第11回　設計図フォーラム

良い設計良いコード

2025年7月11日

山田 大介
ビースラッシュ株式会社

本日の内容

1．ソフトウェア疲労

2．アーキテクチャの実装ビュー

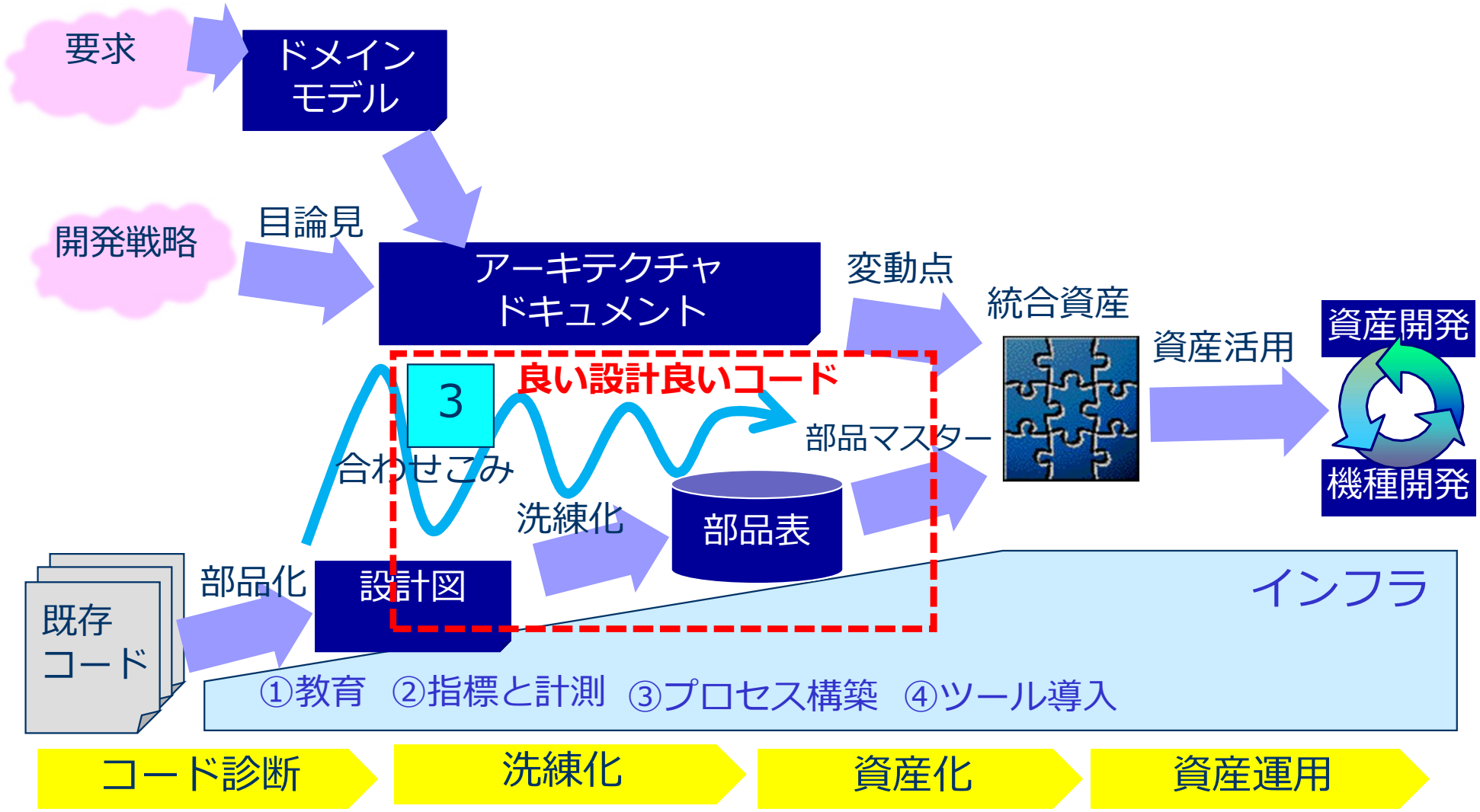
3．コード原則

- 3-1. 部品化コード原則
- 3-2. 構造化コード原則

4．まとめ：セルフチェック

ソフトウェアアーキテクト活動

アーキテクチャとコードの同期で戦略的開発を実現



1. ソフトウェア疲労

ソフトウェア疲労[静的] ソースコードはこんな状況に

	症状	想定原因	現象
S1	**一筆書き**	そもそも設計していない	ひとつの関数やファイルが長い 作った人にしかわからない（作った人も？）
S2	**クローン**		同じコード断片が点在。修正漏れが発生する. grep検索が最も便利な開発ツール
S3	**神様データ**	設計技法を使いこなしていない	すべてを支配しているデータが存在する 修正の波及範囲が大きい
S4	**中央集権**		ひとつのファイルにたくさんの関数がある いつも同じファイルを修正している
S5	**スパゲッティ**		いろいろな関数を呼び出している 副作用が発生する（こちらを直すとあちらが）
S6	**老舗温泉旅館**	全体を見ていない アーキテクト不在	階層を越えた呼び出し／命名規則がない 引継ぎができない（そもそも説明できない）
S7	**一枚岩**		#includeしているファイルが多い／循環依存 分割コンパイルができない

こんな人に設計実装してもらいたい

- アーキテクトが期待する設計実装者

- アーキテクチャを崩さず
- ソフトウェア疲労を未然防止し
- アーキテクチャへフィードバックする

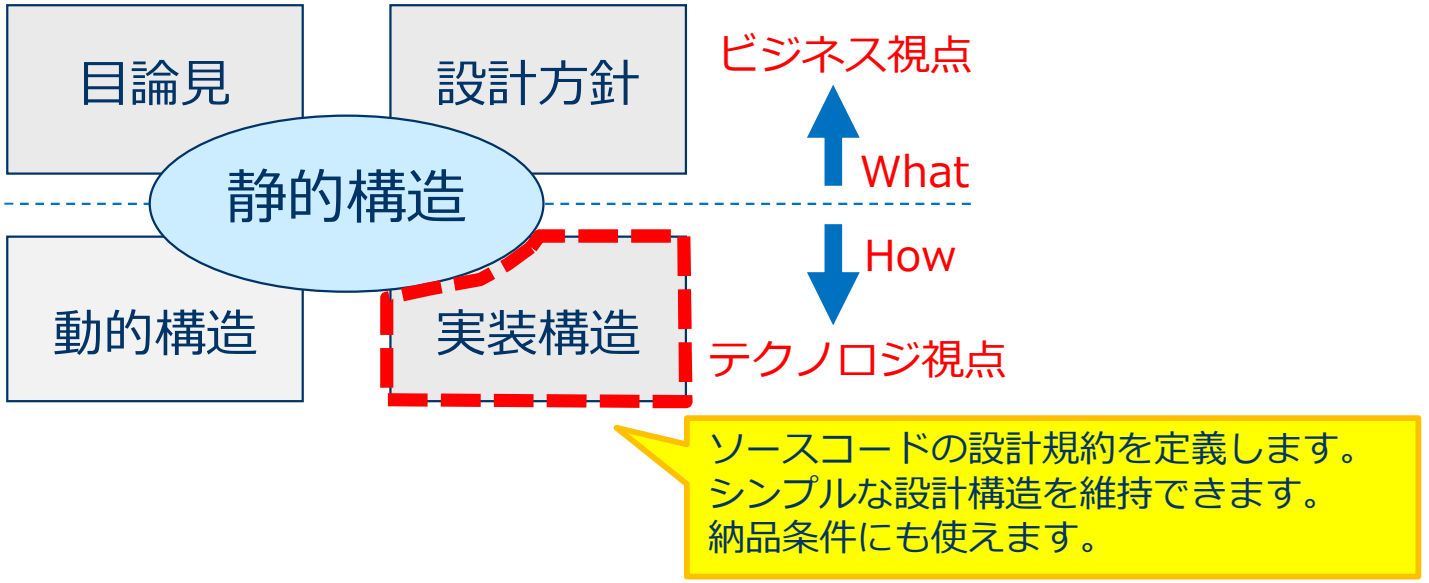
↓その結果

シンプルなコードを保ち
構造が見えて第3者レビューもできる
アーキテクチャを進化させる

2. アーキテクチャの実装ビュー

アーキテクチャの実装ビュー

- 実装ビューでソースコードのフォルダ構成と設計規約を定義
- 開発者へ周知して、設計構造の劣化防止と不具合の未然防止を実現



視点（ビュー）	図表（モデル）	表現内容
概念ビュー	目論見リスト	商品の特徴や差別化を定義する
方針ビュー	設計方針リスト	目論見を実現するための設計方針を定義する
静的ビュー	静的構造図	全体構造を俯瞰する
動的ビュー	動的構造図	パフォーマンスなど設計の勘所を明確にする
実装ビュー	構成図とコード原則	ソースコードのフォルダ構成と図面化のためのコード原則を定義する

モジュール化コード規約：コード原則

- アーキテクチャ設計の実装ビュー
  - アーキテクトはコード規約を作り周知して不具合を未然防止する
  - 開発者と設計レビューでの共通言語として品質を作り込む

- モジュール化コード規則の例

#	コード規約	狙い	要点
1	部品化コード原則	最小の再利用単位	ヘッダファイルと実装ファイルのペア 部品単位の置換ができること
2	構造化コード原則	局所的な構造設計	静的な構造設計 動的なスレッド設計の基本
3	大規模コード原則 (アーキテクチャ)	大局的な構造設計	全体を俯瞰して 重要箇所をピンポイントで押さえる

1．部品化コード原則

最小の部品単位
ヘッダファイルと実装ファイルのペア

No	原則	要点	解説
1-1	ファイル分割	.hと.cがペア	同じ名称のヘッダファイルと実装ファイルを作る。
1-2	What名称	責務を「ひとこと」で表現	ファイル名称は問題ドメインの用語を使う。 同じ用語が重複していない。“Manager”や”Main”は使わない。
1-3	高凝集	凝集度を高く	実装ファイルに同一目的の変数を集める。 ファイル単位の凝集度として、フィールド変数が単一目的。
1-4	変数カプセル化	ファイル内変数	実装ファイルで変数をstatic宣言する。 変数のget/set関数は使わない。 状態変数の取りうる値はenum定義で命名する。
1-5	インタフェース公開	インタフェースと実装の分離	ヘッダファイルで公開インタフェースを定義する。 公開インタフェースを呼び出すことで役割分担して動く。
1-6	入口ひとつ 出口ひとつ	途中で戻らない	関数の途中でreturnしない。 パラメータエラーの早期リターンは許容。
1-7	走り切り	途中で待たない	制御スレッドの途中で待たない。 ※制御スレッドとは、関数の通り道

2．構造化コード原則

局所的な構造設計
静的な構造設計と
動的なスレッド設計の基本

BACKSLASH DESIGN

No	原則	要点	解説
2-1	単方向依存	BOSSを作る レベル化	上位が下位のサービスを利用する関係を形成する。 実装ファイルで利用ヘッダファイルをインクルードする。
2-2	入出力分割	垂直分割 IO分離 STS分割	入力部と出力部を別ファイルにする。 源泉ー変換ー吸収というSTS構造を意識する。 ※STS：Source-Transform-Sink
2-3	求心遠心	物理→論理→物理 のデータの流れ	源泉での物理情報を意味付けしながら上位へ伝える。 上位からの意味情報を具体化しながら吸収へ伝える。
2-4	疎結合	結合度を弱く	実装ファイルで、ファンアウト数を7つ以内にする。(ヘッダインクルード数と関数呼出数) 引数は3つ以内。グローバル変数は使わない。
2-5	直交性	クローンコードを 作らない	アクションは1箇所だけにして、アクションの組み合わせで機能が出せるようにする ※アクションとは関数内の処理
2-6	対称性	シンメトリ	対となるアクションは対称となる箇所に一つだけ書く。 例えば、openに対してその対称の位置にcloseを書く。
2-7	上位トランジション	遷移とアクション の分離	状態変数を書き換える個所をひとつにする。 現在状態とイベントの組合せで次状態を決めて、 アクション内部の条件判断で次状態を決めない。

3．大規模コード原則

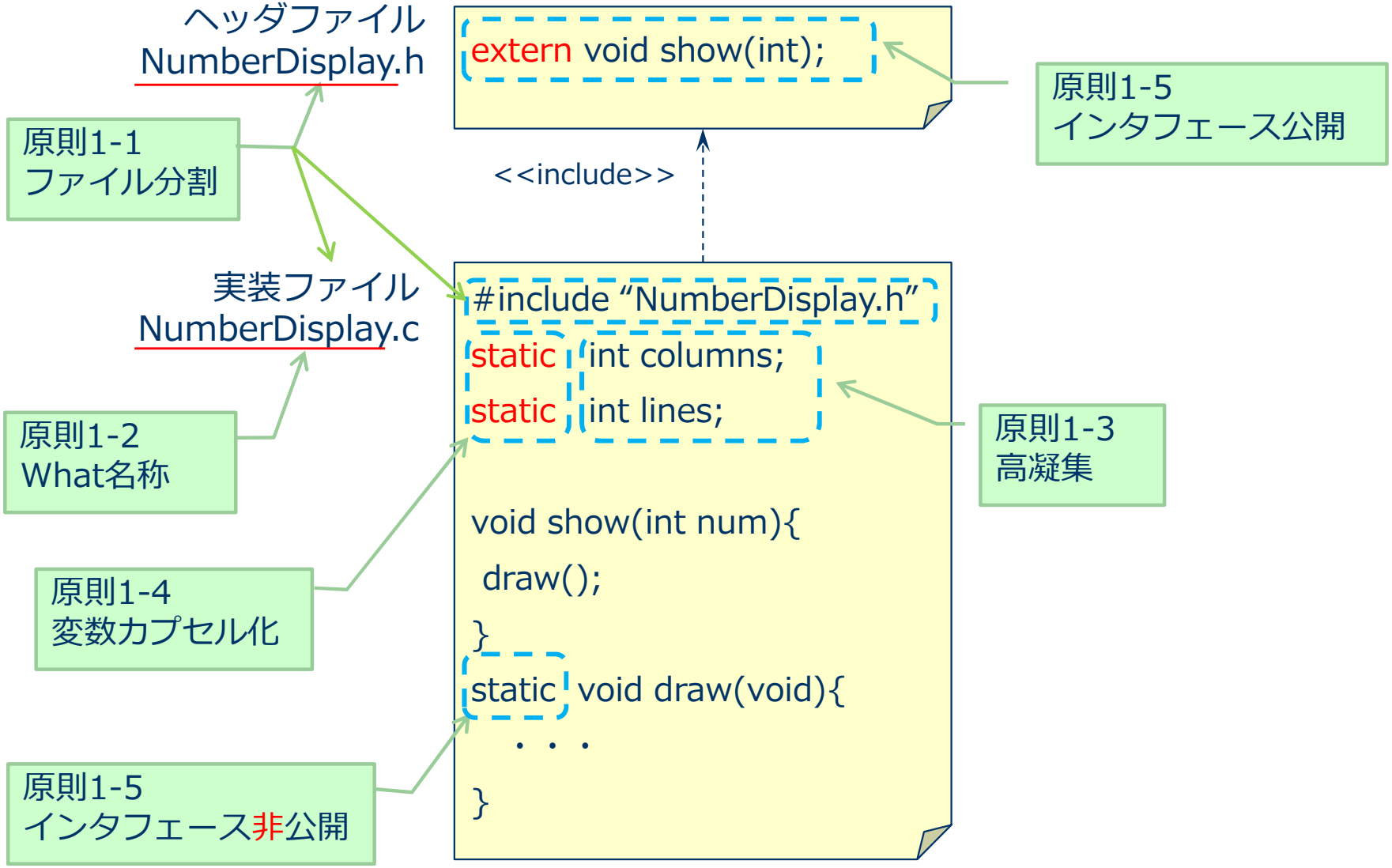
大局的な構造設計
　全体を俯瞰して
　重要箇所をピンポイントで押さえる

No	原則	要点	解説
3-1	コンポーネント粒度	全体俯瞰	フォルダ構成を決めてフォルダ単位で全体を俯瞰する。 ファイルやクラスよりも大きな粒度で見える化する。
3-2	レイヤー化	三層構造	アプリ層／ミドル層／ドライバ層の三層構造。
3-3	横断的関心の分離	垂直パーティション 初期化ルート エラー伝搬ルート	初期化や異常系を垂直パーティションで分離。 異常の発生と復旧の伝達ルートを作る。 正常系の構造をできる限り崩さない。
3-4	異ドメインの分離	垂直パーティション	UI部や通信部を垂直パーティションで分離。 正常系の三層構造をできる限り崩さない。
3-5	依存性マトリクス	厳密な階層化	ひとつ下の階層のみに依存する。 階層を飛び越えた依存や逆方向依存を排除。
3-6	エラー伝播ルート	正常系を崩さない	エラー発生時の再実行／後退復帰／縮退処理／異常終了の 再実行と後退復帰は正常系の形状で対処し、 縮退処理と異常終了は、横断的関心側へ通知し、対処する。
3-7	静動分離	上位スケジューラ	静動分離（制御スレッド起点と論理構造を分離）する。 処理の流れの中でできる限り周期を作らない。

BACKSLASH DESIGN

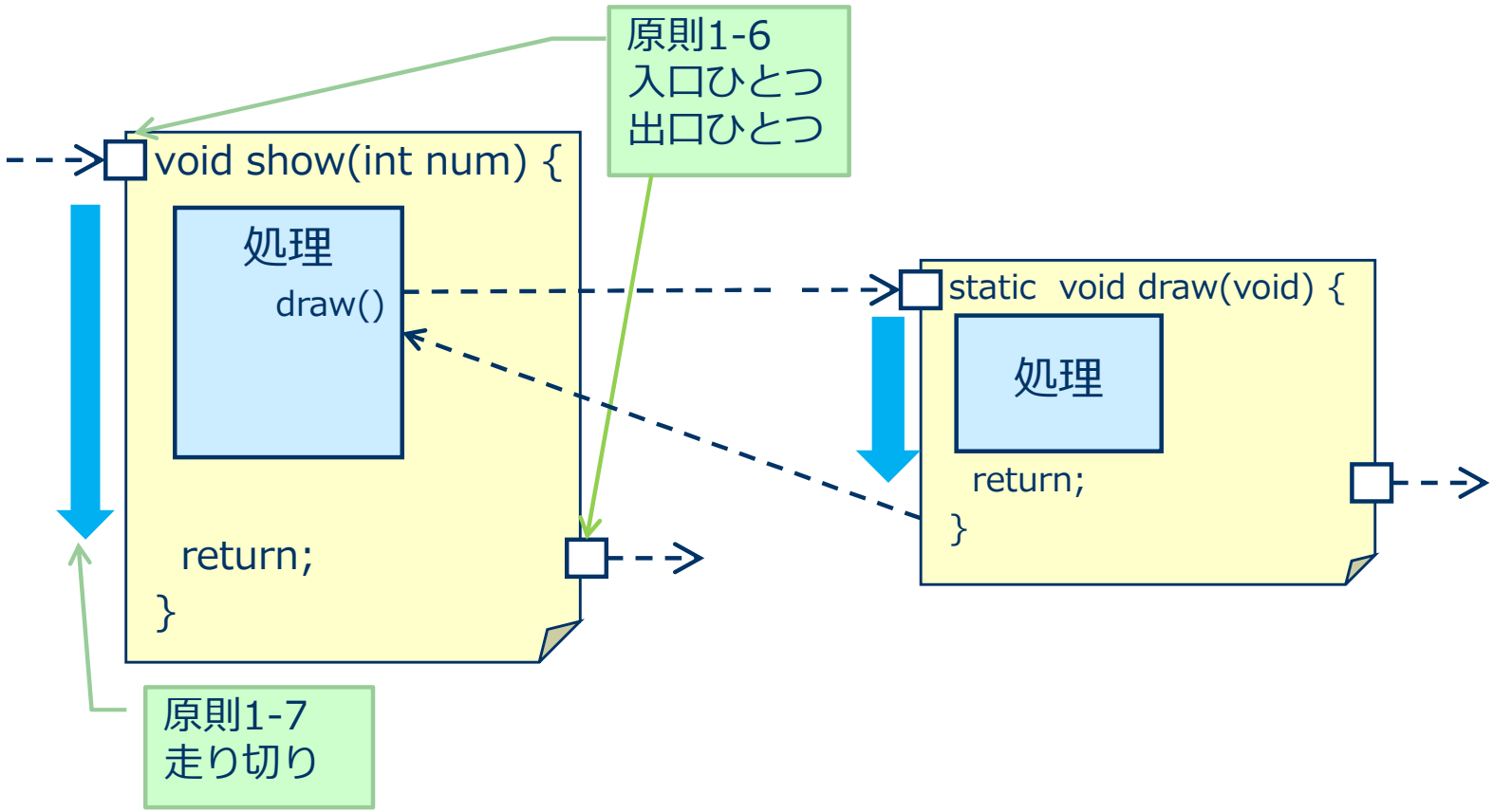
3-1. 部品化コード原則

部品化コード原則



部品化コード原則：制御スレッド編

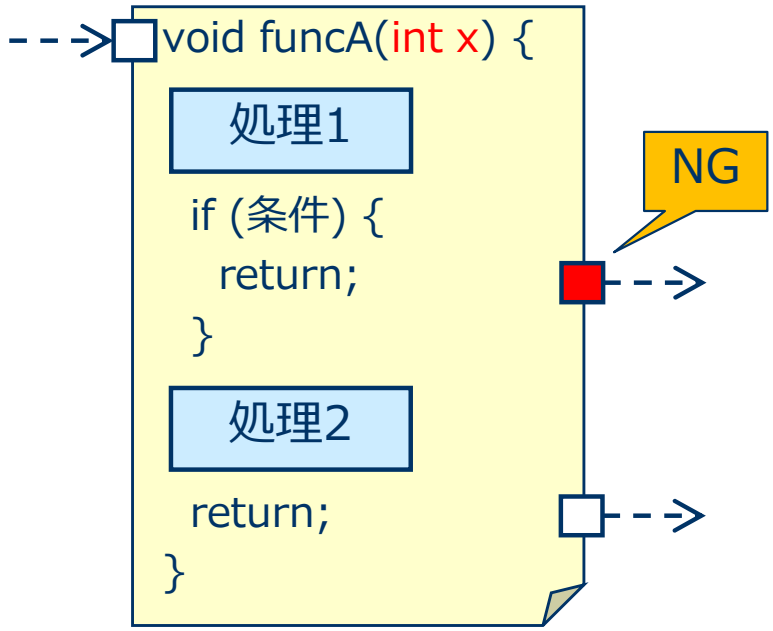
- 関数の入口から出口までの制御スレッドを直進させる
  - 制御スレッドとはプログラムの通り道のこと
  - 制御スレッドの出口を複数作らない
  - 制御スレッドの途中でイベント待ちや遅延処理をしない



途中リターンや途中待ちをしない

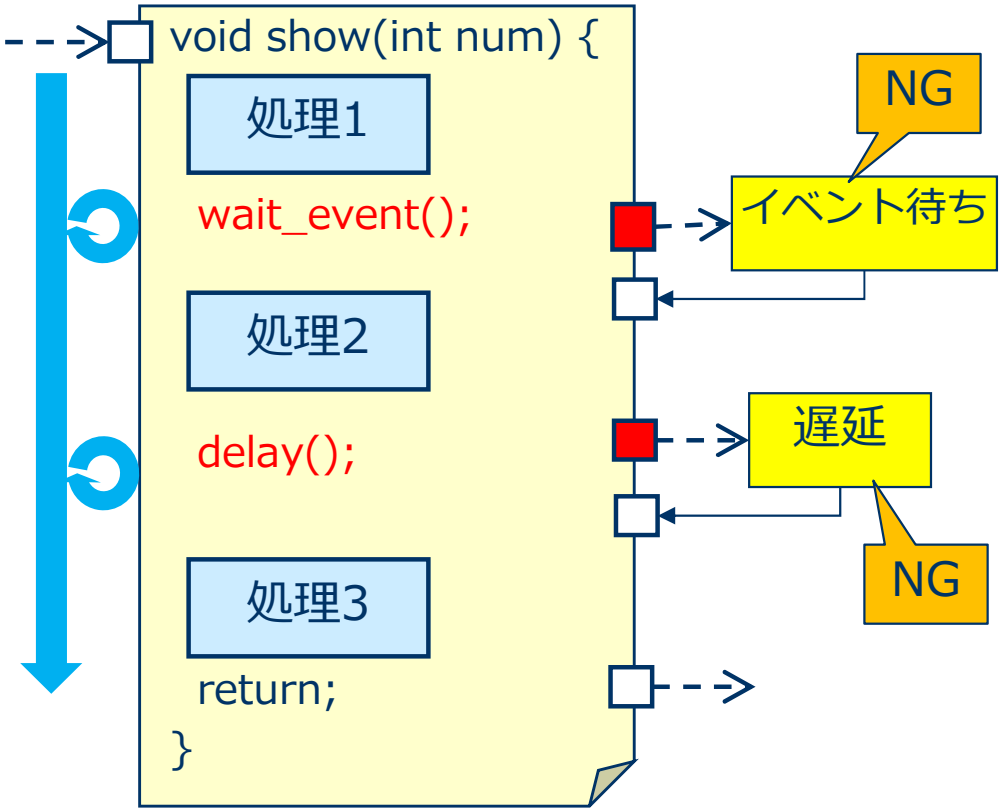
- 処置途中でのreturn
  - 関数の開始処理と終了処理の対称さが無くなり、予期せぬ不具合発生の危険性
  - 凝集度が低く、単体テストは難しくなる

途中リターン



- 制御スレッドの途中で待たない
  - イベント待ちや遅延処理をしない
  - 再現性の低い不具合につながる

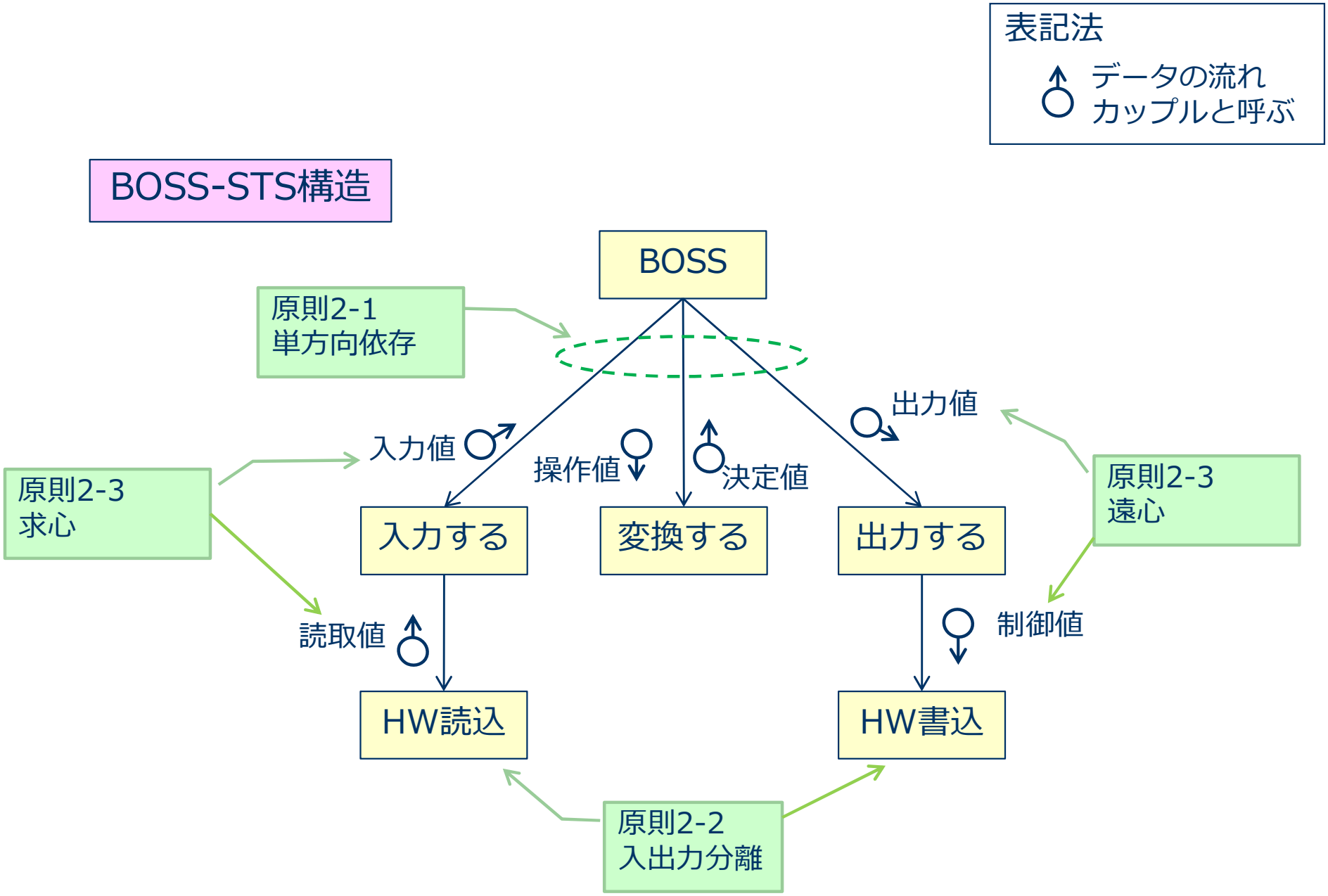
途中で待つ



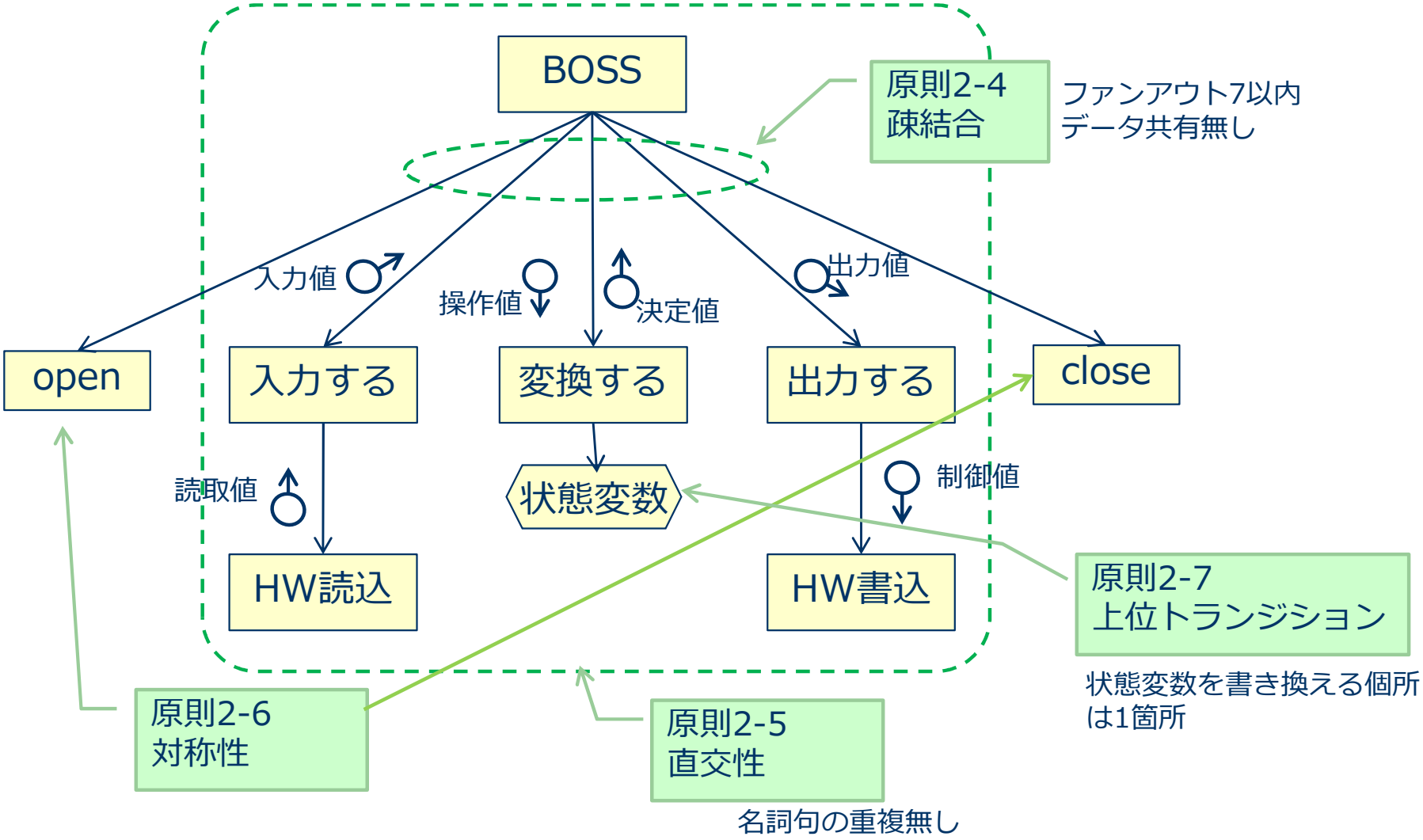
BACKSLASH DESIGN

3-2. 構造化コード原則

構造化コード原則：基本形

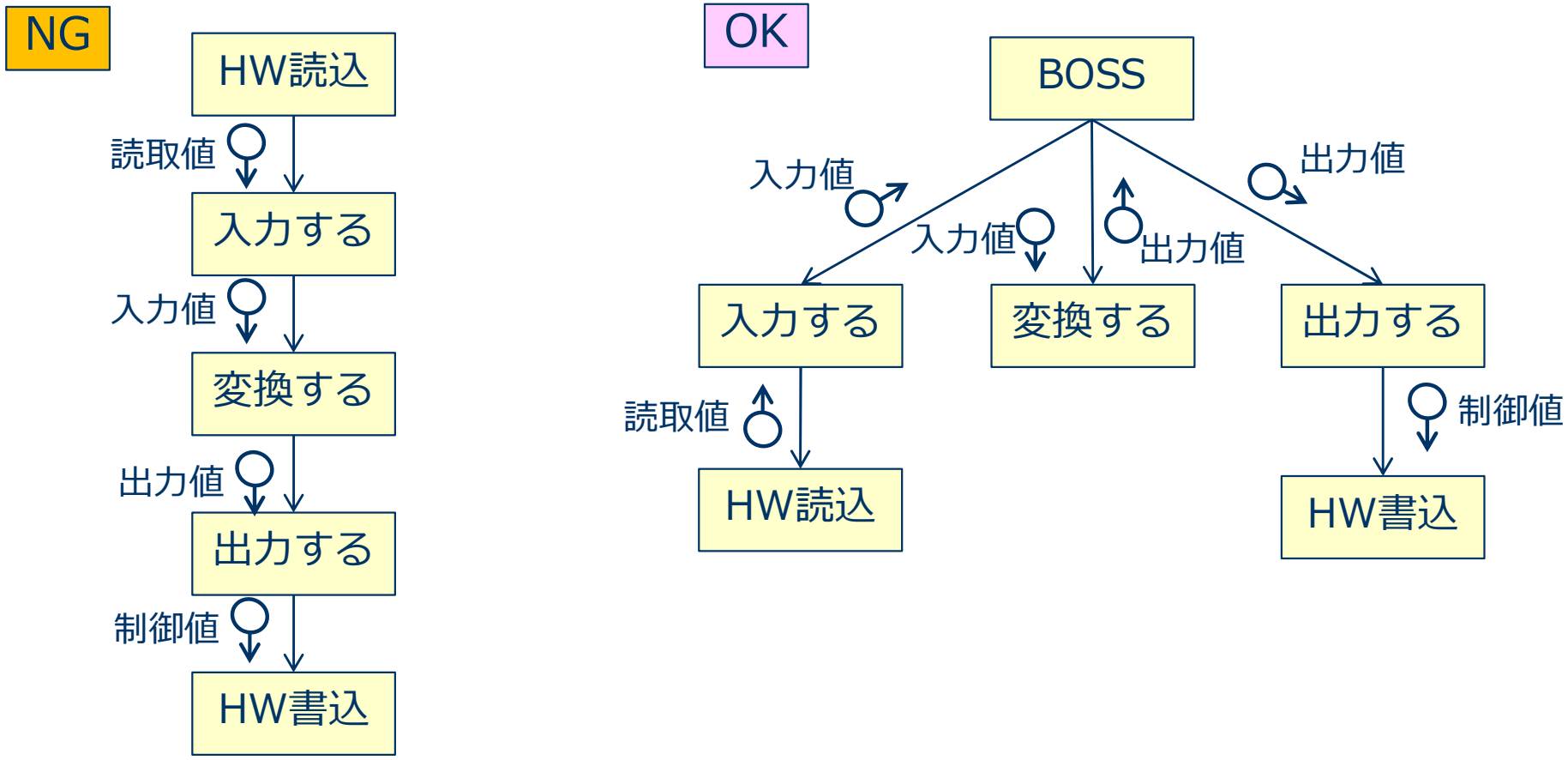


構造化コード原則：全体バランス



一筆書きとBOSS-STS構造

- 一筆書きは制御の流れとデータの流れが同じ
  - 影響範囲が分からない
- BOSS-STS構造は、上下の利用構造と求心遠心のデータの流れ
  - 変更箇所が特定しやすく、他への影響も少ない



4．まとめ

コード原則チェックリスト

- 項目を理解できると、図面とコードを同期させた開発ができます
- 6割以上が「〇」のエンジニアを探しましょう

典型的なケース

No	部品化コード原則	〇／×
1-1	ファイル分割	〇
1-2	What名称	×
1-3	高凝集	×
1-4	変数カプセル化	〇
1-5	インタフェース公開	×
1-6	入口ひとつ 出口ひとつ	×
1-7	走り切り	×

No	構造化コード原則	〇／×
2-1	単方向依存	×
2-2	入出力分割	×
2-3	求心遠心	×
2-4	疎結合	×
2-5	直交性	×
2-6	対称性	×
2-7	上位トランジション	×

No	大規模コード原則	〇／×
3-1	コンポーネント粒度	×
3-2	レイヤー化	〇
3-3	横断的関心の分離	×
3-4	異ドメインの分離	×
3-5	依存性マトリクス	×
3-6	エラー伝播ルート	×
3-7	静動分離	×

付録：コード原則セルフチェック

No	部品化コード原則	○／×
1-1	ファイル分割	
1-2	What名称	
1-3	高凝集	
1-4	変数カプセル化	
1-5	インタフェース公開	
1-6	入口ひとつ 出口ひとつ	
1-7	走り切り	

No	構造化コード原則	○／×
2-1	単方向依存	
2-2	入出力分割	
2-3	求心遠心	
2-4	疎結合	
2-5	直交性	
2-6	対称性	
2-7	上位トランジション	

No	大規模コード原則	○／×
3-1	コンポーネント粒度	
3-2	レイヤー化	
3-3	横断的関心の分離	
3-4	異ドメインの分離	
3-5	依存性マトリクス	
3-6	エラー伝播ルート	
3-7	静動分離	

アーキテクトと
コード原則の実践で
シンプルで構造図もあり
進化できるソフトウェアへ