機能ブロックと擬似コードを用いたS/W開発事例

～機能ブロックと擬似コードをテストに用いる～

CodingStyle by AkioTORIMOTO

目指すところ

○多くのメンバで開発したシステムであっても，

ソフトウェアの構造は，**単一のモノ**となるように。

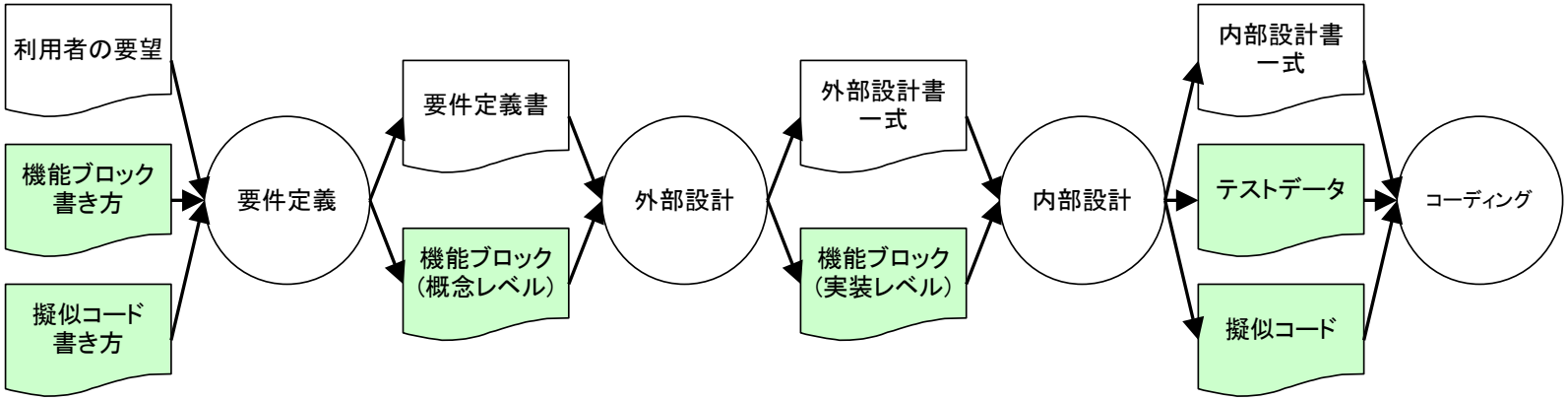
○ソフトウェアの**品質が高いレベル**になる方向に。

○ソフトウェアテストの**品質が均質**となる方向に。

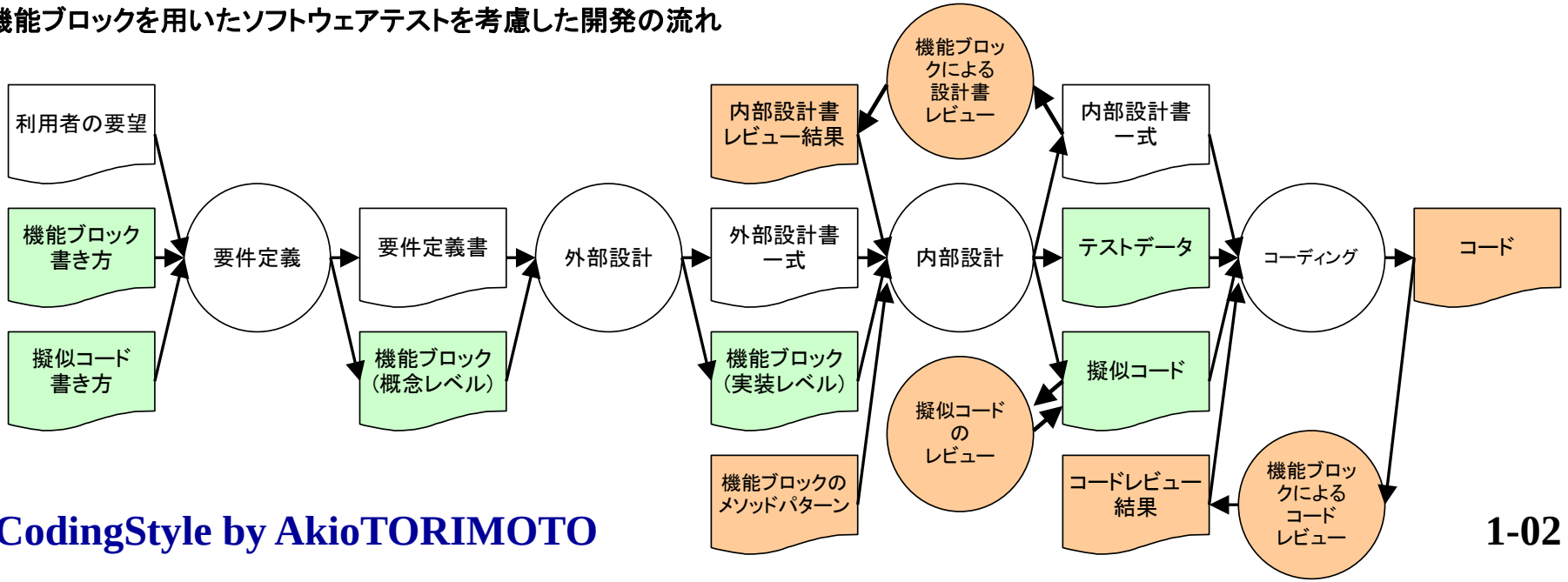
○**レビューで設計書とコード**の**バグをなくす**。

内部設計とコーディングを見直す

2012年度の事例発表での施策



機能ブロックを用いたソフトウェアテストを考慮した開発の流れ



2つのツール

～機能ブロックと擬似コード～

機能ブロックと擬似コードの特徴

- ○**すぐに始める**ことができる
- ○**つくる，確認，修正が簡単**
- ○つくる，確認，修正を**繰り返すことが簡単**
- ○**コードに直接，影響する**
- ○**好きな粒度**でつくることができる
- ○**仕様書との間で相互に変換し易い**
- ○**機能ブロックと擬似コードの間で変換し易い**

仕様書と機能ブロックと擬似コード

[仕様]
システムが
情報(整数の値)を受け取る入力機能と
情報が整数か否かを判定する判定機能と
データベースに情報を出力する出力機能を
持つソフトウェアの概念設計

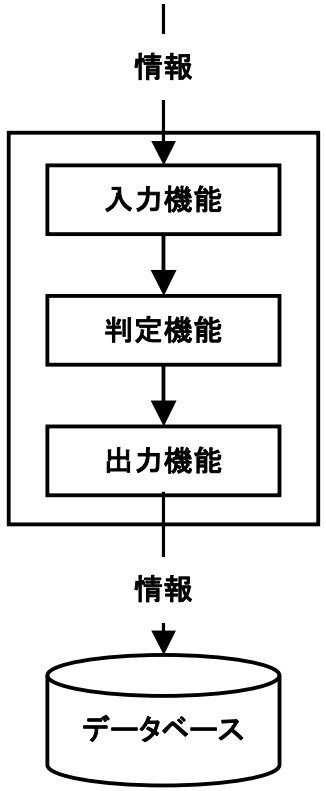
相互変換

相互変換

```
int     main( void )
    if ( 入力機能( 情報を入力する領域 ) == 入力失敗 ) {
        return 0;
    }
    if ( 判定機能( 情報 ) == 判定結果は整数でない ) {
        return 0;
    }
    出力機能( 情報 );
    return 0;
}
```

擬似コード

相互変換



機能ブロックでの概念設計

開発，テストプロセスとの対応

ツール	開発プロセス				テストプロセス			
	要件定義	外部設計	内部設計	コーディング	要求テスト	仕様テスト	設計テスト	コードレビュー
機能ブロック	○	○	○	○	○	○	○	○
擬似コード			○	○			○	○

凡例
○　：　そのプロセスに用いることができる。

擬似コード

～設計しながらコーディング～

擬似コードの用い方

○**分岐**，**選択**，**繰り返し**，**格納**と**コメント**で書く

○**スコープ**を意識し，**インタフェースと**

データの管理を明確にするように書く

○メンバで**共通の言葉**を用いて書く

○**擬似コードを置き換える**，あるいは，

コメント文にすることで，

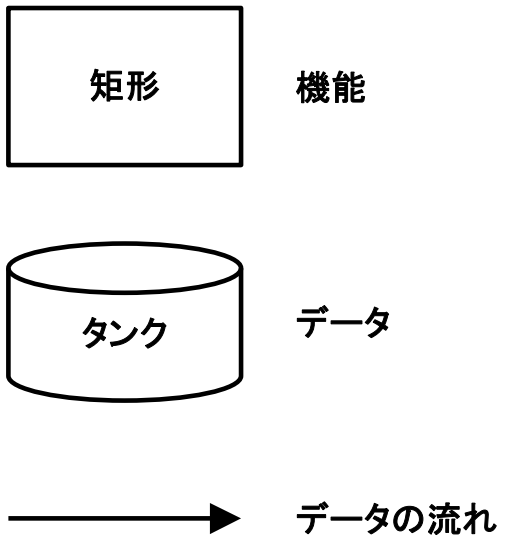
コーディングが行えるように書く

機能ブロック

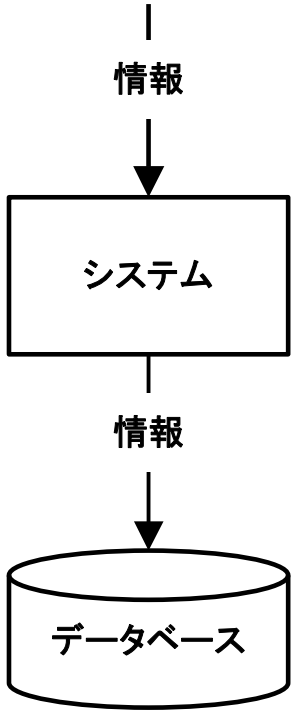
～3つの図形で，アウトラインを描く～

機能ブロックとは

機能の関係を矩形とタンクと矢印で整理したもの。



機能ブロックを構成する図形



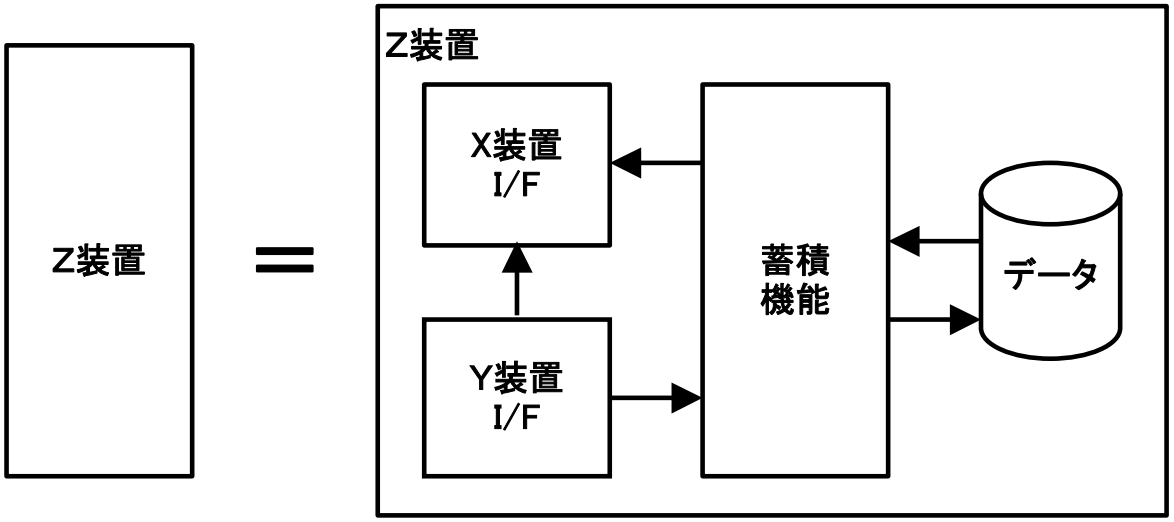
機能ブロックのイメージ

機能ブロックとは

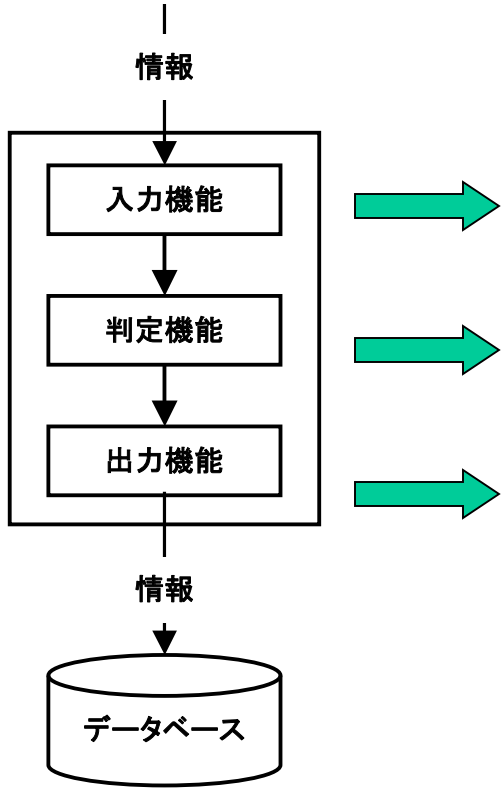
1機能＝n×機能＋m×データ

n ＝ 1～

m ＝ 0～



機能ブロックを擬似コードに



機能ブロックでの概念設計

```
int    main( void )
    if ( 入力機能( 情報を入力する領域 ) == 入力失敗 ) {
        return 0;
    }
    if ( 判定機能( 情報 ) == 判定結果は整数でない ) {
        return 0;
    }
    出力機能( 情報 );
    return 0;
}
```

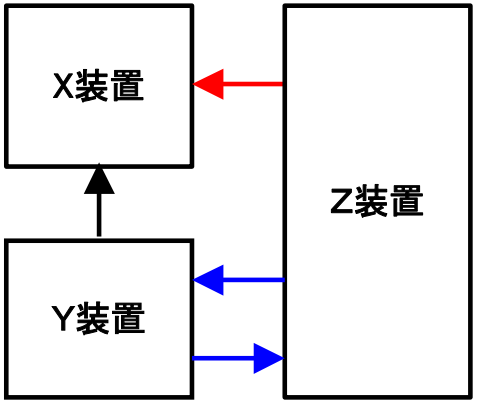
機能ブロックのレベルとタスク

仕様	概念レベル	実装レベル	タスク	コードレベル
1 :	n			
	1 :	m		
		1 :	p	
			1 :	q

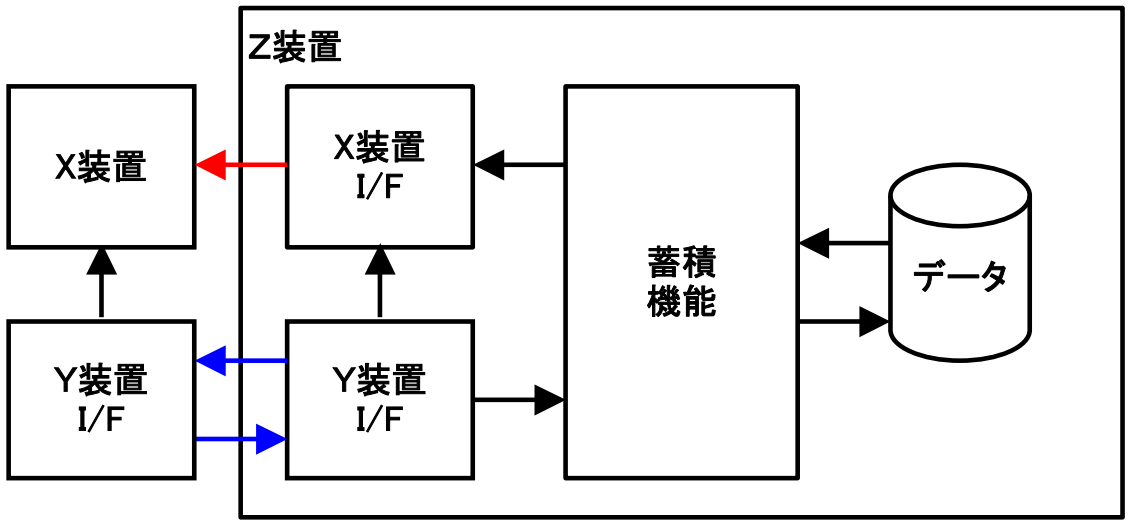
n = 1～
m = 1～
p = 1～
q = 1～

機能ブロックの詳細化 –概念，実装–

ポイント
・機能ブロックの粒度が変わっても，データの流れが変わらないこと
（X装置～Z装置間のI／F，Y装置～Z装置間のI／Fは，同じとなるように，機能ブロックを描く。）



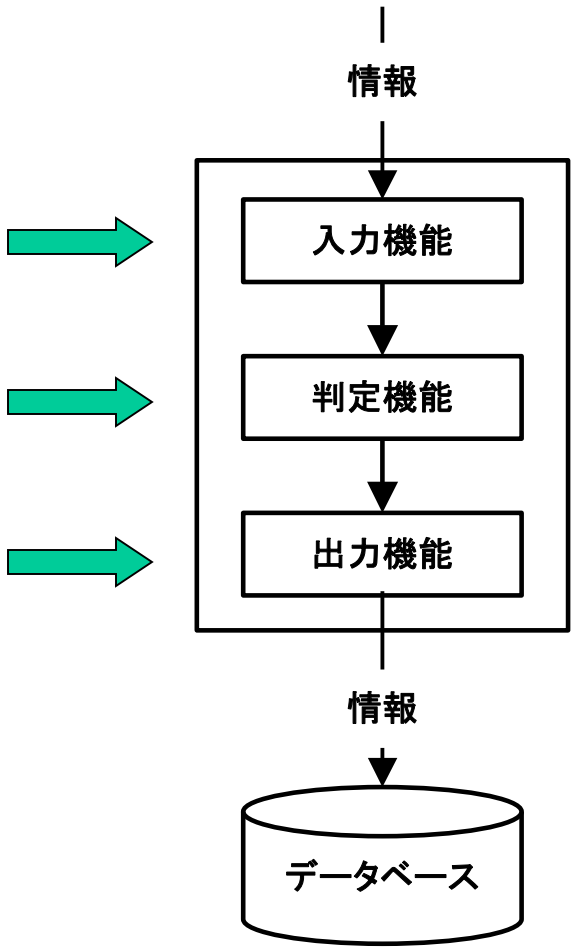
装置単位の機能ブロック



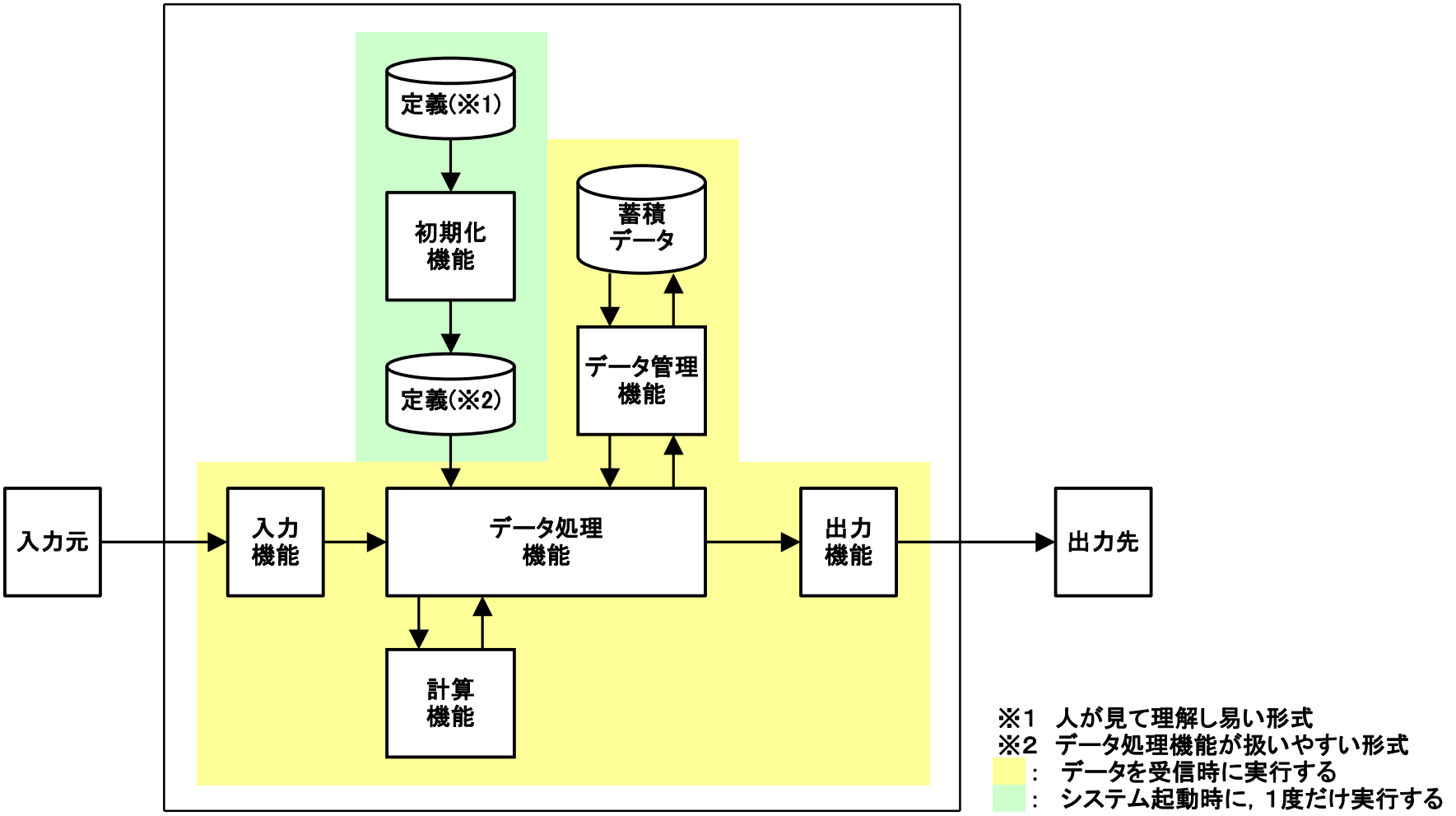
機能ブロック（Z装置のみ，装置内の機能ブロック）

機能ブロックの詳細化 –コードレベル–

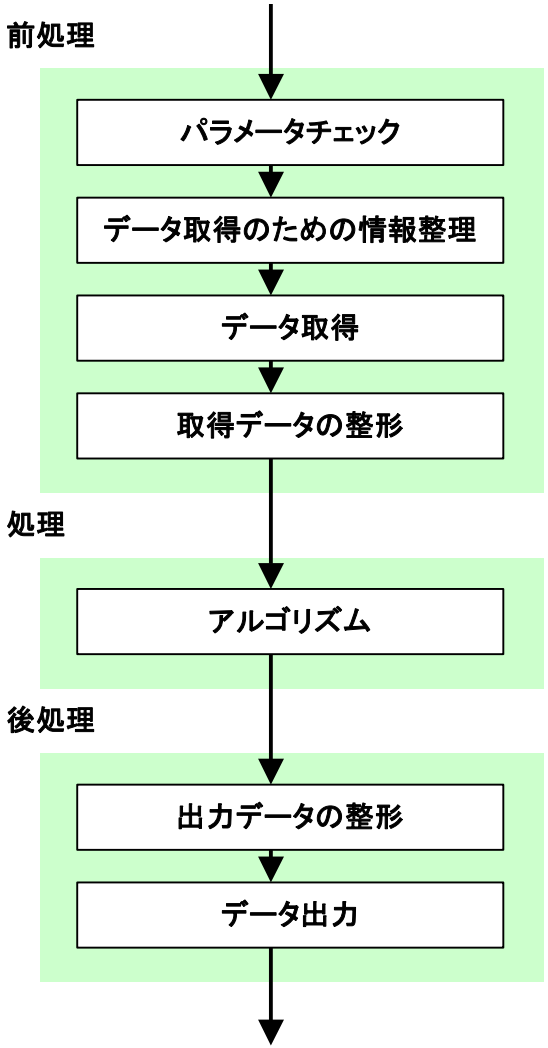
```
int    main( void )
{
    if ( 入力機能( 情報を入力する領域 ) == 入力失敗 ) {
        return 0;
    }
    if ( 判定機能( 情報 ) == 判定結果は整数でない ) {
        return 0;
    }
    出力機能( 情報 );
    return 0;
}
```



機能ブロックの1機能の基本構成



機能ブロックのメソッドパターン



目指すところ

- 各機能は，機能単位でテストが行えること。
- データ取得とデータ出力以外は，開発言語だけに依存すること。
- 前処理，後処理の機能は省くことが出来る。
- 前処理，後処理の機能は複数存在することが出来る。
- 前処理の機能は順序を入れ替えることが出来る。
- データ取得，アルゴリズム，データ出力を同じスコープにしない。

テストし易い構造のソフトウェア

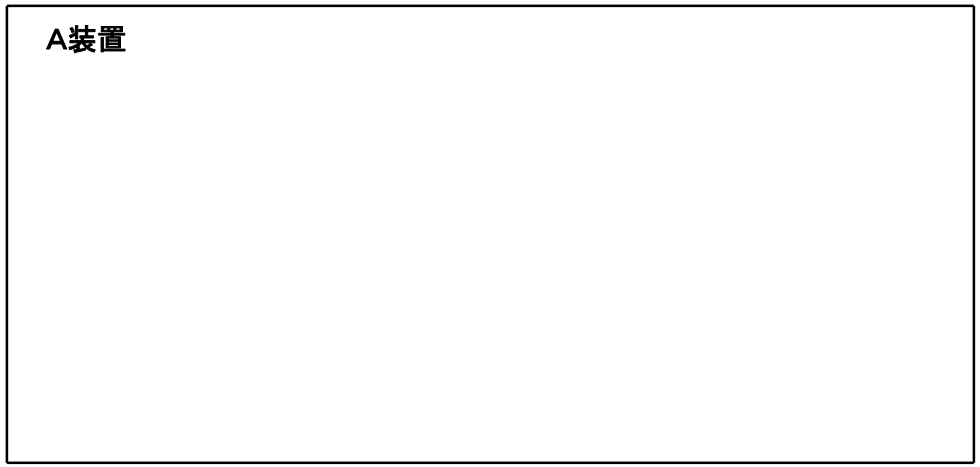
要件定義～外部設計

～機能ブロックで始める設計とデザインレビュー～

機能ブロックを描く

[仕様]
A装置は，値Bを受け取る。
ファイルに定義されている
値Dが1のとき，受け取った値Bを2倍して，出力する。
計算結果は保存する。
ファイルはA装置起動中に変更されない。

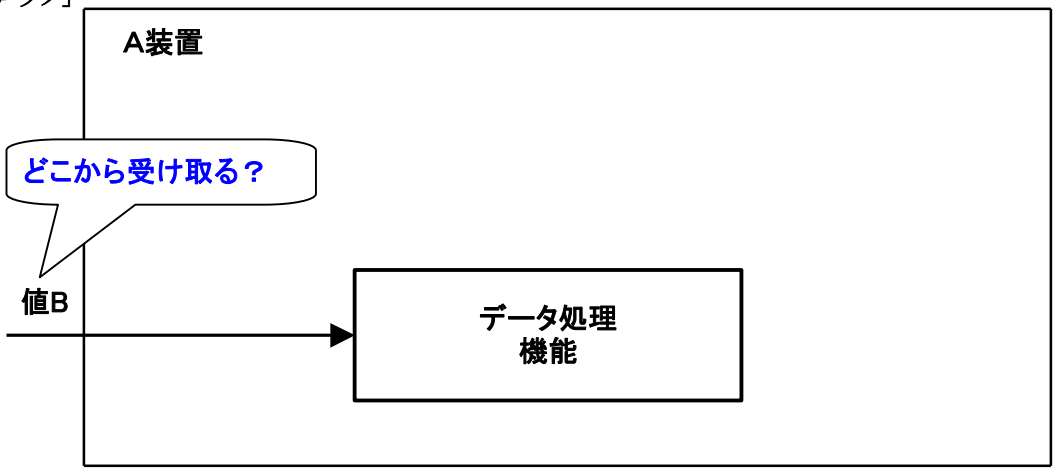
[機能ブロック]



機能ブロックを描く

［仕様］
A装置は，値Bを受け取る。
ファイルに定義されている
値Dが1のとき，受け取った値Bを2倍して，出力する。
計算結果は保存する。
ファイルはA装置起動中に変更されない。

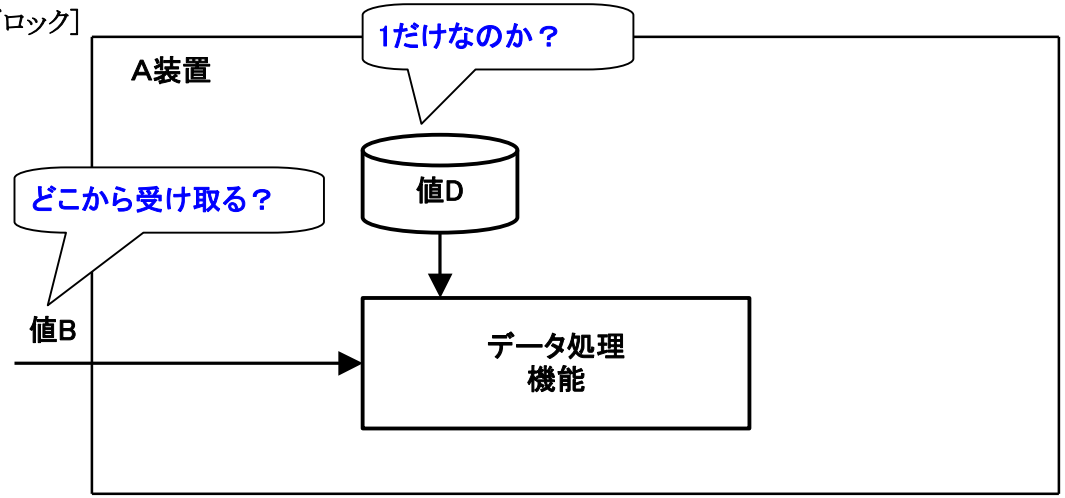
［機能ブロック］



機能ブロックを描く

[仕様]
A装置は，値Bを受け取る。
ファイルに定義されている
値Dが1のとき，受け取った値Bを2倍して，出力する。
計算結果は保存する。
ファイルはA装置起動中に変更されない。

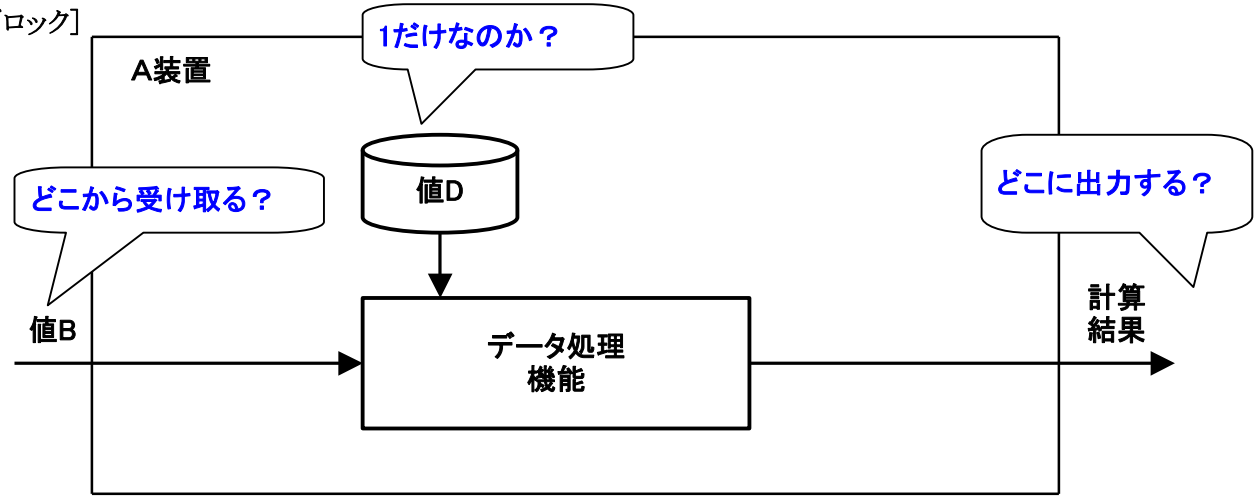
[機能ブロック]



機能ブロックを描く

[仕様]
A装置は，値Bを受け取る。
ファイルに定義されている
値Dが1のとき，受け取った値Bを2倍して，出力する。
計算結果は保存する。
ファイルはA装置起動中に変更されない。

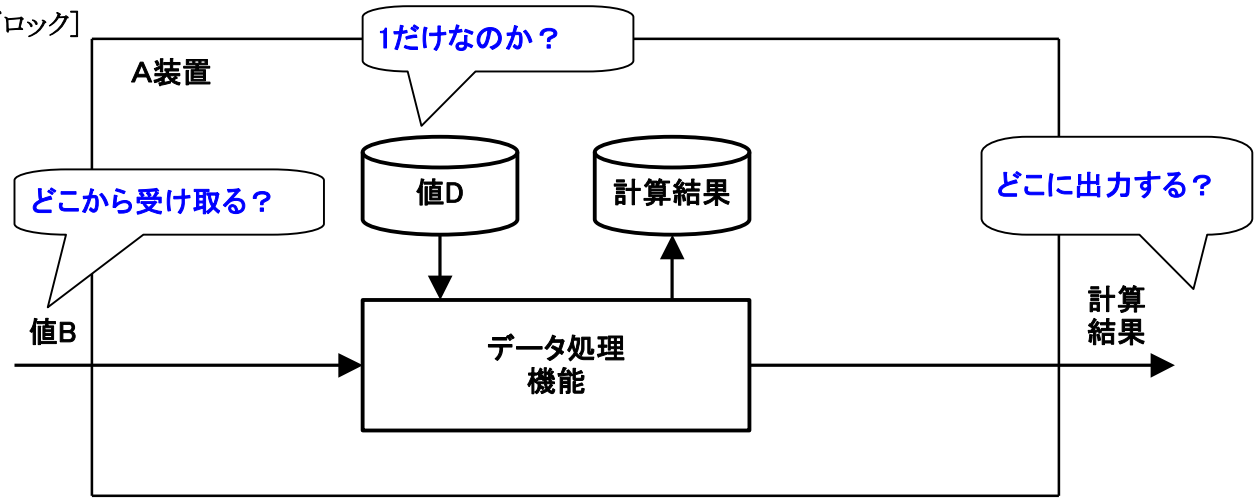
[機能ブロック]



機能ブロックを描く

［仕様］
A装置は，値Bを受け取る。
ファイルに定義されている
値Dが1のとき，受け取った値Bを2倍して，出力する。
計算結果は保存する。
ファイルはA装置起動中に変更されない。

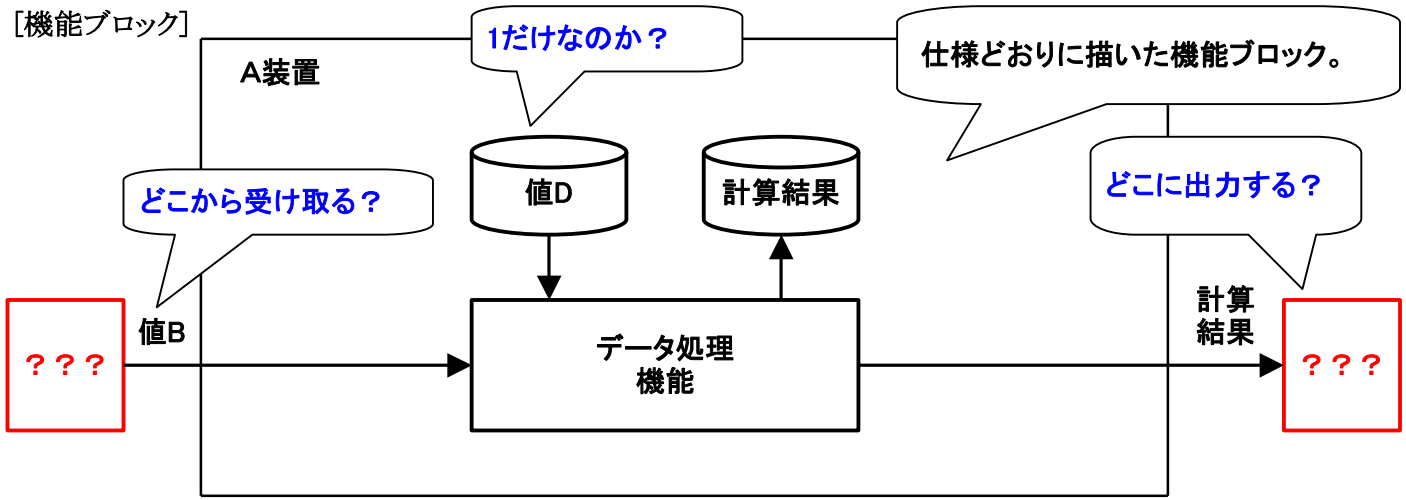
［機能ブロック］



機能ブロックを描く

［仕様］
A装置は，値Bを受け取る。
ファイルに定義されている
値Dが1のとき，受け取った値Bを2倍して，出力する。
計算結果は保存する。
ファイルはA装置起動中に変更されない。

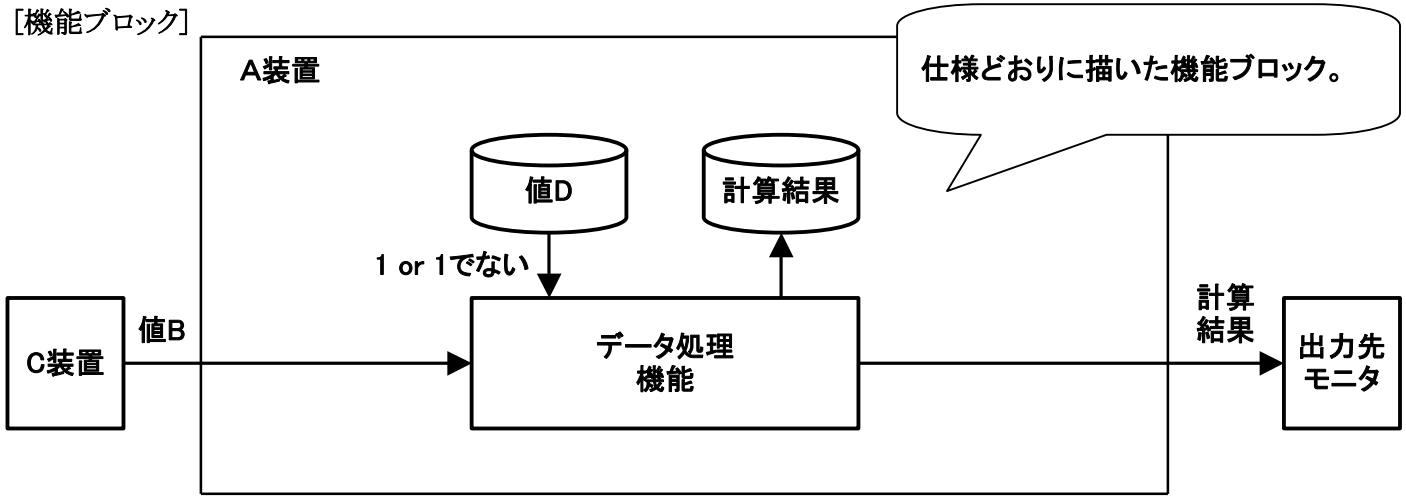
［機能ブロック］



機能ブロックを描く－レビュー後－

［機能ブロックのレビューにより修正した仕様］
A装置は，C装置より，値Bを受け取る。
ファイルに定義されている
値Dが1のとき，受け取った値Bを2倍して，出力先モニタに出力する。
値Dが1でないとき，受け取った値Bを，出力先モニタに出力する。
計算結果は保存する。
ファイルはA装置起動中に変更されない。

［機能ブロック］



内部設計～コーディング

～仕様どおりの設計から，より良い設計・製作へ～

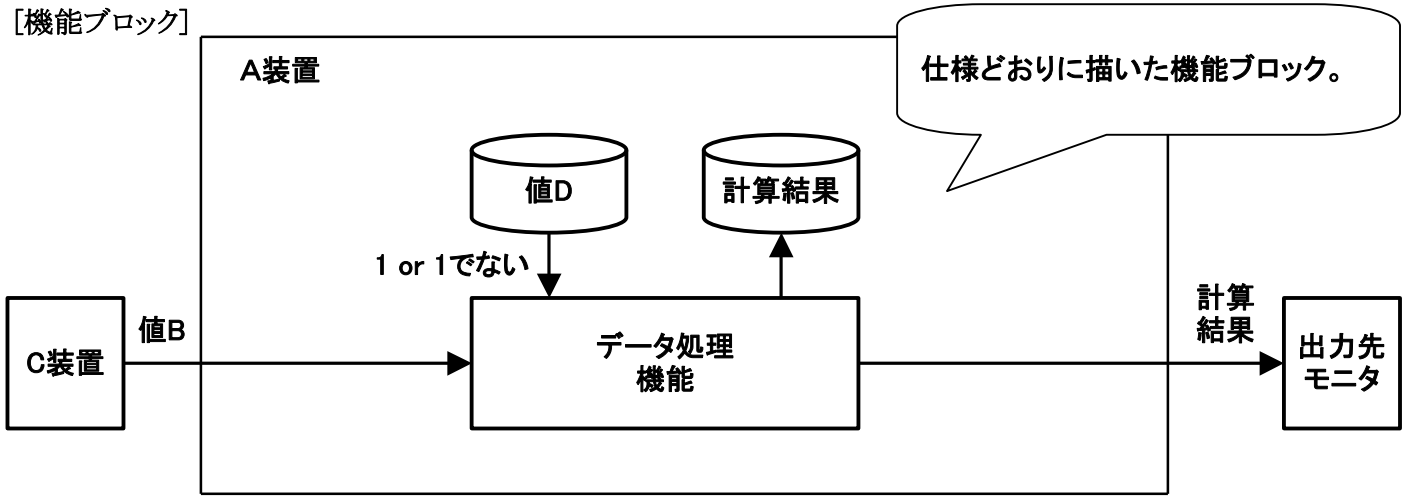
機能ブロックを描く－スコープの考慮－

○**依存性**のスコープ
　動作に必要な**機能を独立させる**
○**テスト**のスコープ
　ユニットテストフレームワークを用いたテストの
　対象範囲を明確にする
○**時間**のスコープ
　機能が**動作するタイミング**を明確にする
○**データ**のスコープ
　データの管理者，機能との関係を明確にする

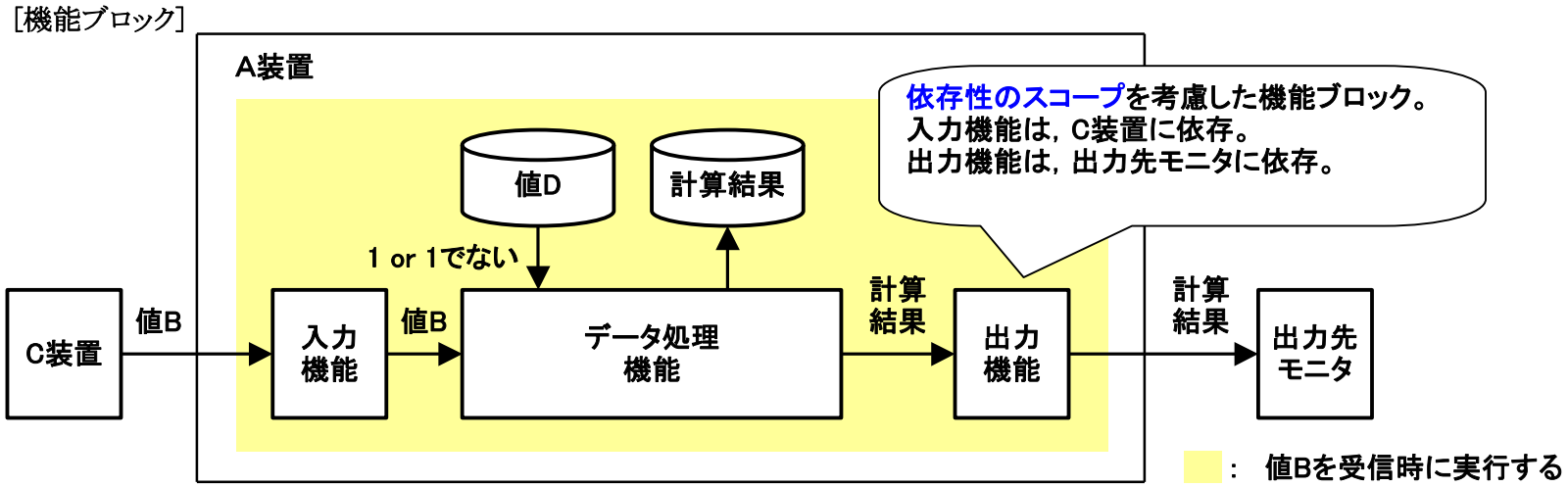
機能ブロックを描く－スコープの考慮－

［仕様］
A装置は，C装置より，値Bを受け取る。
ファイルに定義されている
値Dが1のとき，受け取った値Bを2倍して，出力先モニタに出力する。
値Dが1でないとき，受け取った値Bを，出力先モニタに出力する。
計算結果は保存する。
ファイルはA装置起動中に変更されない。

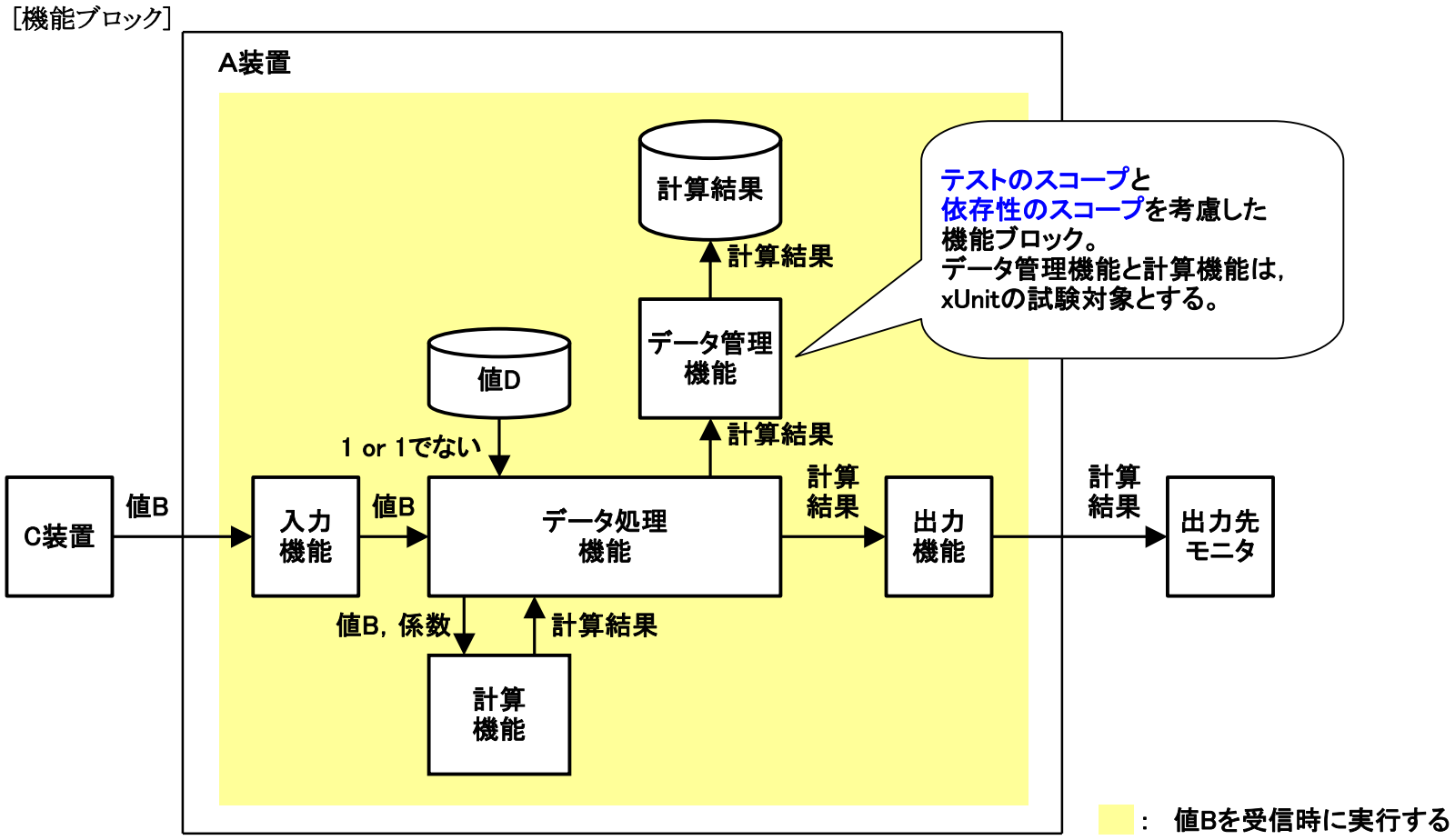
［機能ブロック］



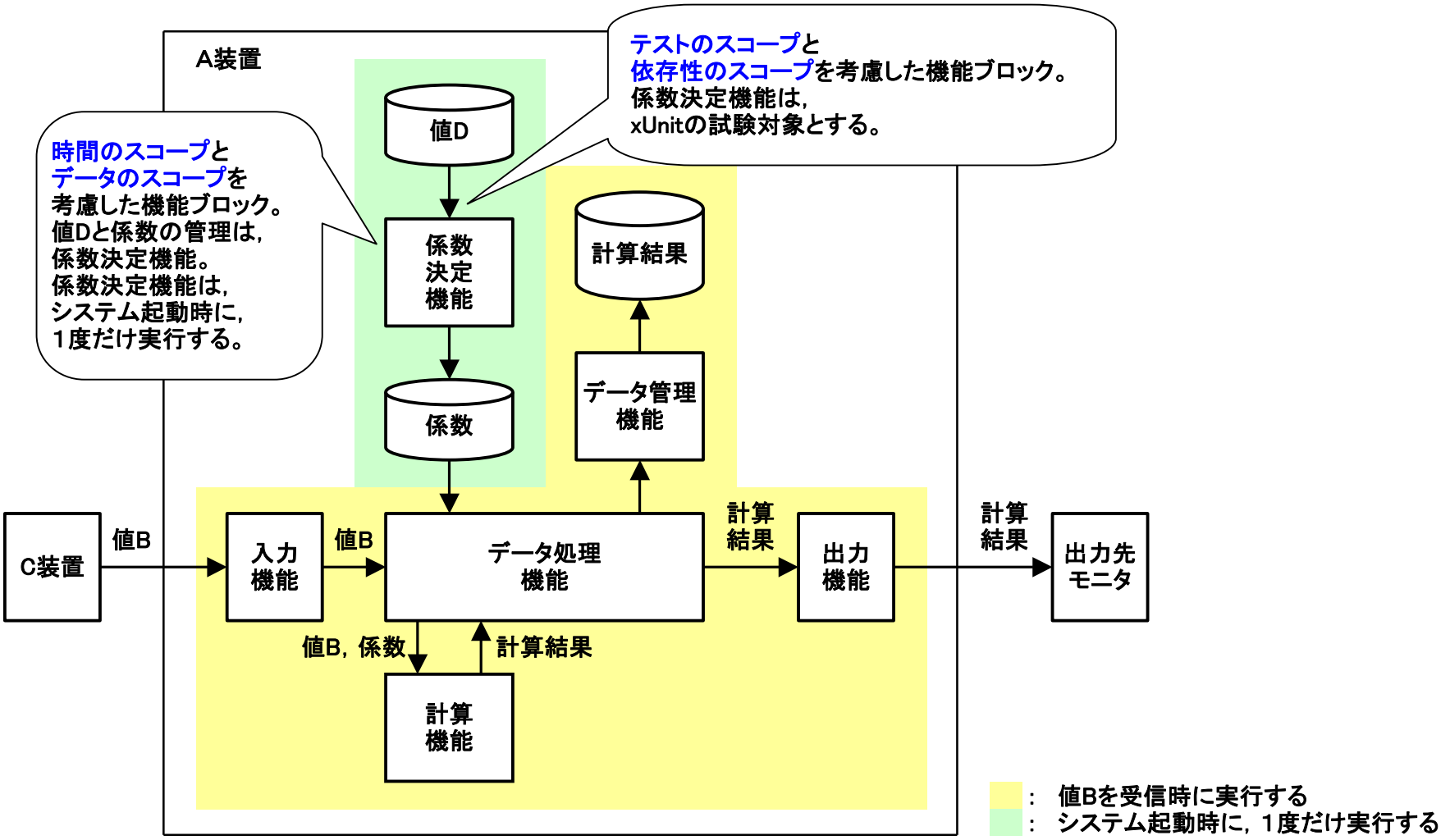
機能ブロックを描く－スコープの考慮－



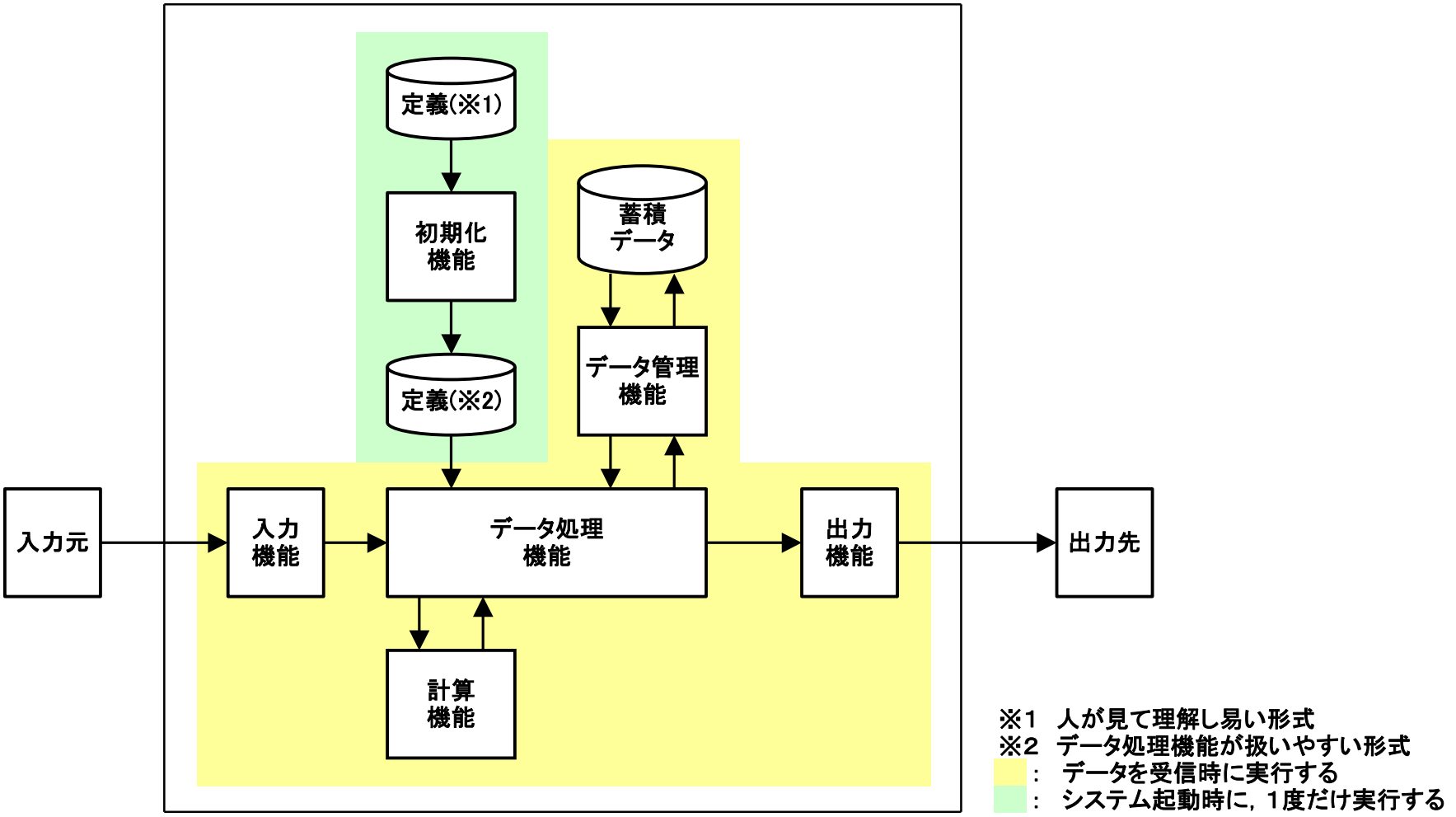
機能ブロックを描く－スコープの考慮－



機能ブロックを描く－スコープの考慮－



機能ブロックを描く－1機能の基本構成－



意図した設計，意図したコードへ

～テストツールとしての機能ブロックと擬似コード～

テストにかかるコスト

○**テストのし易さ**でコストはかわる

○**パラメータの数と値域**でコストはかわる

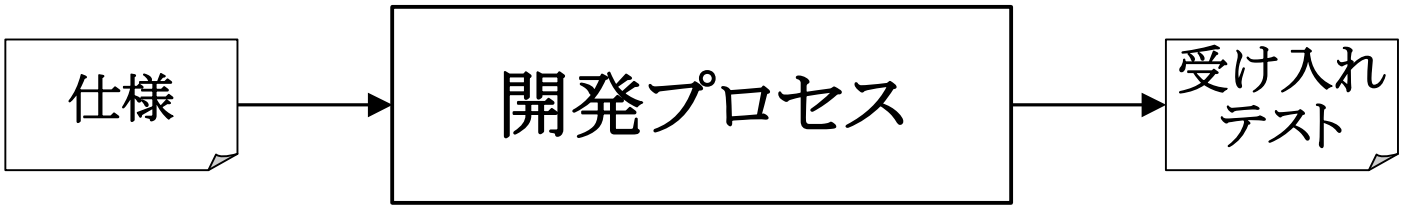
○**バグの改修時間**でコストはかわる

○**バグによる再テストの時間**でコストはかわる

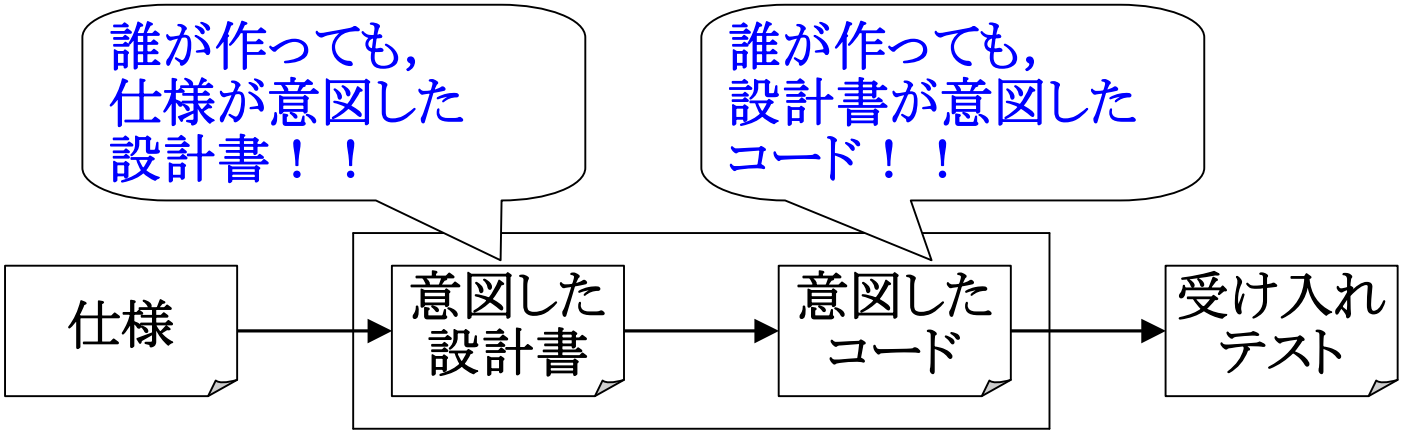
意図した設計で，意図したコードで，

意図したテストが行えれば，コストが少なくなる。

1つの仕様，1つの受け入れテスト

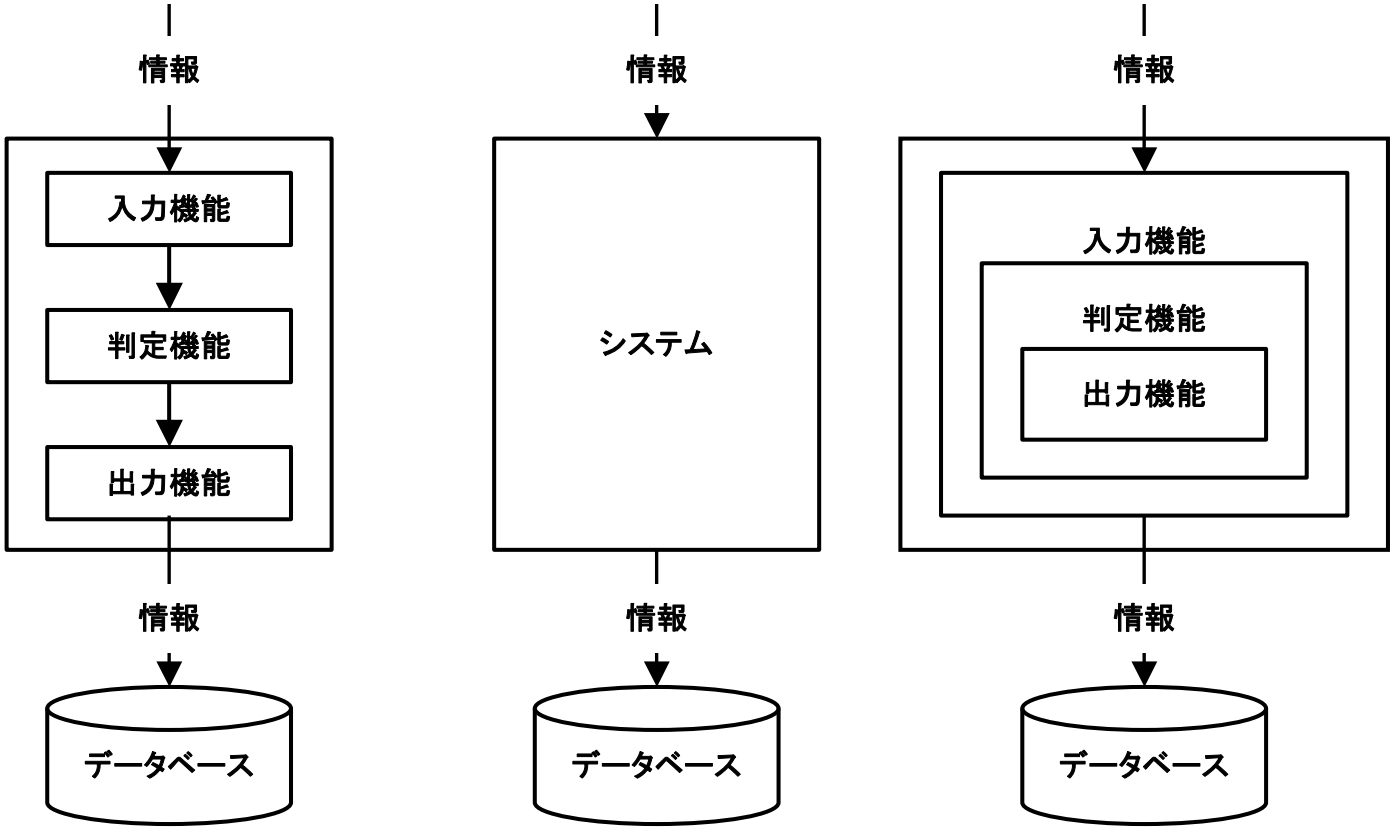


1つの仕様，意図した設計とコード

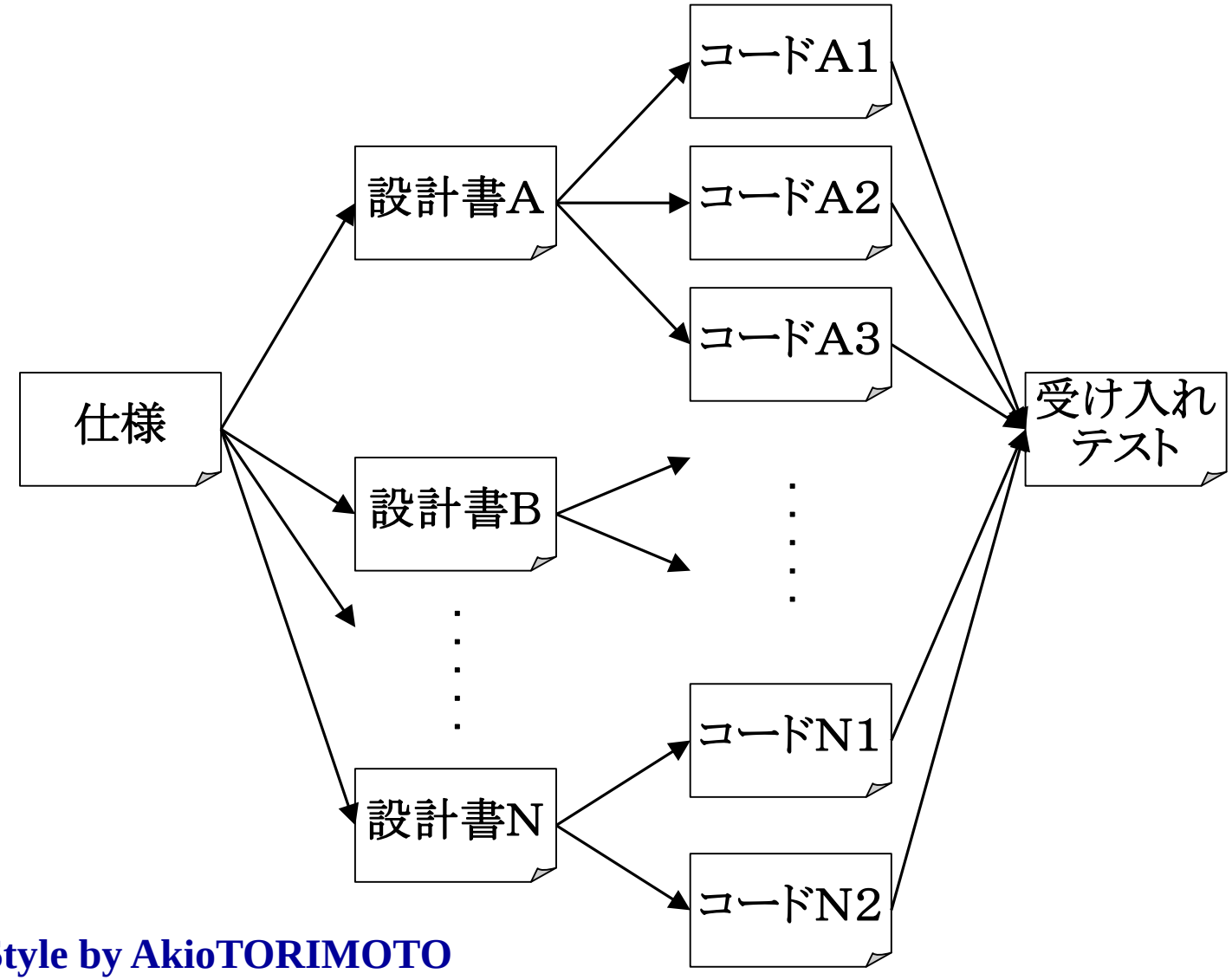


1つの仕様，意図と異なる設計

[仕様]
システムが情報（整数の値）を受け取る入力機能と
情報が整数か否かを判定する判定機能とデータベースに情報を出力する出力機能を
持つソフトウェアの概念設計



1つの仕様，異なる設計とコード



仕様から意図した設計とコードへ

○仕様書と設計書の間の相違を繰り返しチェック

○設計書の誤りを発見，修正

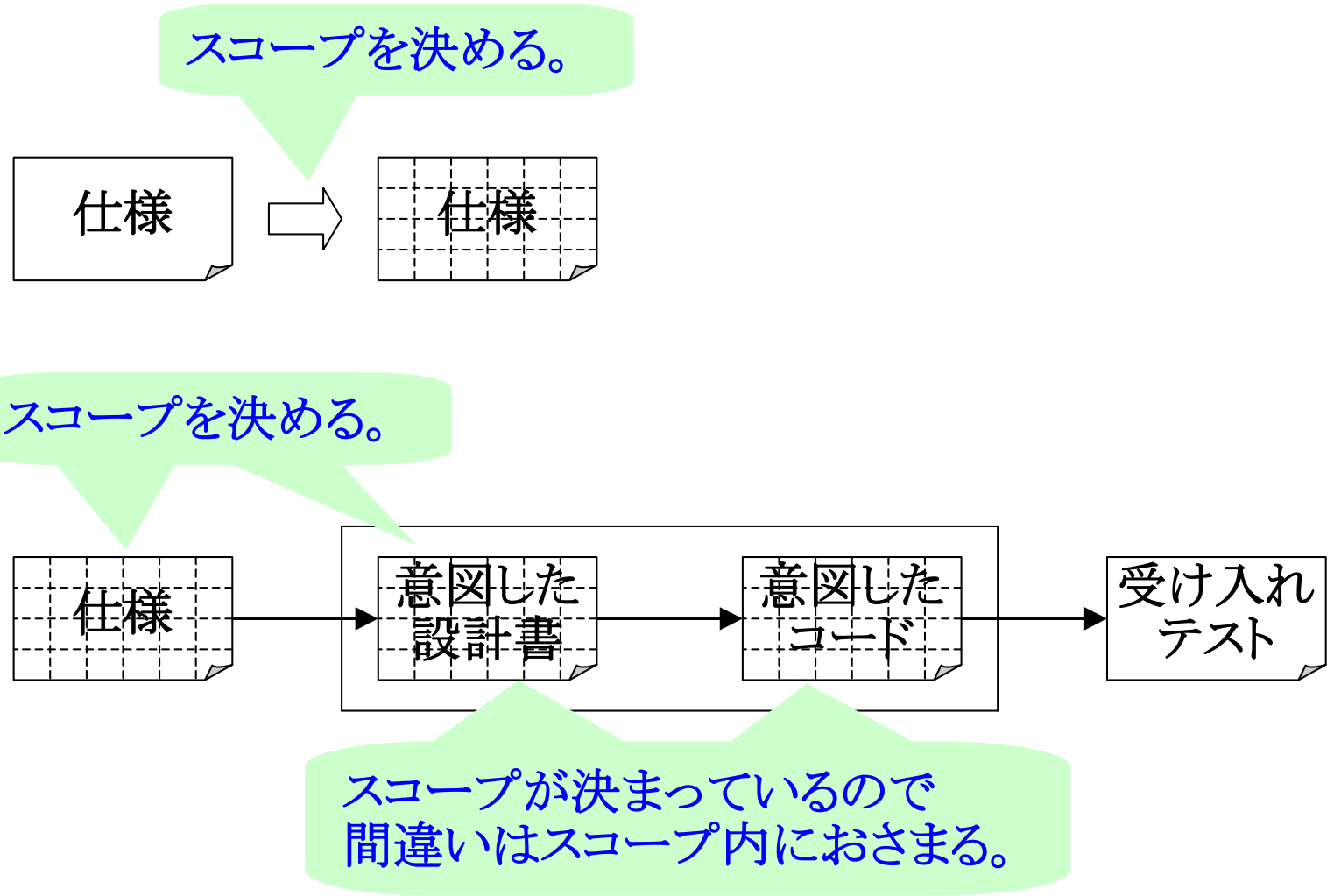
○設計書とコードの間の相違を繰り返しチェック

○コードの誤りを発見，修正

仕様から意図した設計書とコードを作るために

機能ブロックと擬似コードを用いる。

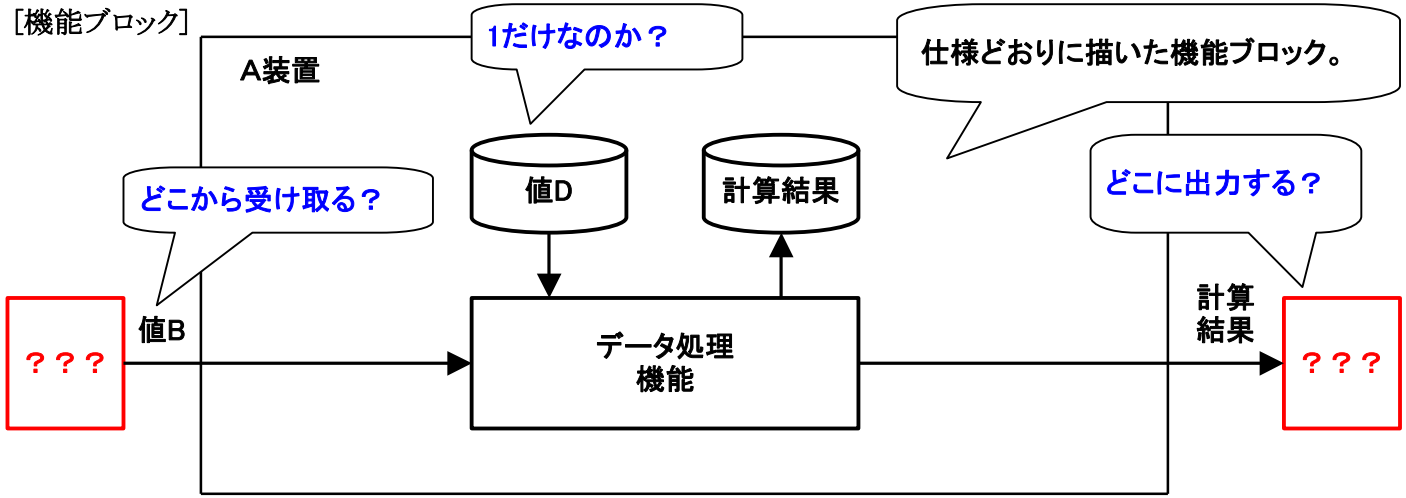
スコープを決め，意図したものに



機能ブロックでレビュー－仕様書－

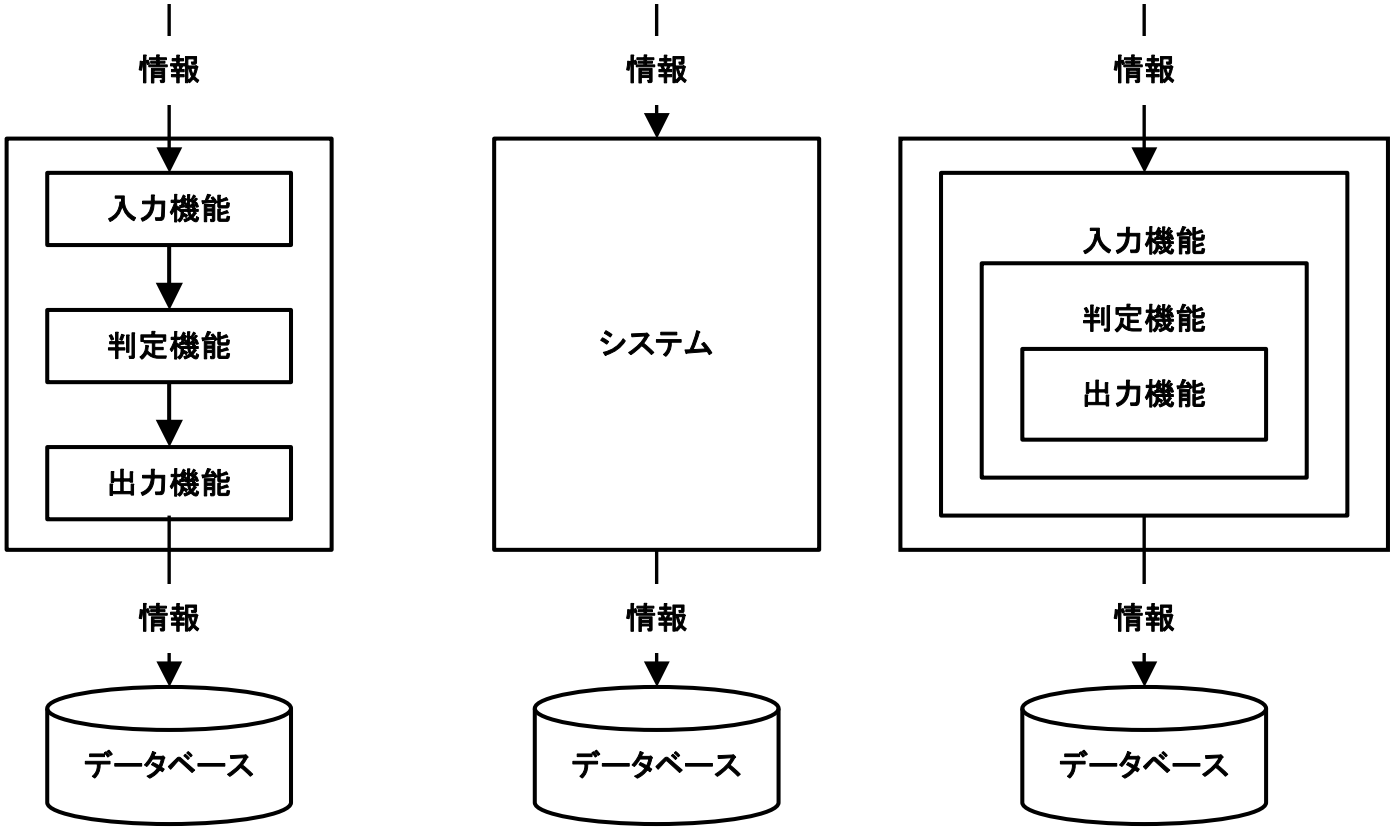
［仕様］
A装置は，値Bを受け取る。
ファイルに定義されている
値Dが1のとき，受け取った値Bを2倍して，出力する。
計算結果は保存する。
ファイルはA装置起動中に変更されない。

［機能ブロック］



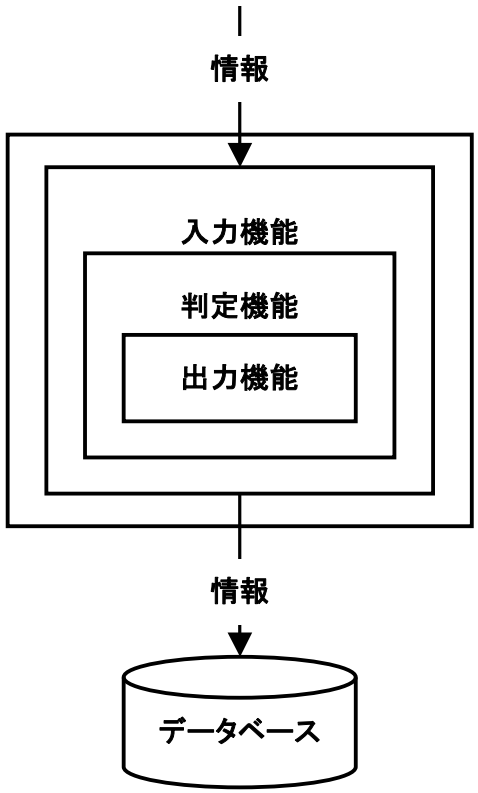
機能ブロックでレビュー－設計書－

[仕様]
システムが情報（整数の値）を受け取る入力機能と
情報が整数か否かを判定する判定機能とデータベースに情報を出力する出力機能を
持つソフトウェアの概念設計



機能ブロックによるコードレビュー

```
int    main( void )
{
    if ( 入力機能( 情報を入力する領域 ) == 入力成功 ) {
        if ( 判定機能( 情報 ) == 判定結果は整数 ) {
            出力機能( 情報 );
        }
    }
    return 0;
}
```



擬似コードによるレビュー

```
int    main( void )
    if ( 入力機能( 情報を入力する領域 ) == 入力失敗 ) {
        return 0;
    }
    if ( 判定機能( 情報 ) == 判定結果は整数でない ) {
        return 0;
    }
    出力機能( 情報 );
    return 0;
}
```

擬似コード

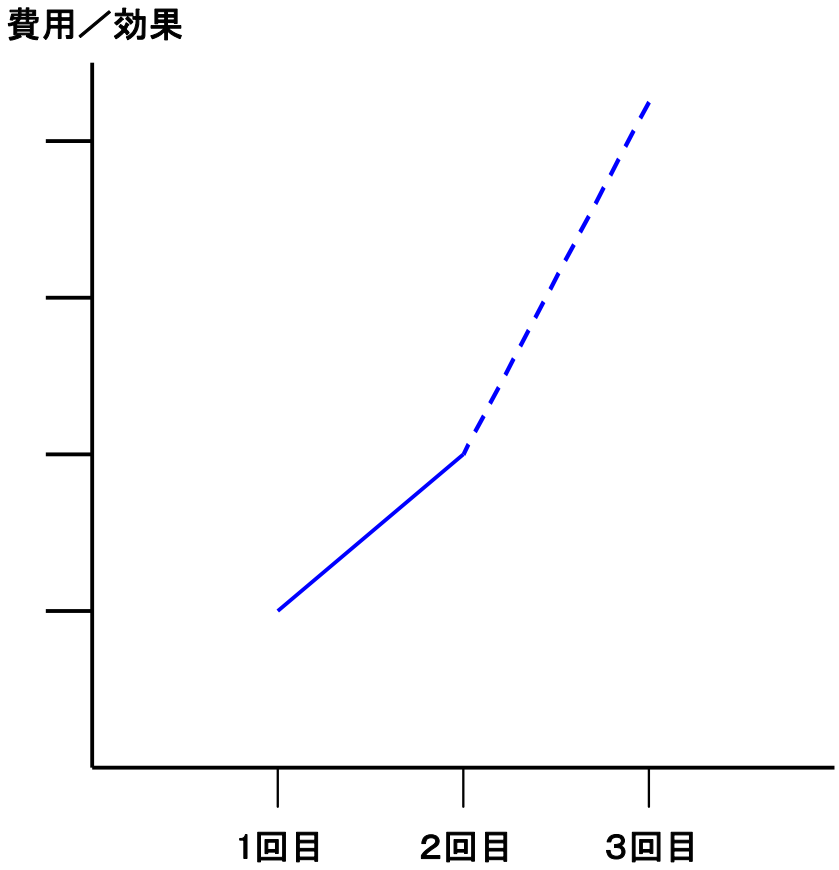
```
int    main( void )
    if ( 入力機能( 情報を入力する領域 ) == 入力成功 ) {
        if ( 判定機能( 情報 ) == 判定結果は整数 ) {
            出力機能( 情報 );
        }
    }
    return 0;
}
```

擬似コード

施策の効果

～2つのツールを用いた効果～

施策の効果 –費用と効果–



施策の効果 –流用できるものが増えた–

○**ドキュメントの流用**がし易くなった。

○**ソフトウェア資産の流用**がし易くなった。

○**ノウハウの蓄積**が増えた。

機能ブロックと擬似コードを用いることで
設計書とコードが整理されたものに。

施策の効果 –2つのツールのレビューの効果–

○**レビューが設計**となる。

○レビュー結果を上流工程に伝えることで
上流工程で機能ブロックを用いるようになる。
上流工程で描かれた機能ブロックが
設計書の一部になる。

○**レビューが楽しく**なる。

○**レビュー結果を楽しみに**してもらえるようになる。

○**レビュー結果をみるのが楽しく**なる。

この先は

機能ブロックのこの先(1)

○**コードを分析する**のに，機能ブロックを使用。
コードに直接，矩形を描き
データの流れを矢印で描くことで，
コードの全体を理解する。

○3つの図形以外に**好きな図形を気ままに追加**。
自分だけのドキュメントは，
自分だけが一番理解し易い機能ブロックで。

○**思うがままの粒度**で本質を見抜くように。

機能ブロックのこの先(2)

○機能ブロックでコードレビューを行っていると
テストし難いコードに似たような特徴があった。
（**アンチパターン**）

○アンチパターンがあるなら，
理想となるパターンがあるのでは？
（**機能ブロックによるあるべきパターン**）

ご清聴感謝致します。