機能ブロックと擬似コードを用いたS/W開発事例

～上流工程で後工程の作業に直結したアウトラインを描く～

発表者　　：鳥本　明男

メルコ・パワー・システムズ株式会社

目次

1 はじめに

機能ブロックと擬似コードを用いたS/W開発事例を発表します。

本開発事例は，
　上流工程で後工程の作業に直結したアウトラインを
　機能ブロック，擬似コード，テストデータで描きます。
要件定義，外部設計，内部設計の工程でテストを考慮します。
S/Wの製作担当が，コーディングを行うまでに，テストし易い設計を行うということです。

２　解決したい問題

○[品質の向上]と[納期短縮]を図りたい。

○上流工程で後工程で発生しそうな問題を取り除きたい。

○上流工程と後工程の要求のトレーサビリティを確保したい。

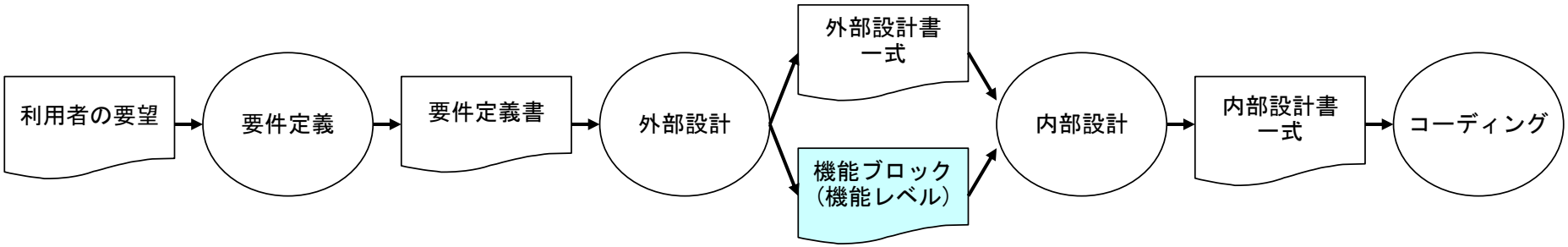
○要件定義，外部設計，内部設計，コーディングの各工程を
　複数の担当者で効率的に進めたい。

○ユニットテストフレームワーク(xUnit)を用いたユニットテストの定着と
　範囲拡大を行い，業務毎の特殊性にとらわれないコードを増やしたい。
　また，効率的にユニットテストを行いたい。

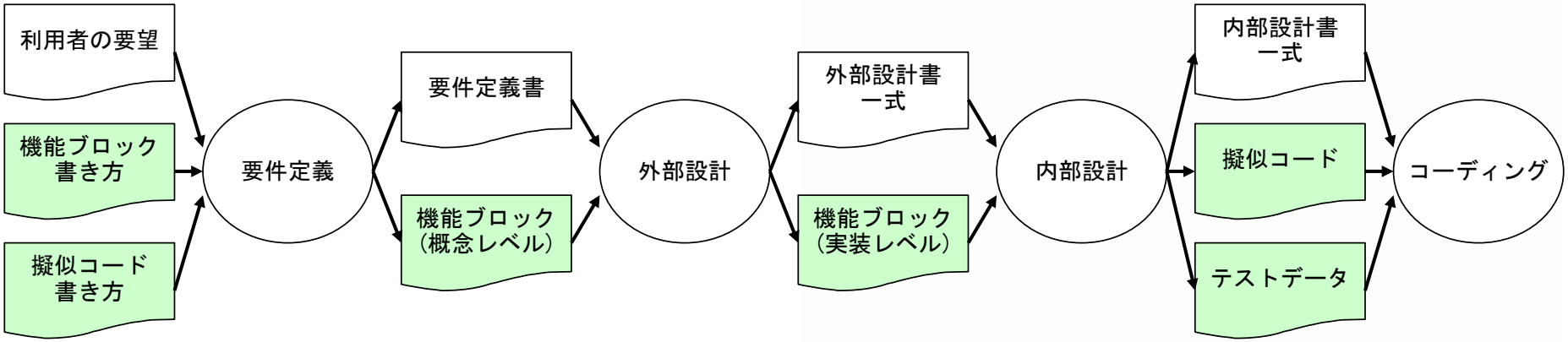
○開発に，シンプルで統一的な考え方やツールを用いたい。

３　問題に対する施策　～施策前と施策後の作業のイメージ～

施策前の作業のイメージ



施策後の作業のイメージ



4 施策の効果 ～問題に対する効果～

機能ブロック(概念レベル，実装レベル)，擬似コード，テストデータで
スコープを決めることで，次の効果が得られる。

○スコープが正しければ不具合はスコープ内で起こる。
不具合の発生箇所が局所的(スコープ内)になる。
スコープを上流工程で決めることで，
上流工程から品質の作り込みが行える。

○上流工程と後工程の要求の**トレーサビリティを確保**できる。

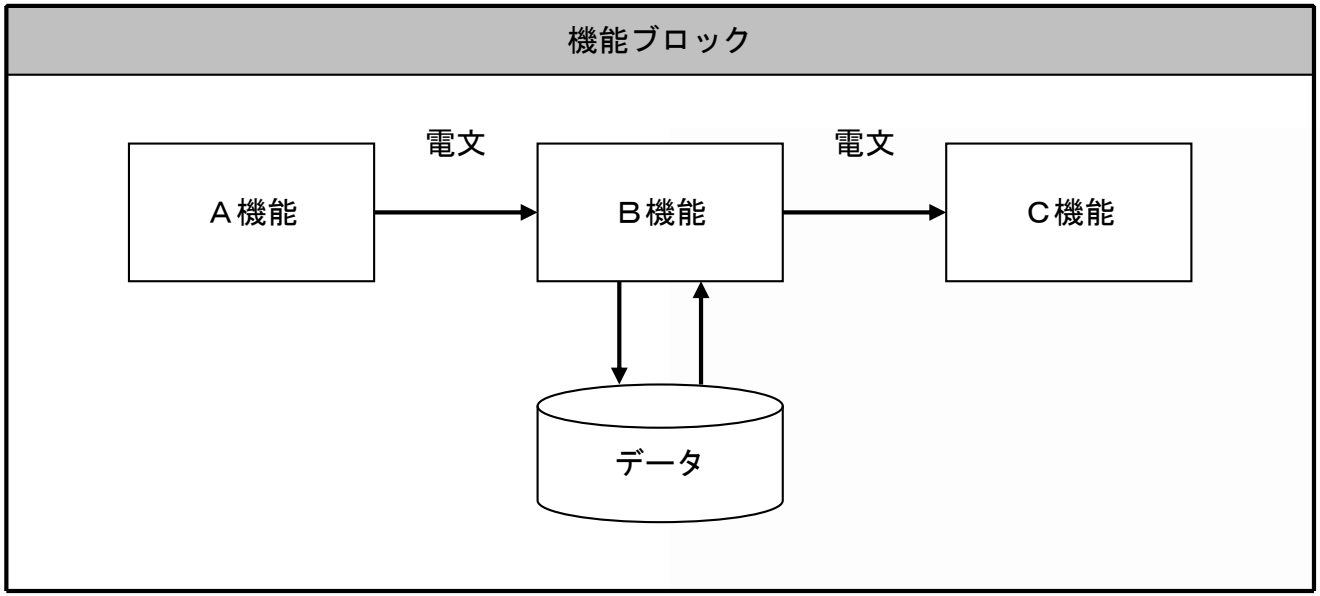
○**機能の依存関係が明確**になり複数の担当者での効率的な作業が行える

○**開発言語にのみに依存するコード**を増やすことができ，
効率的なユニットテストに繋がる。**(5-7参照)**
このようなコードが業務毎の特殊性を取り除けたコードに繋がる。

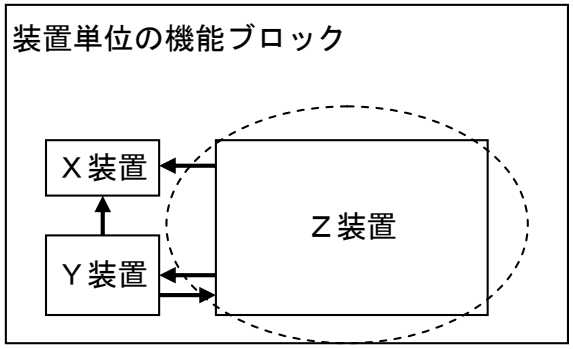
5　機能ブロックの具体例　～機能ブロックとは～

機能間の関係を，
　矩形(機能)同士を，矢印(データの流れ)で結び，整理したもの。

構成部品
　矩形(機能)，矢印(データの流れ)，コメント，データ

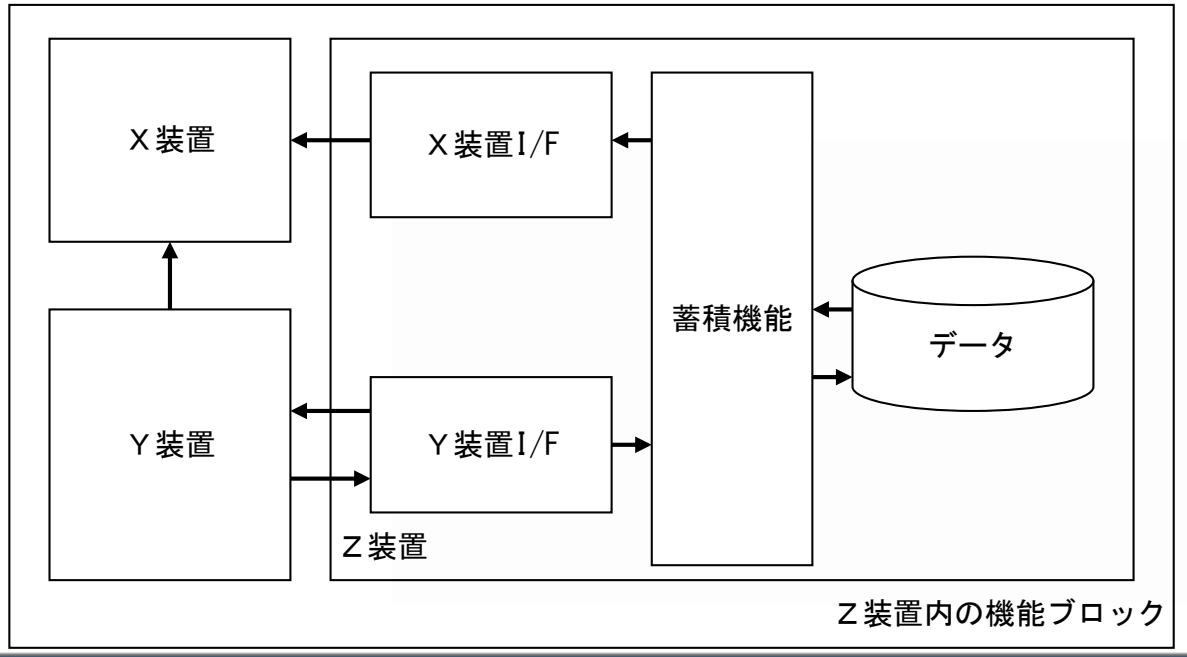


5 機能ブロックの具体例 ～描き方のポイント①～



装置単位の機能ブロックから，
装置内の機能ブロックへ。

Ｚ装置内の機能ブロックを描く。



5　機能ブロックの具体例　～描き方のポイント②～

後工程の作業に直結したアウトラインを示す機能ブロックを描く。

○関連する全ての仕様書を基に，機能ブロックを描く。

○装置を1つの機能と見做し，装置単位の機能ブロックを描く。

○装置単位の機能ブロックのデータの流れを踏襲し，
　装置内の機能ブロックを描く。

○依存するものにより機能を分ける。
　依存するものとしては，
　　データ，
　　動作するタイミング，
　　実行環境
　などがある。

５　機能ブロックの具体例　～描き方のポイント③～

上流工程から品質を作り込むために，機能ブロックを用いて，
S/W設計・製作に直結する**仕様書のDR**を行い，後工程に繋げる。

ポイントは以下。

〇データの管理者とデータの流れに着目して描く。

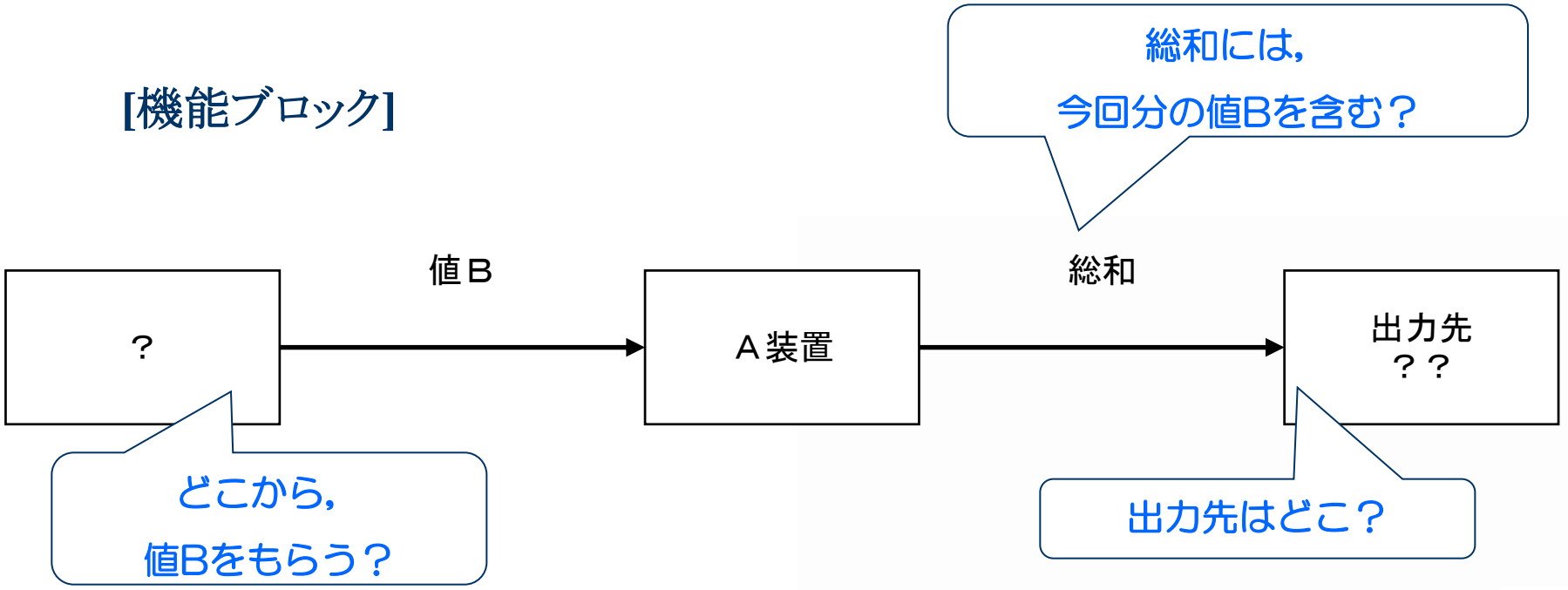
〇仕様書に記載されている文章を，そのまま描く。
　不明な点は，不明なままに描く。

〇不明な点を必要に応じて補う。補ったものがわかるようにしておく。

5　機能ブロックの具体例　～描き方のポイント④～

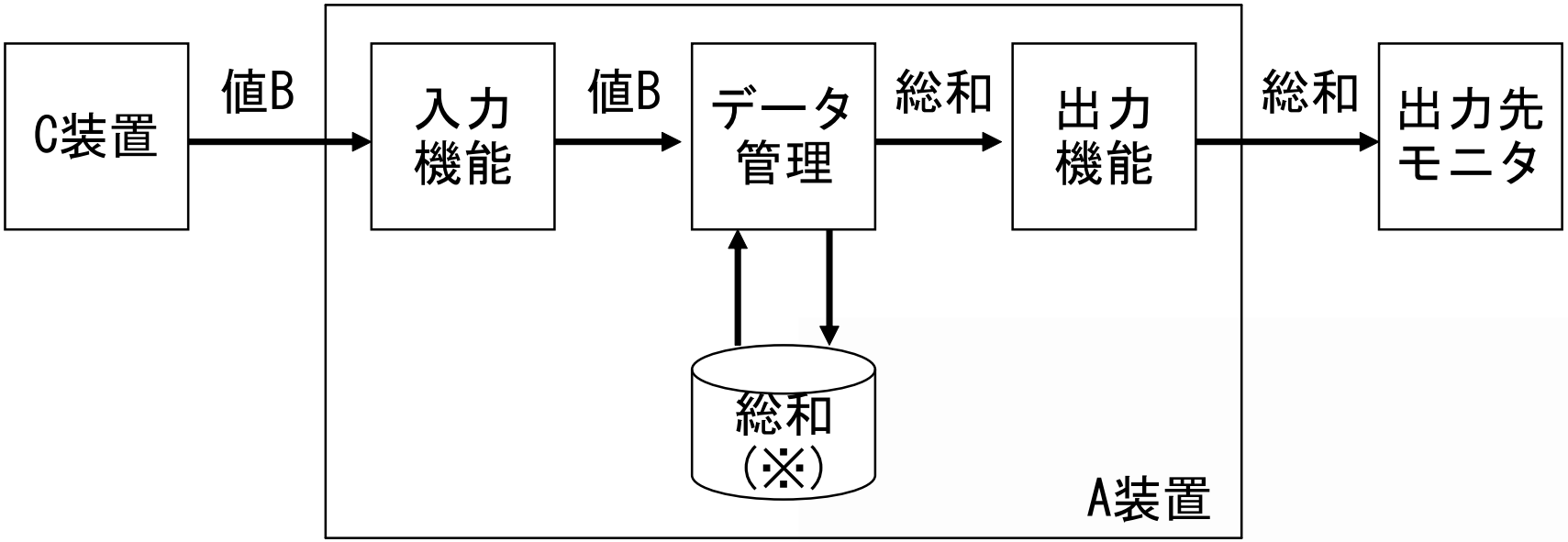
[要件定義]
A装置は，値Bを受け取り保存。
過去に保存した値Bとの総和を出力。

[機能ブロック]



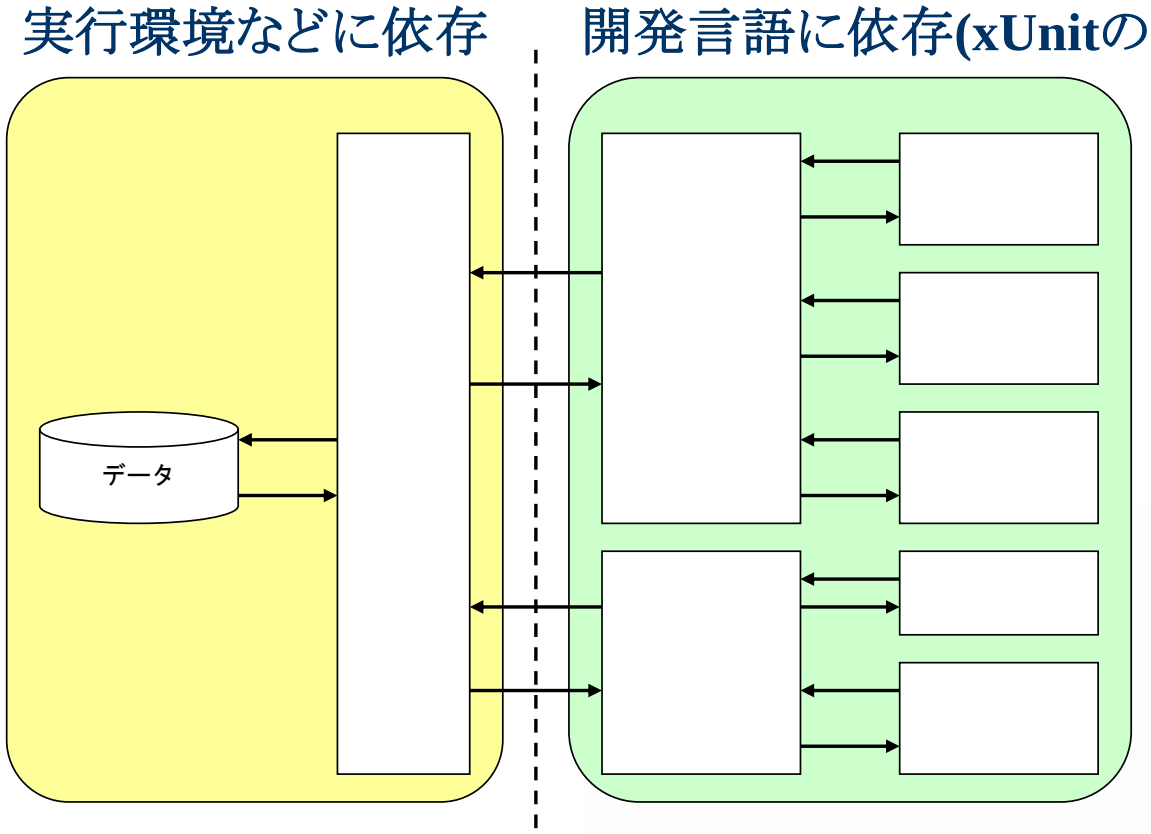
5　機能ブロックの具体例　～描き方のポイント⑤～

後工程の作業に直結したアウトラインを示す機能ブロックを描く。



※総和は，受け取った値Bと過去の総和を加算したもの

５　機能ブロックの具体例　～依存性とタイミングの考慮～



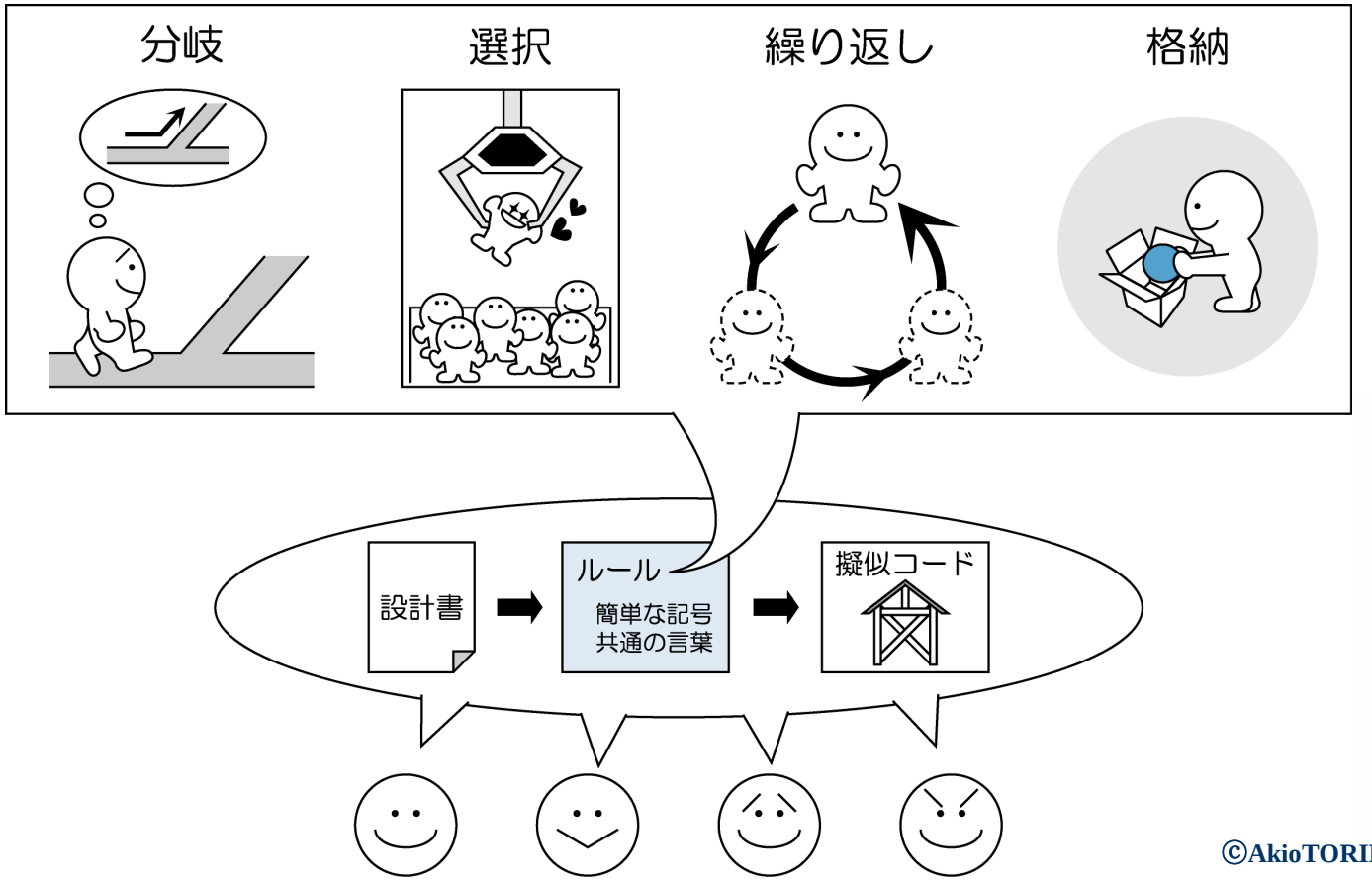
タイミングは実行環境に依存している側のコードで考慮する。

6　擬似コードの具体例　～擬似コード①～

```
メイン処理（　）　{
　総和　＝　０；
　入力機能（　受け取った値Bを受け取るエリア　）；
　データ管理（　値B,　　値Bを含む前の総和（＝総和）,　　値Bを含んだ総和を返すエリア（＝総和）　　）；
　出力機能（　総和　）；
　先の入力機能に戻る。
}

データ管理（　値B,　　値Bを含む前の総和,　　値Bを含んだ総和を返すエリア　）　{
　値Bを含んだ総和を返すエリア　＝　値Bを含む前の総和　＋　値B；
}

入力機能（　受け取った値Bを受け取るエリア　）　{
　ここで，C装置からの値Bの受け取りを待つ。
　受け取った値Bを受け取るエリア　＝　受け取った値B。
}

出力機能（　総和　）　{
　モニタに，総和を出力する。
}
```

6 擬似コードの具体例 ～擬似コード②～

簡単に作れる「**コードのような文章**」で**骨組を先に製作**し，レビューすることで，
短い時間で誤りを取り除き，スコープを決めることで誤り箇所を局所的にする。



６　擬似コードの具体例　～描き方のポイント～

○言語仕様のうちの**分岐**，**選択**，**繰り返し**，**格納**と**コメント**で描く。

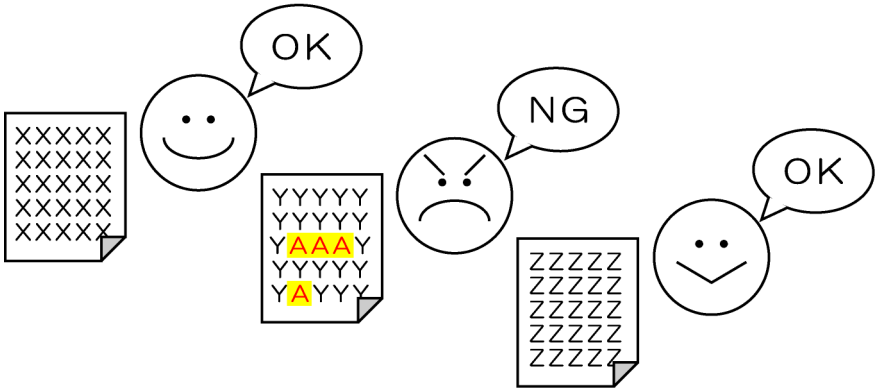
○**スコープ**を意識し，**I/Fとデータの管理**を明確にする。

○開発で用いる言葉を，開発メンバ内で**共通の言葉**とし，これを用いる。

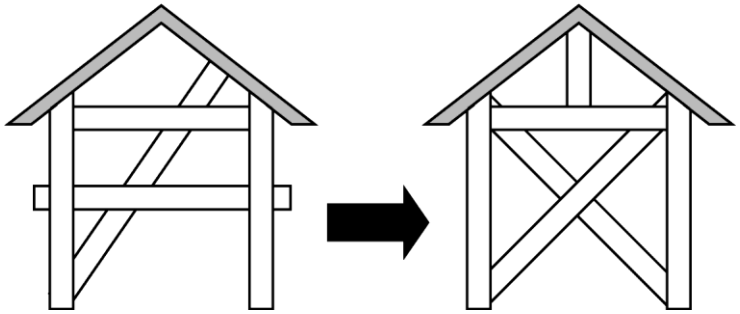
○**擬似コードを置きかえる**，あるいは，コメント文にすることで，
コーディングが行えるように描く。

6 擬似コードの具体例 ～事前のコードレビュー～

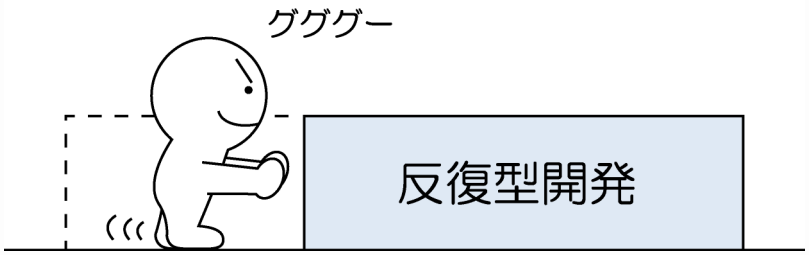
レビューが簡単



変更が容易



反復型開発の後押し



7　テストデータの具体例　～上流工程からの客観性～

○上流工程の担当者が擬似コードを描いた後，テストデータを作成する。
　テストデータは，**Excel**等を用いて，**csv**形式で作成し，
　1行で1つのテストとする。

○テストデータの作成と並行して，
　コーディングを行う担当者は，コーディングを行う。

○コーディング作業と並行して，
　テストコードを製作する担当者は，テストコードを製作する。

○コーディング，テストコード，テストデータを別々の担当者で製作する。

○テストデータを上流工程の担当者が作成することにより，
　上流工程からの客観性をもつことができる。

8 開発事例 ～要件定義～

[要件定義]

ポジションという情報とその状態（1か，0）を管理する。
複数のポジションの状態で，状態が決まるポジションを集約ポジションと呼ぶ。
1つのポジションの状態で，状態が決まるポジションを個別ポジションと呼ぶ。
個別ポジションは，集約ポジションに影響を与えるものがある。
入力は，個別ポジションと状態の組。
出力は，個別ポジションと状態と状態変化の組と集約ポジションと状態と状態変化の組。
この機能を，ポジション状態集約機能，と呼ぶ。

個別ポジションと集約ポジションと状態の関係の例は次のとおり。

個別ポジションと集約ポジションの関係

個別ポジション	影響を与えるポジション
A	D
B	D
C	集約対象外

集約ポジション
D

8 開発事例 ～要件定義の整理～

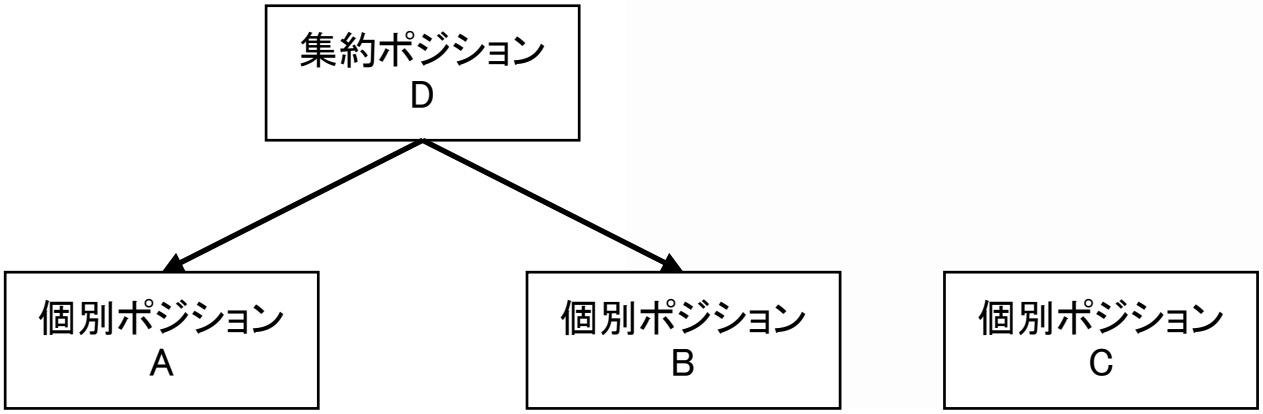
[状態の関係]

個別ポジションAの状態は，1かO。
個別ポジションBの状態は，1かO。
集約ポジションDの状態は，個別ポジションA，Bがともに，Oのとき，O。それ以外は，1。

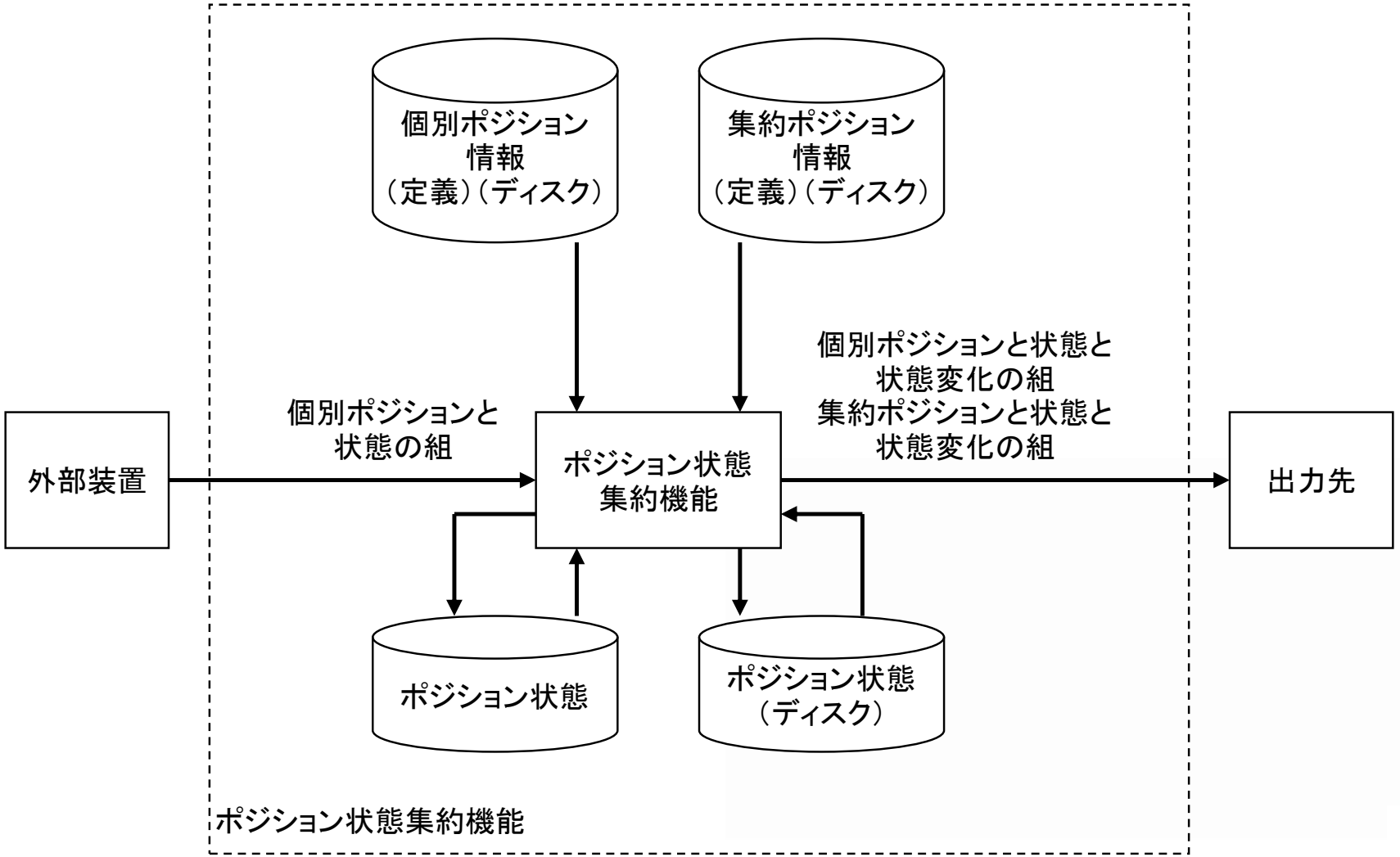
個別ポジションA，Bの状態と集約ポジションDの状態の関係

集約ポジションDの状態	個別ポジションAの状態	個別ポジションBの状態
1	1	1
1	1	0
1	0	1
0	0	0

[個別ポジションと集約ポジションの関係の概念図]



8 開発事例 ～良くみかける設計(イメージ)～



8 開発事例 ～施策を施す際の設計のポイント～

○動作に必要なもので機能を独立させる。(**依存性のスコープ**)
(外部装置，出力先とのI/Fなど)

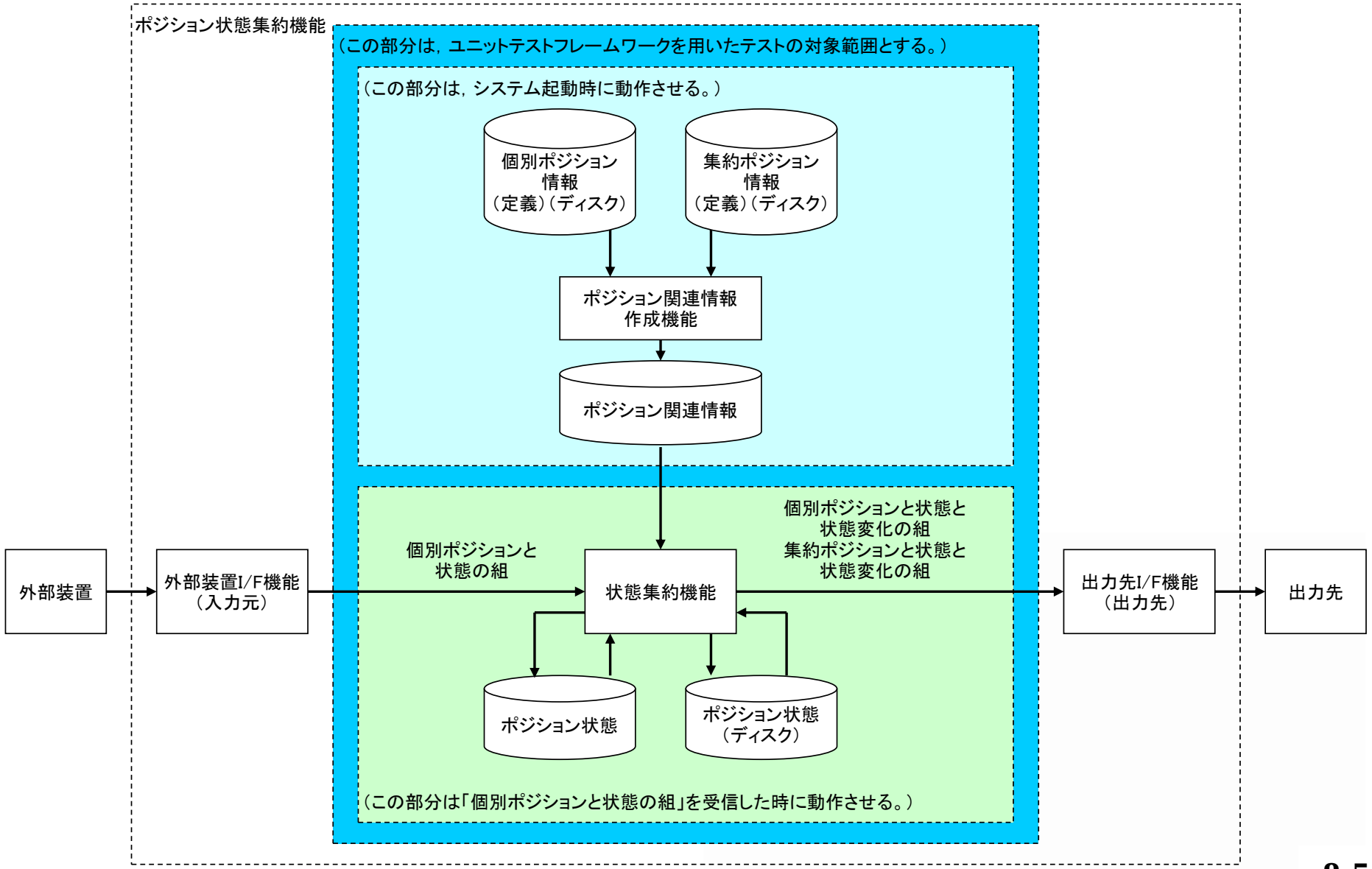
○ユニットテストフレームワークを用いたテストの対象範囲を明確にする。
(**テストのスコープ**)

○機能が動作するタイミングを明確にする。(**時間のスコープ**)
(システムが起動した際，処理が完結するものとしないものを分ける。

○機能の関係を明確にする。(**機能のスコープ**)

○データの関係，データの管理者を明確にする。(**データのスコープ**)

次ページに，施策を施した設計(イメージ)を示す。



8 開発事例 ～生産性の実績(定量的なデータ)～

従来の開発手法での見積もり工数	6weeks
実績工数	4weeks
個別ポジションの数	300点
集約ポジションの数	50点
擬似コードのライン数	約1kライン
コードのライン数（コメント含む）	約6kライン
単体テストのチェックポイント数	約68,000点
開発メンバ	4名
システムテスト時のバグ数	0件

8 開発事例 ～[品質の向上]と[納期短縮]に繋がった理由～

○作業を行う際の確認作業が最小で済む。**行うべき作業で悩まない。**

○データの管理者(機能)を明確にし，データの競合が起こらなかったこと。
テストし易いデータ構造となったこと。

○開発言語にのみ依存するコードが増え，**テストし易いS/Wの構成となり，テストが行い易いユニットテストの項目数が増えた**こと。

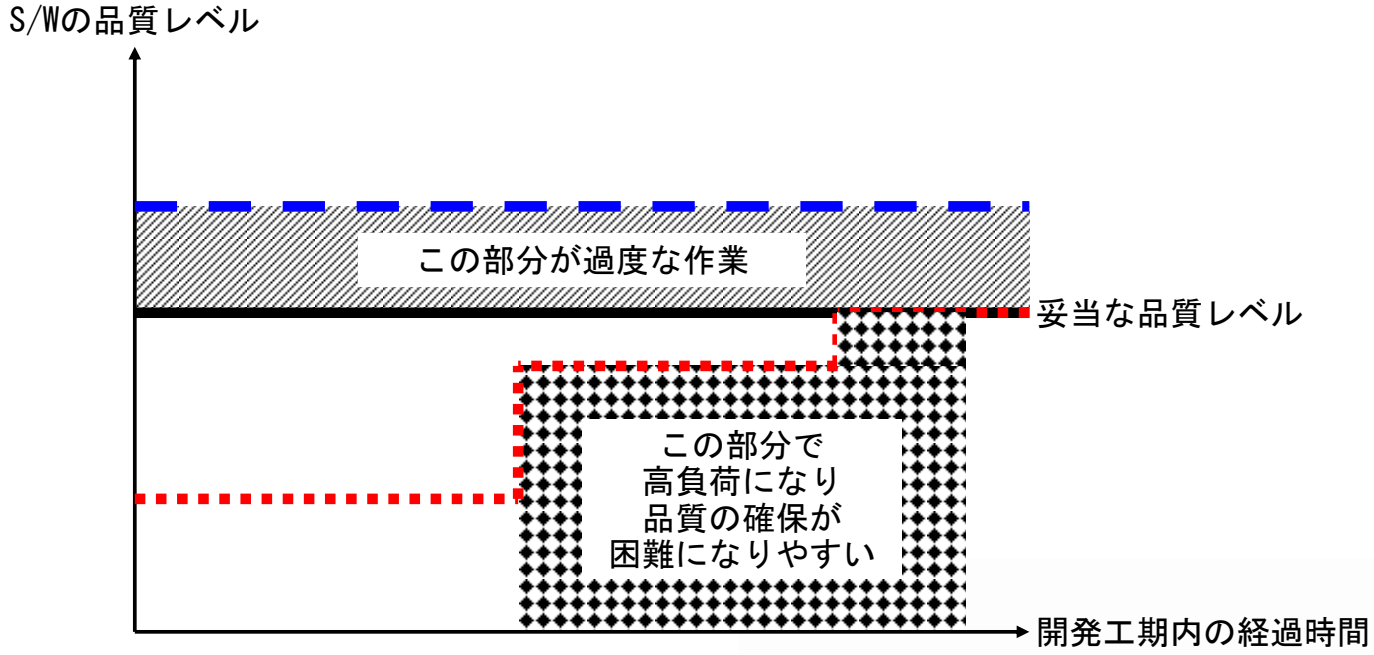
○開発言語にのみ依存するコードが増え，個々人の開発PCにて
テストが行えたこと。(実機の使用スケジュールの影響が受けにくくなる。)

○**ユニットテストでのバグが少なかった**こと。

○**品質のレベルが妥当**であったこと。
過度な作業を行わない。作業に漏れがない。手戻りが少ない。
工期の後半に品質不足による補てんのための作業が不要となったこと。

次ページに，納期短縮に繋がった，品質のレベル(イメージ)を示す。

8 開発事例 ～品質のレベル(イメージ)～

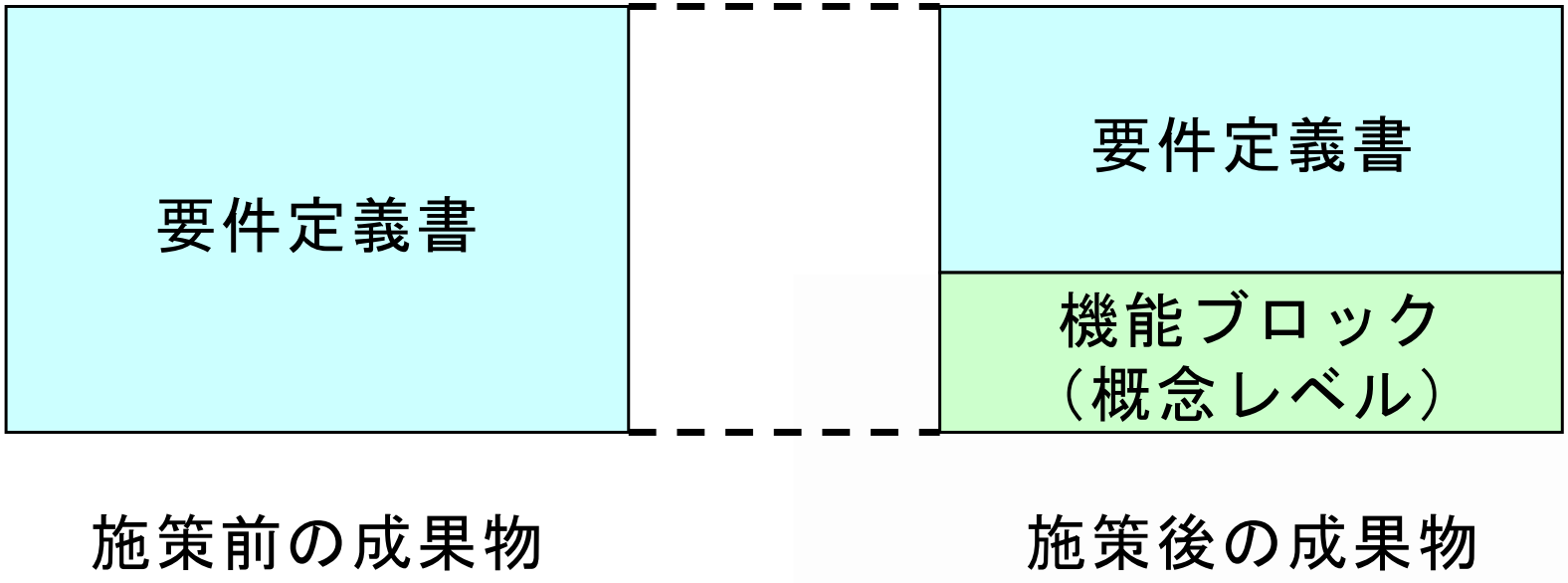


品質が十分でない機能の品質のレベルの推移
（開発工期内の後半で一気に妥当な品質レベルまで品質を上げる）

品質が過剰な機能の品質のレベルの推移
（開発工期内の後半でもこの品質を維持し続けなければならない。
維持し続けないと動作しない個所が存在し易くなる。
工数が増えないように過剰な箇所を取り除くことを目指すことになる。）

8 開発事例 ～施策前後の成果物のイメージ～

施策前後で行うべき作業は同じ。
施策後は,
行うべき作業の一つ(機能ブロック(概念レベル))が明確になる。

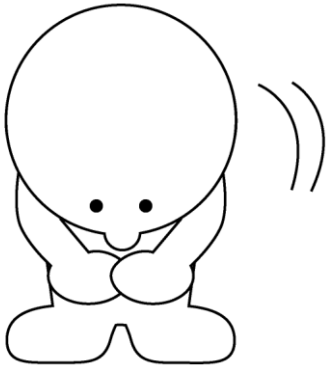


施策前後の成果物のイメージ

９　さいごに　～今後の展開・課題～

機能ブロック，擬似コードはシンプルな考え方であるため，
行っていることは理解し易い反面，抽象度が高く，
機能ブロック，擬似コードを描く担当者の実力が如実に表れる。

機能ブロック，擬似コードを正しく描くことができる担当者の育成を，
　実践方法の講習会の開催，
　個々人で独自に学習できる課題の配布と課題の添削，
で行っている。



ご清聴ありがとうございました

本発表についての問い合わせ先

メルコ・パワー・システムズ株式会社
技術統括部　ビジネスチーム6　鳥本
Torimoto.Akio@ze.MitsubishiElectric.co.jp