

ソフトウェア設計手法に ついて調べてみた (Bottom編)

hide00310 (ヒデ)

自己紹介

名前 : hide00310 (ヒデ)

趣味 : VRChatワールド作成

環境 : Unity, Python, C++など



X : @hide00310

はじめに

本LTの内容

- ソフトウェア設計手法について
自身が勉強した内容を自分なりの解釈で紹介

注意

- 私個人の独自の解釈や主張が含まれる場合がある
- 独学&勉強途中のため間違っているかも

本LTの目的1/2

ソフトウェア開発って難しくないですか？

何が難しい？

- ソフトが解決しようとする要求自体が難しい
 - →地道に取り組むしかない...
- ソフトの設計・コーディングが難しい
 - →先人たちの知恵があるはず！

本LTの目的2/2

個人開発の難しさ

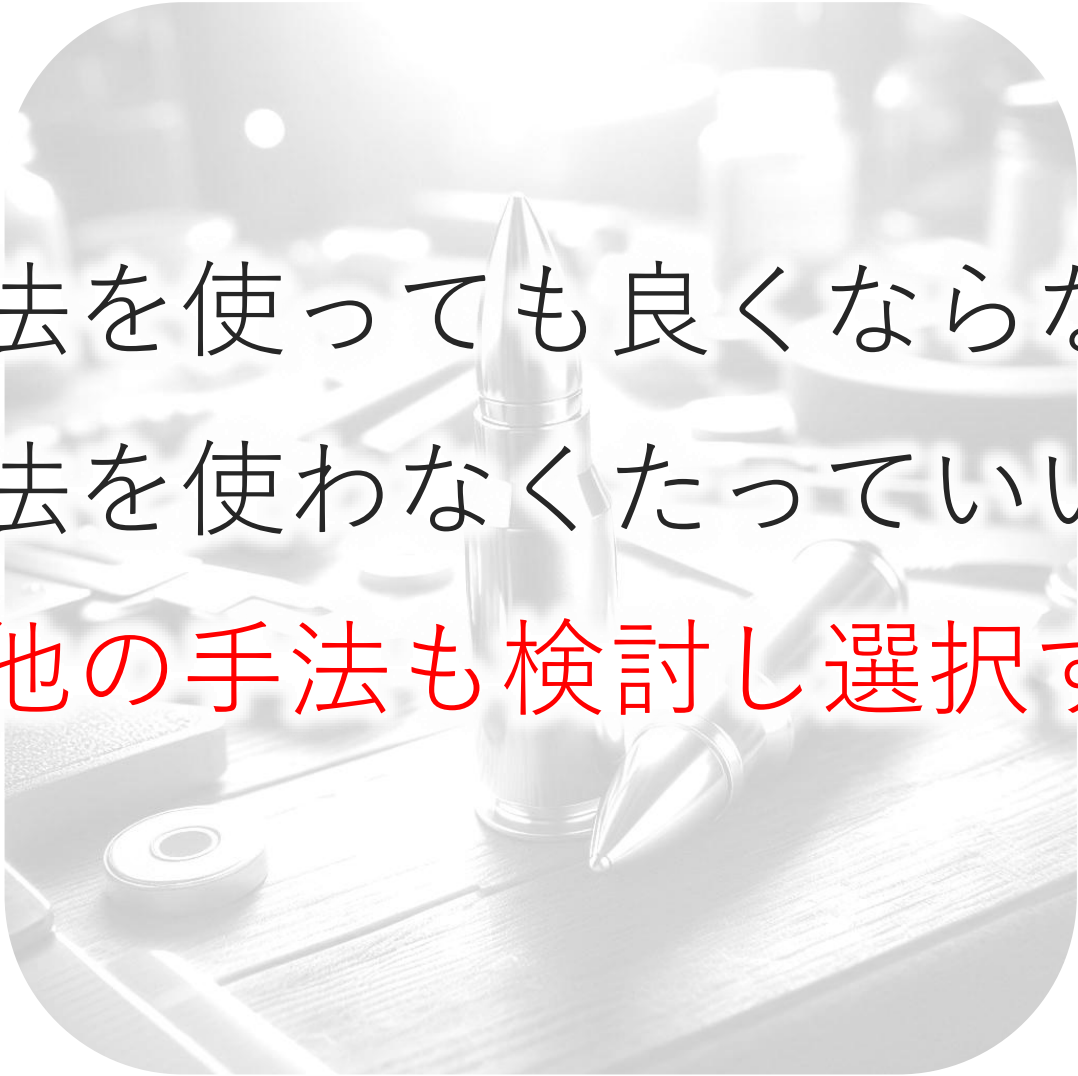
- 要求定義からコーディングまで幅広い開発が必要
- ソフトウェア設計手法について現在独学中
- 勉強した内容を紹介したい
- 皆さんの個人開発の効率UPになれば!

紹介の前に



銀の弾丸はない

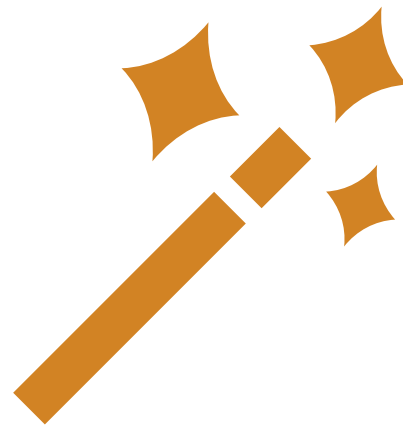
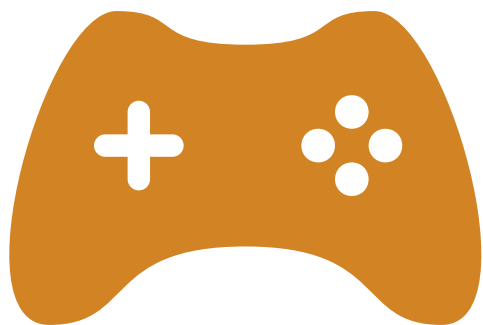
=魔法のような便利な手法は存在しない



紹介する手法を使っても良くならないこともある
紹介する手法を使わなくたっていい
大切なのは他の手法も検討し選択すること

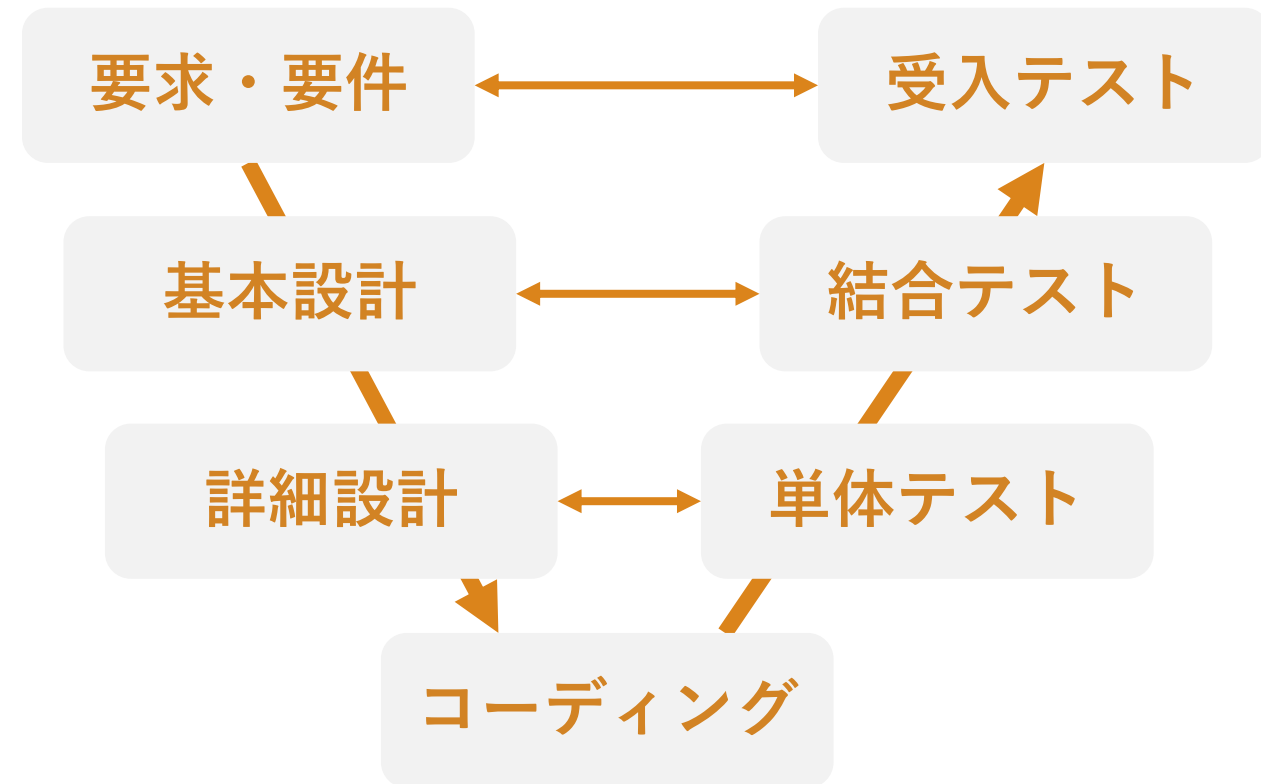
例について

すべてRPGゲームを作る例で説明



スコープについて

- V字モデルに当てはめて手法を紹介
- スコープの定義で開発手順ではない



V字モデルの例1/3

こんな
RPGゲームを
作りたい

要求・要件

受入テスト

基本設計

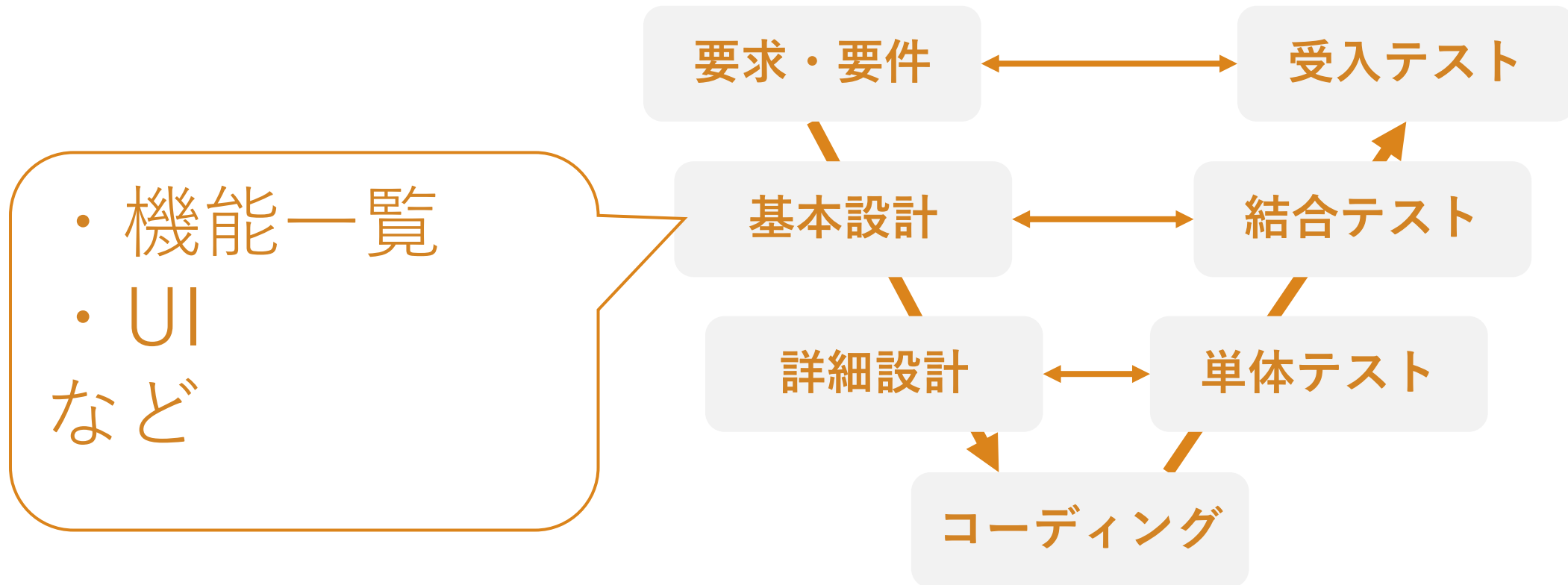
結合テスト

詳細設計

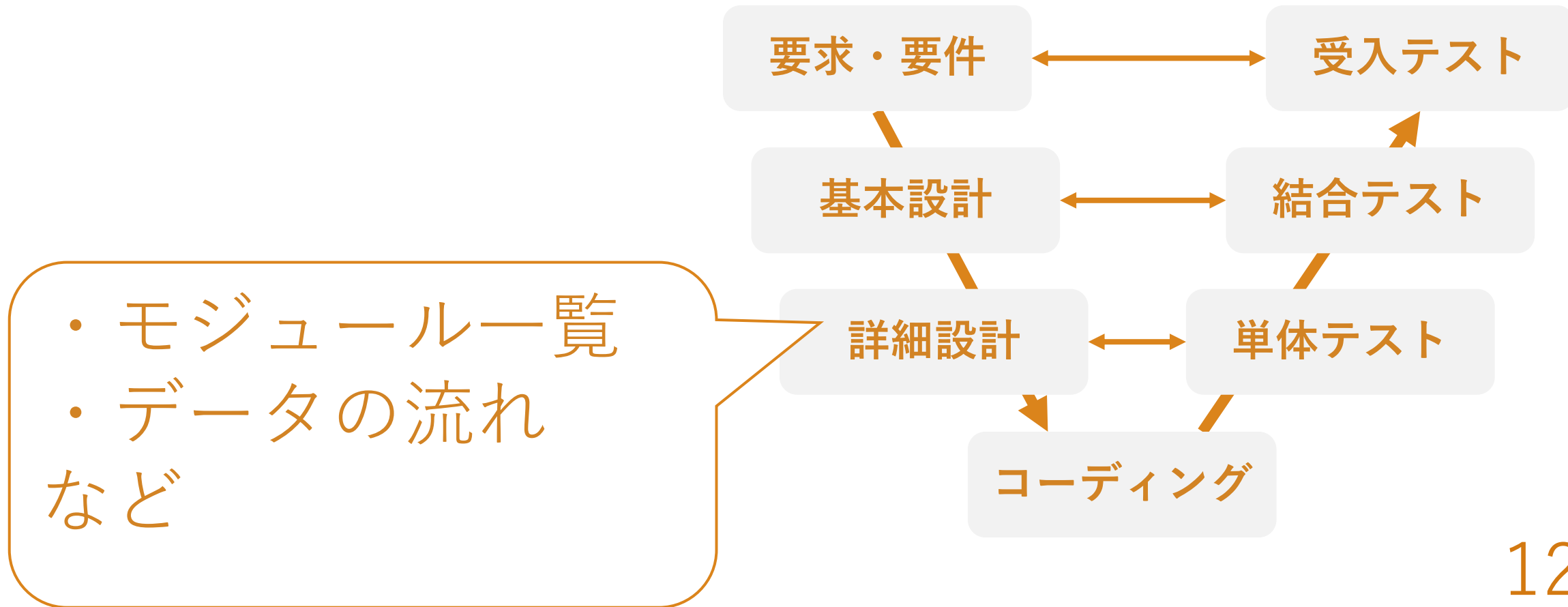
単体テスト

コーディング

V字モデルの例2/3



V字モデルの例3/3

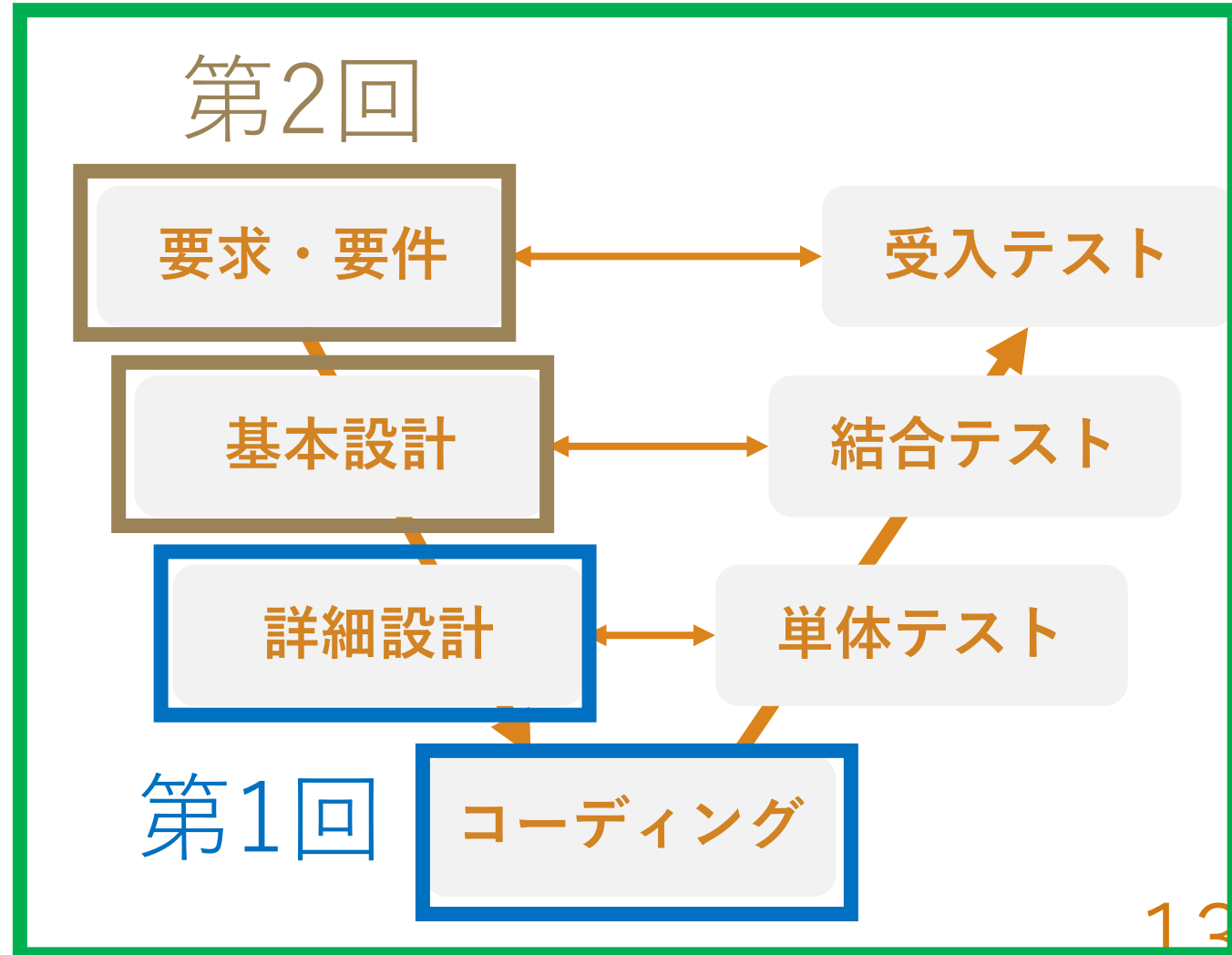


LTの流れ

コーディングから
要求設計まで
全3回にわたって発表
(予定...)

それぞれの発表は
独立しており
全部見なくても大丈夫
(なはず...)

第3回



アジェンダ

第1回←今回

- 良いコードとは
- モジュール分割

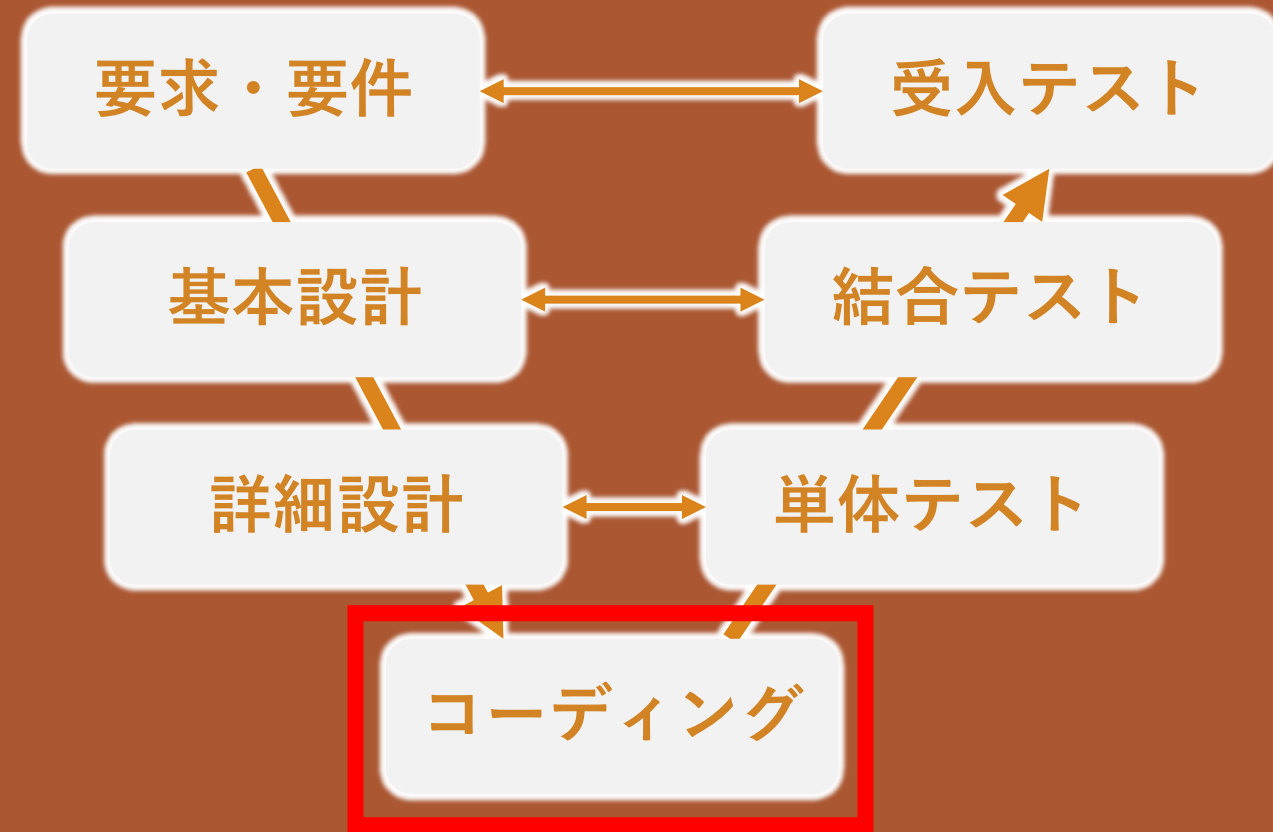
第2回

- オブジェクト指向
- SOLID原則など

第3回

- クリーンアーキテクチャ
- アジャイル開発など

コーディング



良いコードとは1/2

× 悪いコード：読みにくい

○ 良いコード：読みやすい

(ここでは定義)

次ページから良いコードにする方法を
いくつか紹介

良いコードとは2/2

こんなことはありませんか？

- 変数や関数名が分かりにくい
- 長いコードを全部読まないといけないと何をしているか分からない

→適切な命名と分割が重要

命名

より意味が分かる名前を使う

✗ bool updated 何が更新されたのか
分からない



○ bool updatedButton1

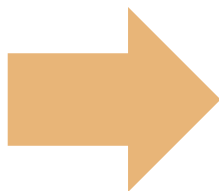
分割

複雑な処理は関数に分ける

- エディタの1ページに収まるレベルが適切?

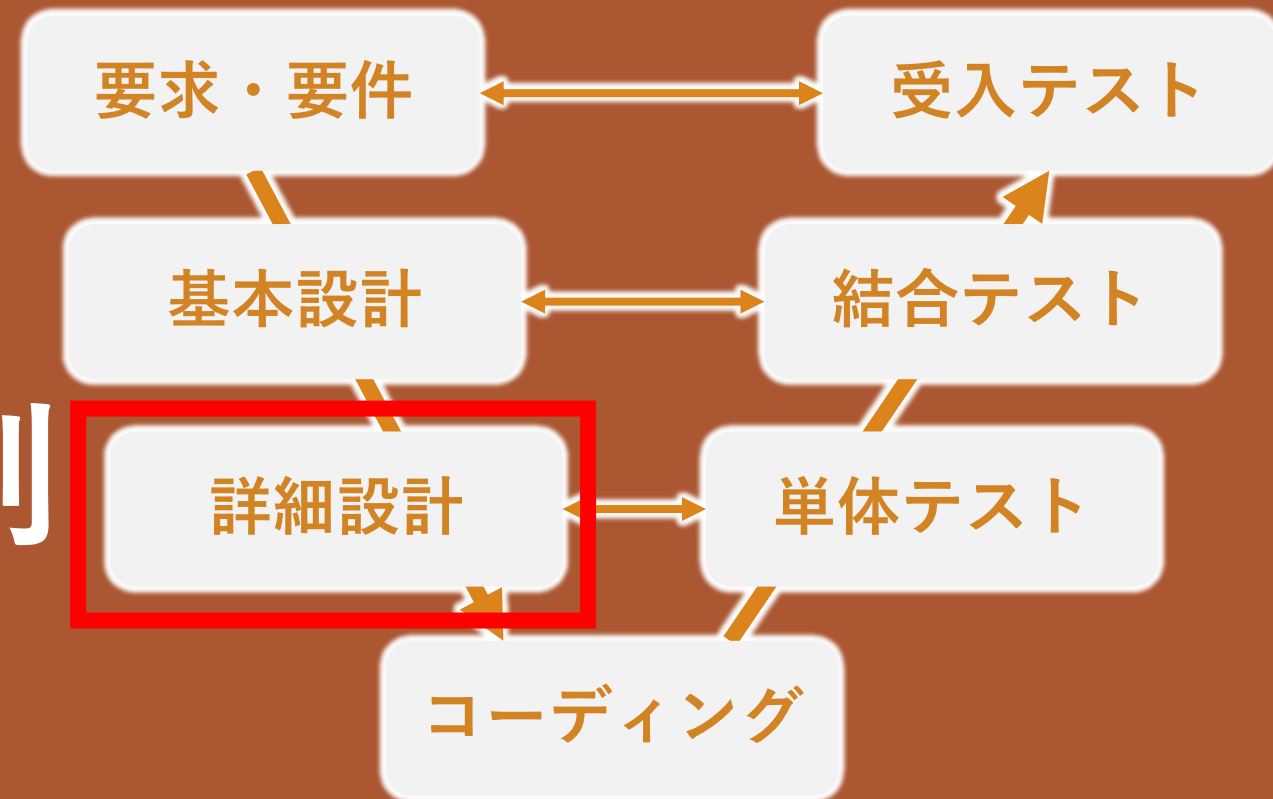


```
if ○  
    if ×  
        ...  
else △  
    if ...
```



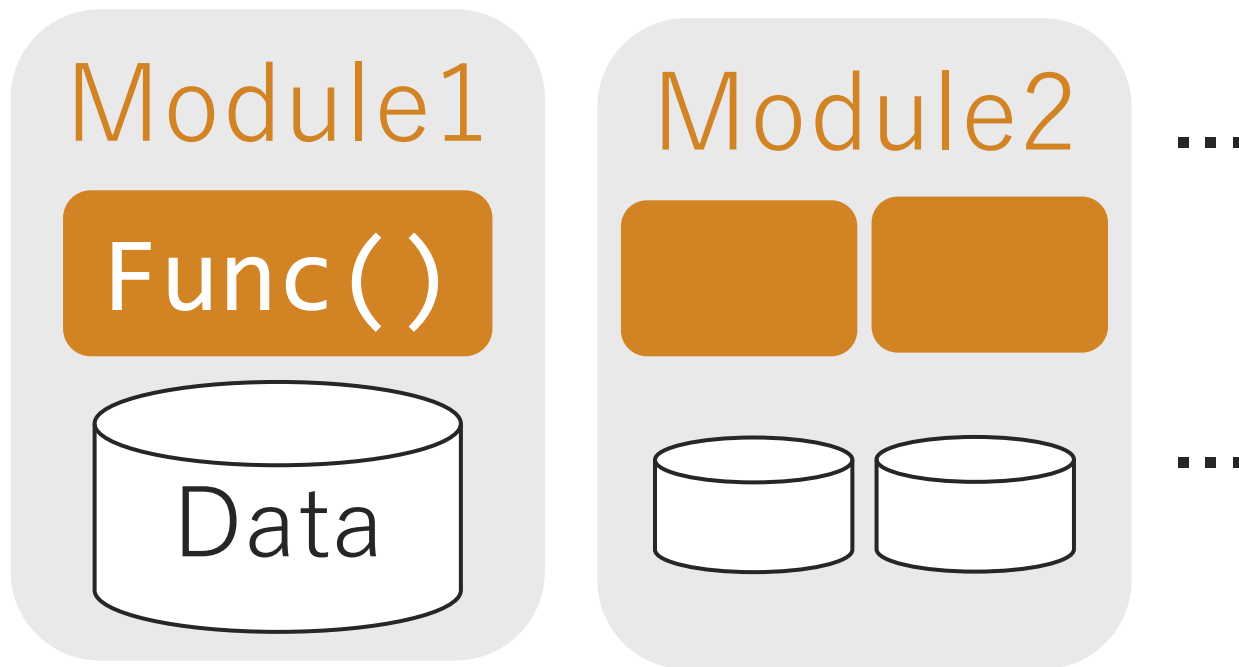
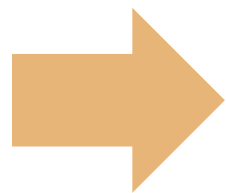
```
if ○  
    if ×  
        FuncA()  
else  
    FuncB()
```

モジュール分割



モジュール分割とは

プログラムを(複数の)関数とデータの
まとまりに分けること(ここでは定義)



良いモジュール分割とは

凝集度と結合度という指標で判断できる

× 低凝集または高結合

○ 高凝集かつ低結合

→ 次ページから凝集度・結合度について
説明

× 凝集度が低いと...

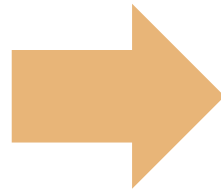
こんなことはありませんか？

- プログラムが巨大すぎて理解できない
- 1つの修正が他のコードに影響しないか不安
- 同じ処理が複数個所に書かれていて無駄

凝集度の定義

1モジュールが担当する役割が
どれだけ少ないか

✕ 低い



○ 高い



○凝集度が高い利点

- モジュールの役割が明確で理解しやすい
- 修正の影響が限定的
- 同じ処理を再利用できる

× 凝集度が低い例



○凝集度が高い例



× 結合度が悪いと...

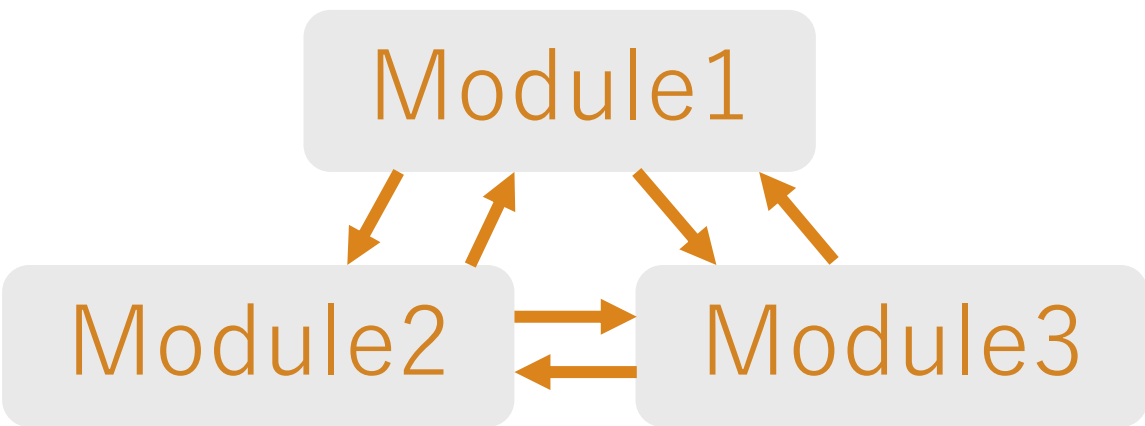
こんなことはありませんか？

- 1つの修正で複数モジュールを変更しないといけなくなった

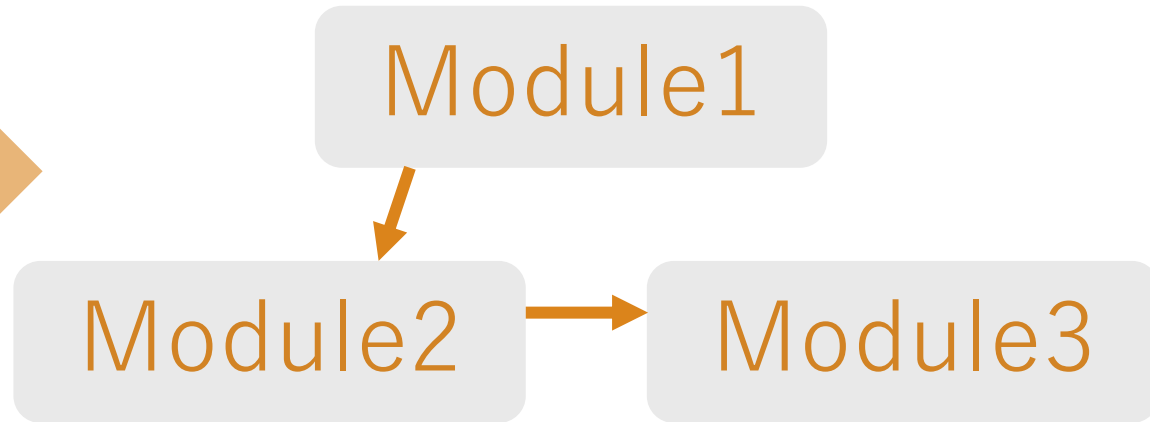
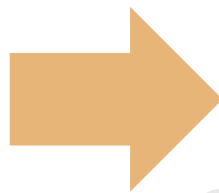
結合度の定義

モジュール間がどれだけ関係していかないか

✕ 高い



○ 低い



○結合度が低い利点

機能修正の影響範囲が少ない

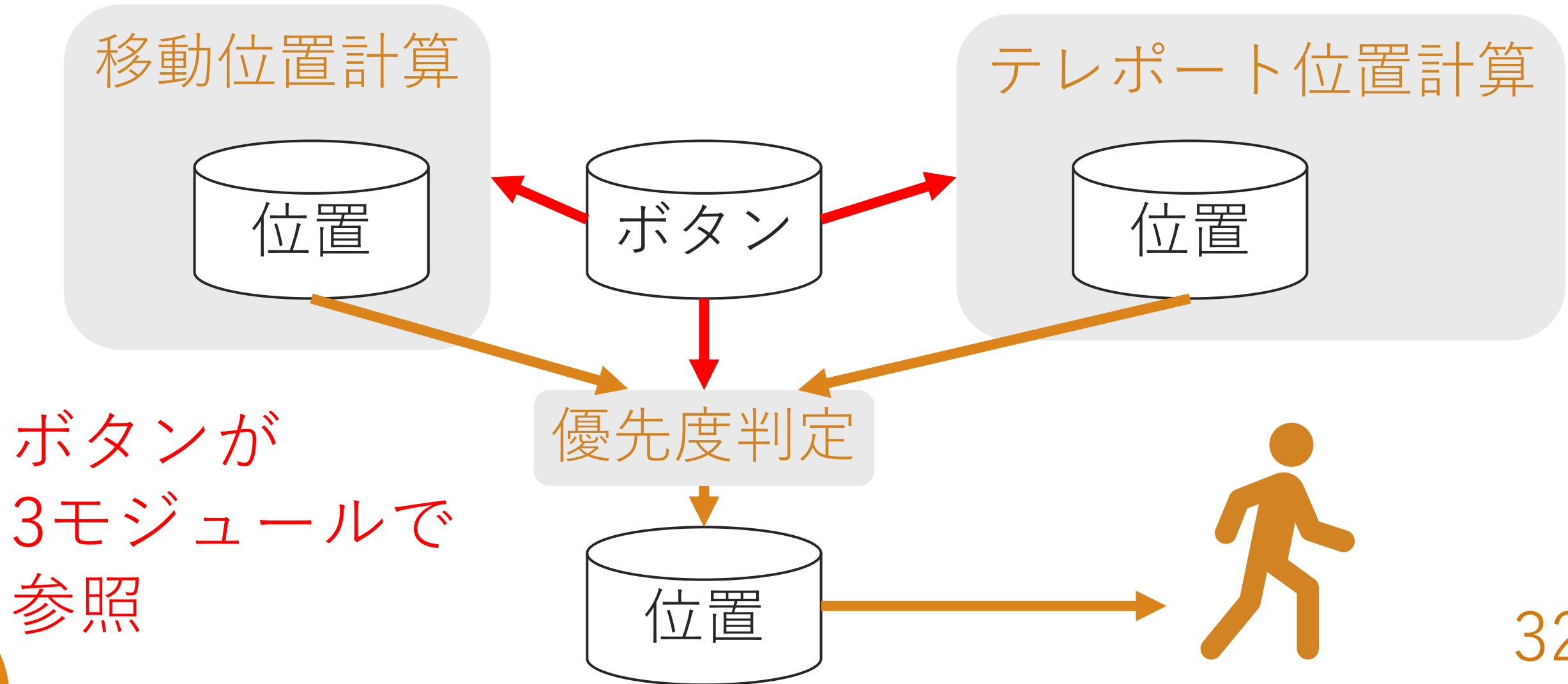
× 結合度が高い例1/3

要求

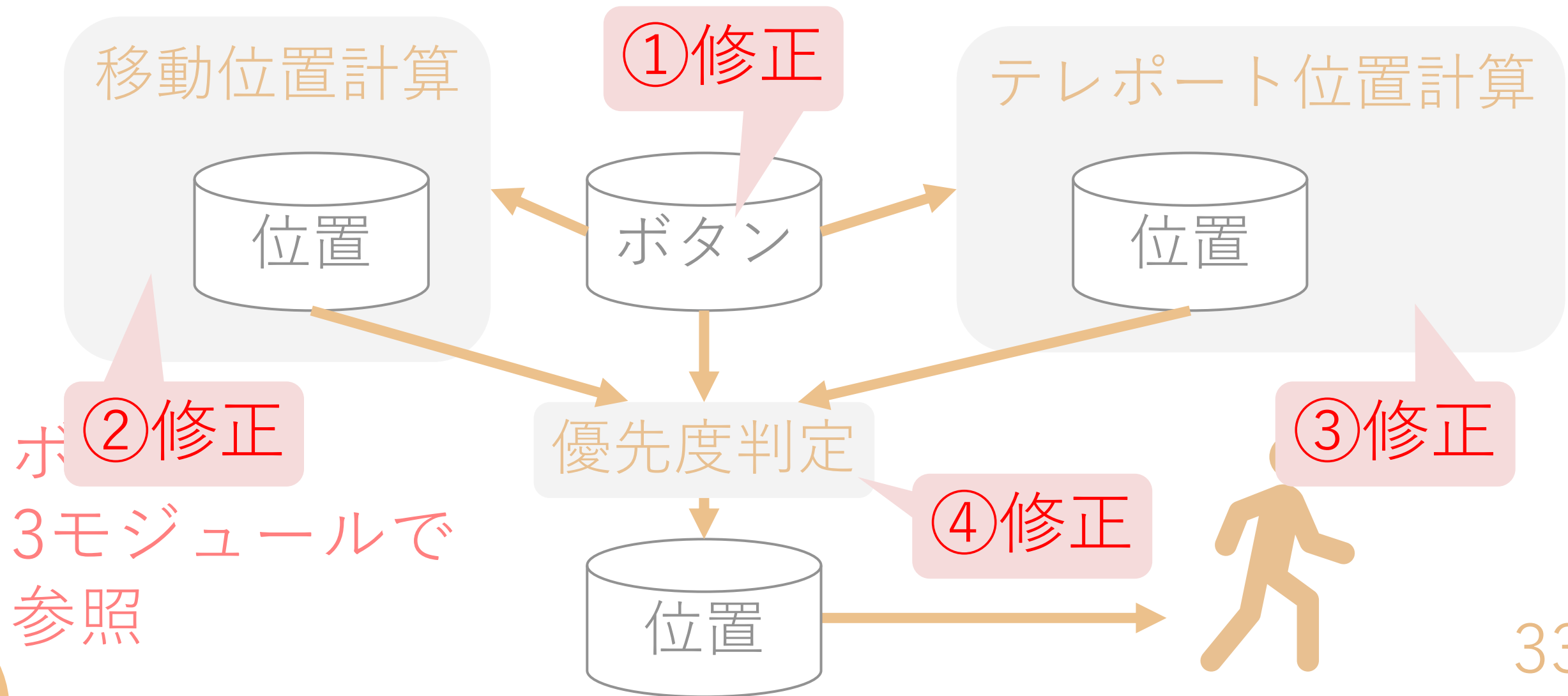
- 「移動」ボタンまたは「テレポート」ボタンでキャラクター位置更新
- 同時にボタンを押した場合は「テレポート」を優先



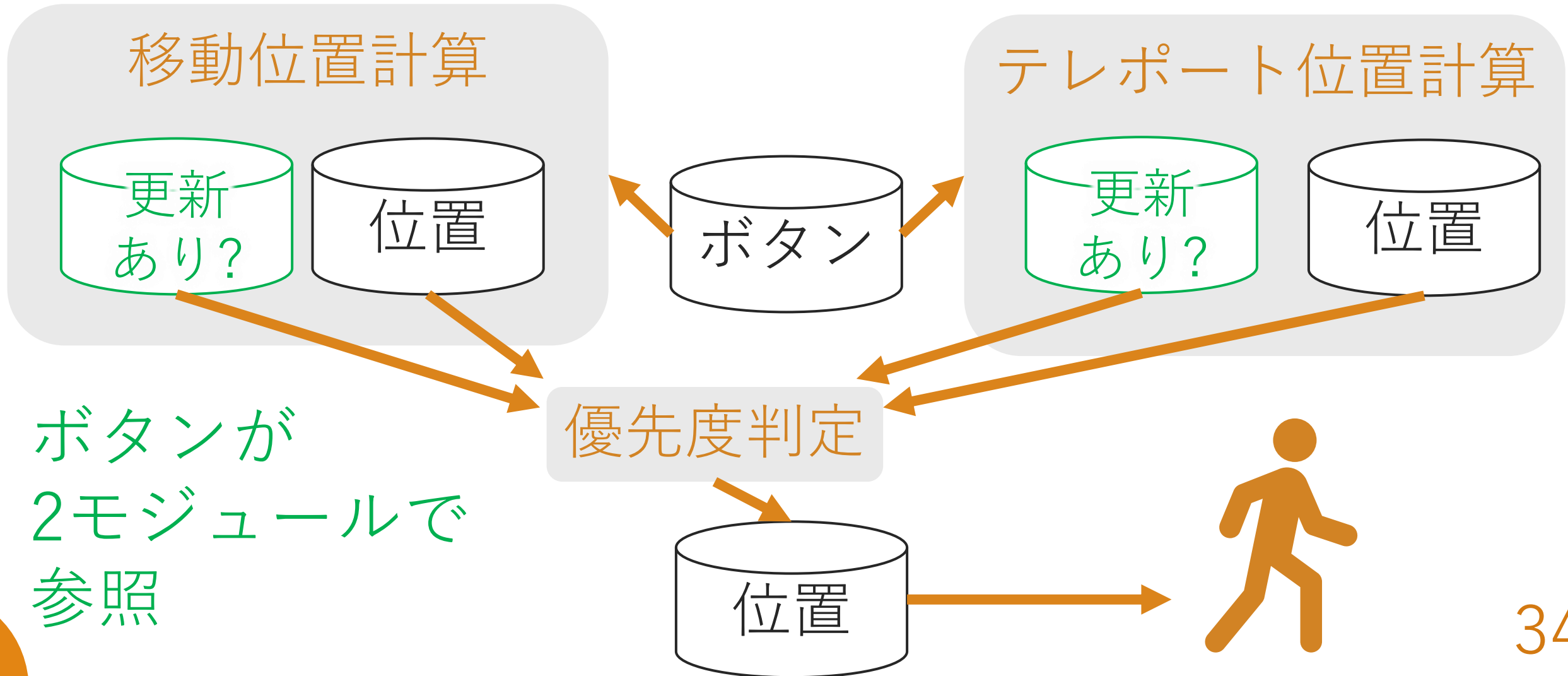
× 結合度が高い例2/3



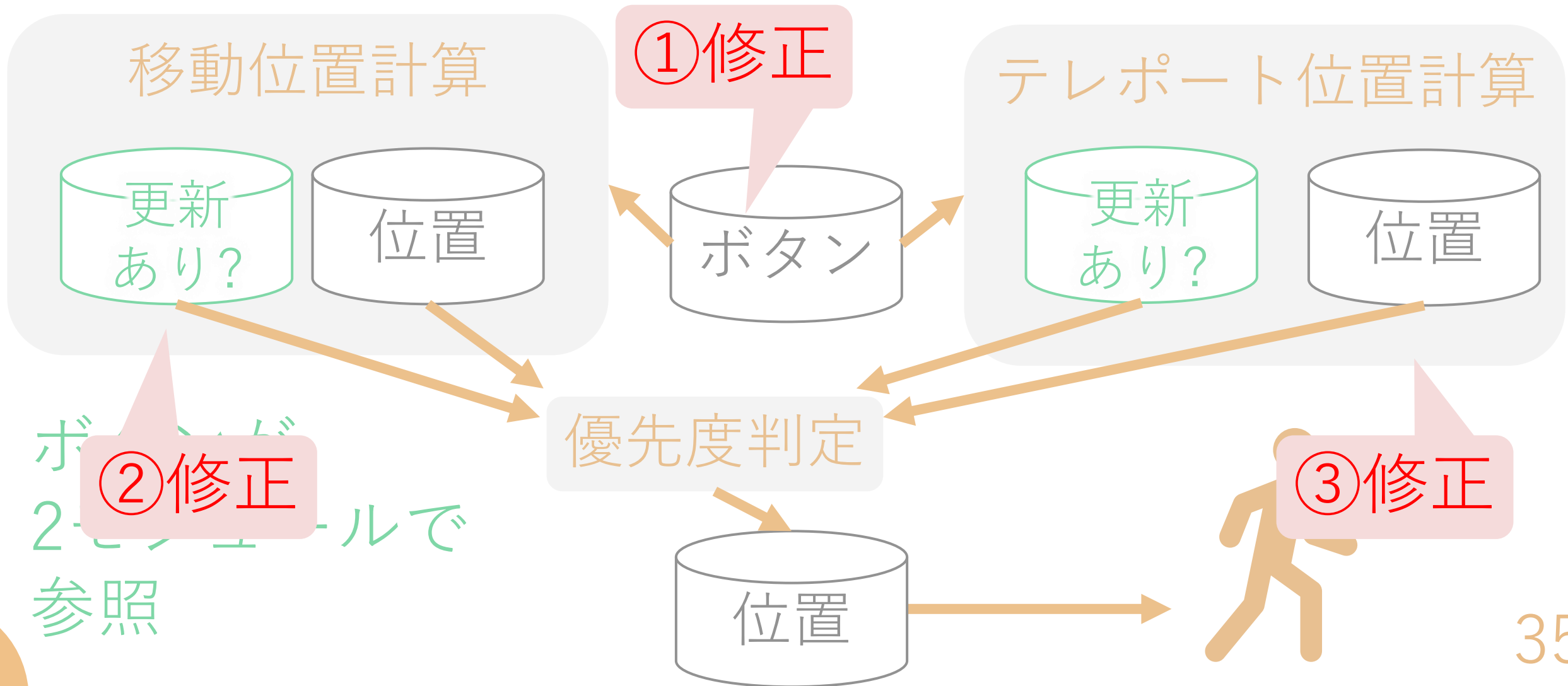
× 結合度が高い例3/3



○結合度が低い例1/2



○結合度が低い例2/2



参考文献

角征典. (2016). リーダブルコード: より良いコードを書くためのシンプルで実践的なテクニック.

- 良いコードの書き方が分かる

田中ひさてる. (2022). ちやうぜつソフトウェア設計入門: PHP で理解するオブジェクト指向の活用.

- ソフトウェア設計全般について体系的に学べる

高山泰基. オブジェクト指向設計実践ガイド: Ruby でわかる進化しつづける柔軟なアプリケーションの育て方.

角征典, & 高木正弘. (2018). Clean architecture: 達人に学ぶソフトウェアの構造と設計.

Vモデル <https://ja.wikipedia.org/wiki/V%E3%83%A2%E3%83%87%E3%83%AB>

凝集度 <https://ja.wikipedia.org/wiki/%E5%87%9D%E9%9B%86%E5%BA%A6>

結合度
<https://ja.wikipedia.org/wiki/%E7%B5%90%E5%90%88%E5%BA%A6#:~:text=%E7%B5%90%E5%90%88%E5%BA%A6%E5%BC%88%E3%81%91%E3%81%A4%E3%81%94%E3%81%86%E3%81%A9,%E3%81%AB%E5%88%86%E3%81%91%E3%81%A6%E8%A1%A8%E7%8F%BE%E3%81%99%E3%82%8B%E3%80%82>

凝集度と結合度について https://affordd.jp/koha_hp/KeyWords/KW.Coupling.html

まとめ

ソフトウェア設計手法について紹介

- 良いコードとは(命名・分割)
- モジュール分割(凝集度・結合度)