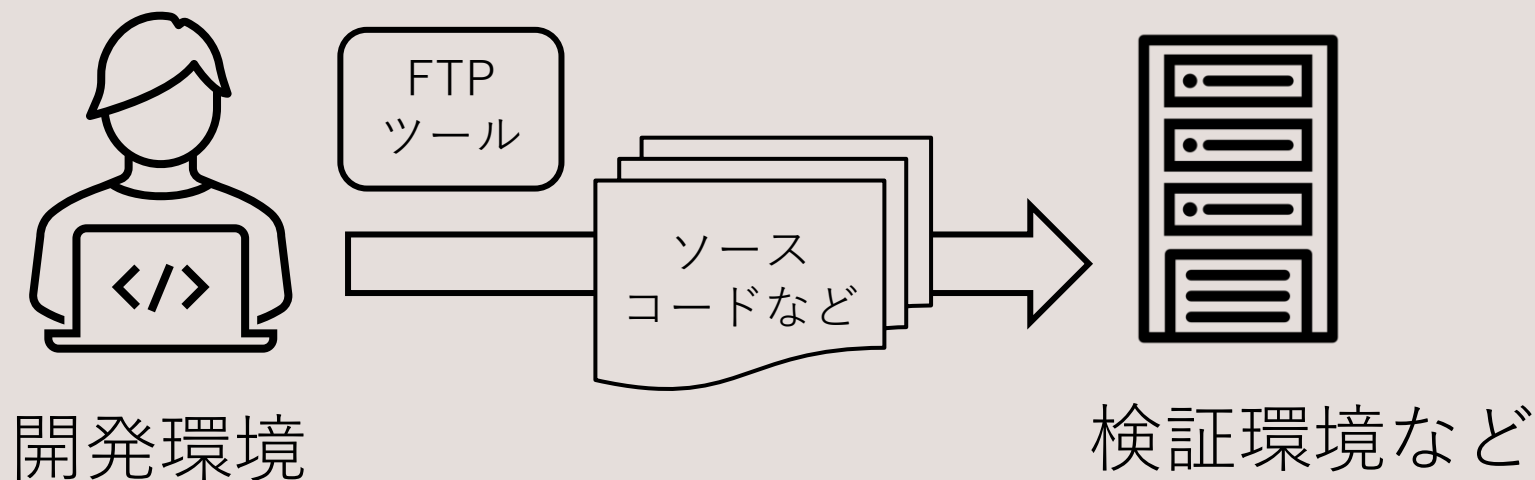
これらを防ぐため、開発環境で作られたシステムを本番環境に近いハードウェアを用意し、発注者が確認するための環境を用意します。これを**検証環境**と言います。

各環境と主な使用者とプログラムの移行



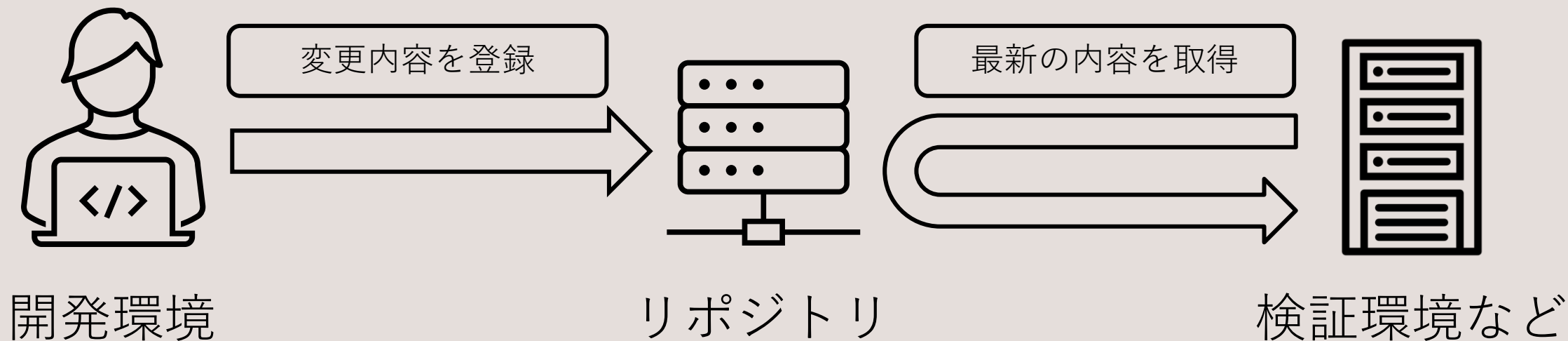
プログラムや設定ファイルの移行

開発環境から検証環境、本番環境にプログラムや設定ファイルを移行することを**デプロイ** (配置) と言います。デプロイの方法としてわかりやすいのは、開発環境で作成したファイルをFTPなどを使い、検証・本番環境にファイルを送受信する方法です。



プログラムや設定ファイルの移行

FTPでのデプロイの場合、更新対象のファイル管理煩雑さや冗長な上書きコピーを引き起こしてしまう可能性が高いです。そこで**バージョン管理ソフトを活用したデプロイ**を行う方法が良く使われます。また、もし検証や本番で問題が発生してもバージョン管理で以前のバージョンに戻すことができます。



プログラムや設定ファイルの移行

さらにデプロイ作業を自動化するためにバージョン管理ソフトに変更が登録されるとその変更を検出して自動的にテストしてデプロイを行うCI/CD(Continuous Integration/Continuous Delivery & Deployment)という方法があります。

