JaSST'10 東京

# テスト初心者　セッション

*～テス活はじめてみよう～*

北都システム(株)　長谷川 聡
宮崎大学　片山 徹郎

# 目次

1. ソフトウェアテストとは？
2. テスト設計とは？
3. テスト設計技法
   - コードカバレッジ(制御フローテスト)
   - 同値分割法
   - 境界値分析
   - **All-Pair法**
4. さいごに

# 1. ソフトウェアテストとは？

# ソフトウェアテストとは？　1/2

## 先人たちの言葉：

・「テストとは、エラーを見つけるつもりでプログラムを実行する過程である。」
***G.J.Myers***

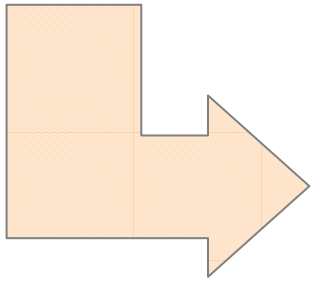
・「テストとは、プログラムまたはシステムの属性を評価することを目的とするあらゆる活動である。」 ***B.Hetzel***

・「テストとは、プログラムを、既知の環境下で選ばれた入力により実行した結果に基づいて、その動作特性を推論する過程である。」 ***J.B.Goodenough***

・「テストでプログラム中のバグの存在は示せても、バグが存在しないということは示しえない。」 ***E.W.Dijkstra***

・「バグを全部見つけるのは無理だと心得よ。」 ***Cem Kaner***

などなど



・エラーを見つける
・属性・特性を評価する
・あらゆる活動
・全部のバグは見つからない？

# ソフトウェアテストとは？　2/2

テストとは:

成果物が定義した要件を満足するかを判定し、目的に合致することを実証し、欠陥を見つけるため、ソフトウェア製品や関連成果物に対し、計画、準備、評価をすること。

全てのライフサイクルを通じて実施する静的、動的なプロセス。

※JSTQB ソフトウェアテスト標準用語集（日本語版）

テストの目的:

1. 欠陥を摘出する。
2. 対象ソフトウェアの品質レベルが十分であることを確認し、その情報を示す。
3. 欠陥の作りこみを防ぐ。

※JSTQBテスト技術者資格制度 Foundation Levelシラバス

# テストの一般原則

原則１：テストは欠陥があることしか示せない

原則２：全数テストは不可能

原則３：初期テスト

原則４：欠陥の偏在

原則５：殺虫剤のパラドックス

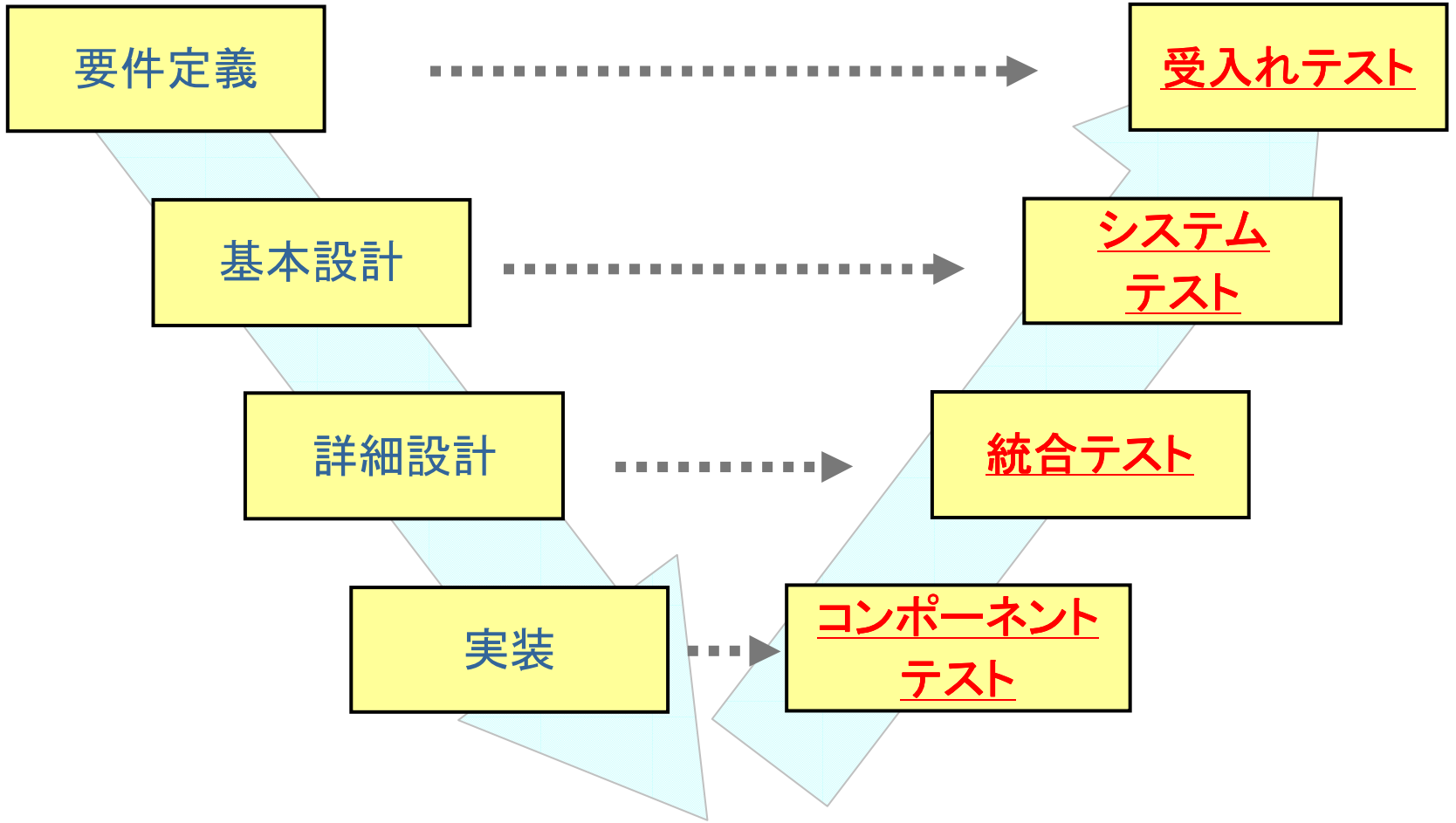
原則６：テストは条件次第

原則７：「バグゼロ」の落とし穴

※JSTQBテスト技術者資格制度 Foundation Levelシラバス

# テストレベル

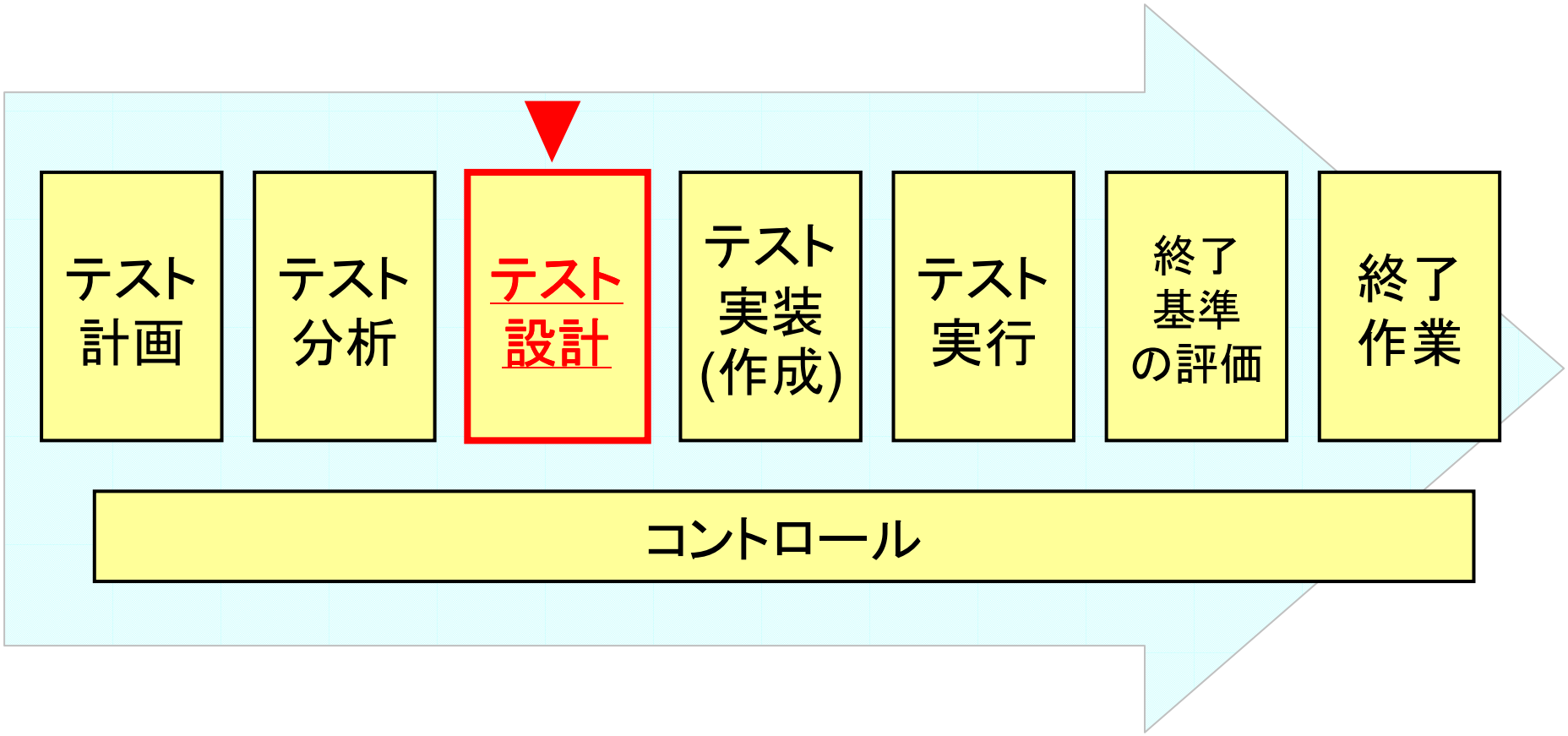
テストには色々なレベルがある。



# 2. テスト設計とは？

# テストのライフサイクル

## 基本的なテストプロセス



# テスト設計の必要性 1/2

## なぜ(ちゃんと)テスト設計しなければならないのか？

- **全部テストすることができないため**

  →量な問題(限られた時間、テストケース数の爆発)、技術的な問題(そもそも「全部」がわからない)

  問題が見つかる可能性の高いテストケースをあげたい。(テストケース自体の質)

- **漏れ、重複を防ぐため**

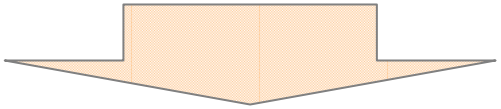
  →テストしたいことが確実にあげられているか？(漏れ)
  また、二つのテストをしても同じバグしか見つからないことはないか？(重複)

- **変更に正しく対応するため**

  →ソフトウェアの変更があった場合にテストへの影響度が容易にわからない。

- **よりよいテスト(改善)につなげるため**

  →なぜそのテストを行ったのか？明確になっていないと分析/改善が困難。



# テスト設計の必要性 2/2

## なぜ(ちゃんと)テスト設計しなければならないのか？

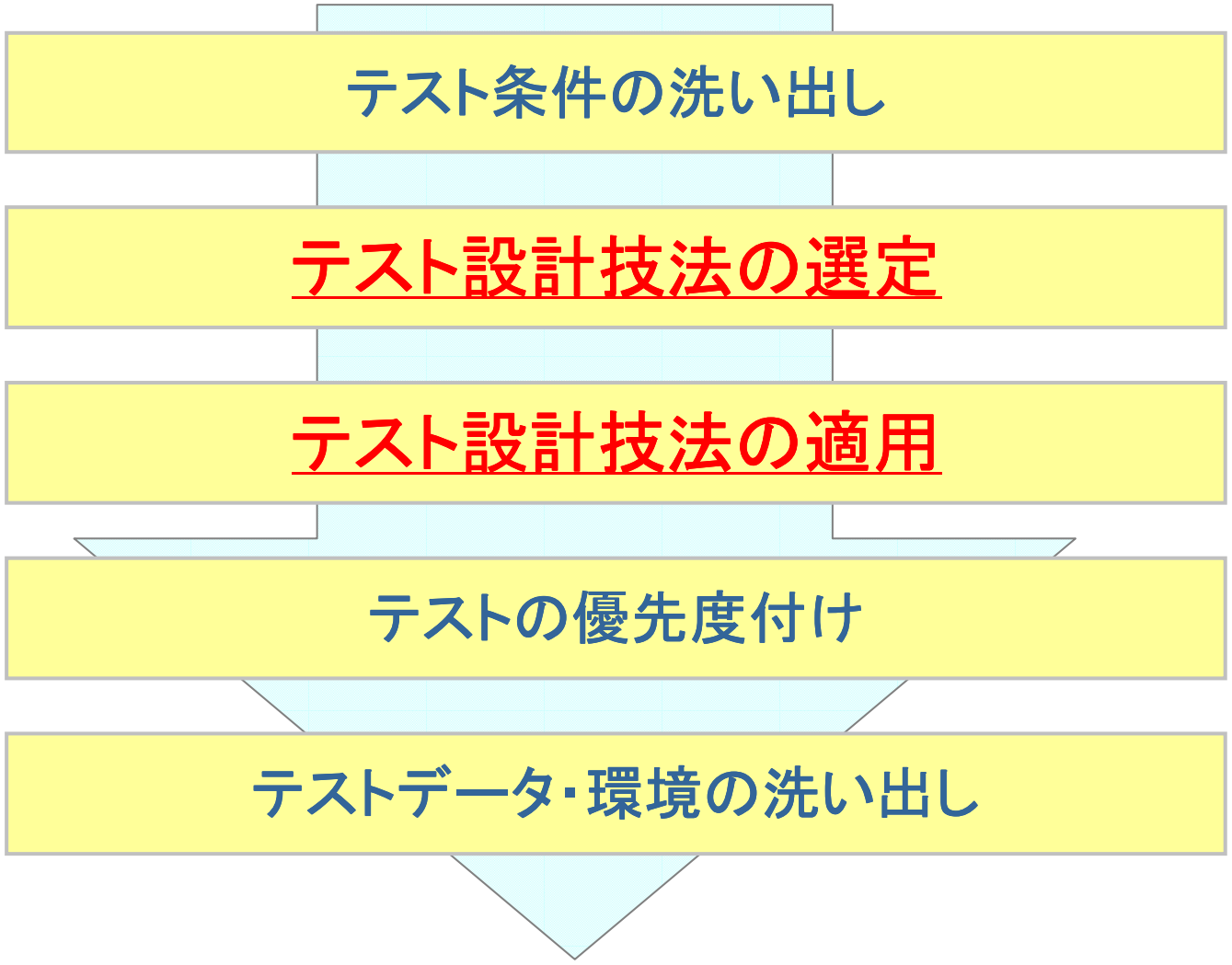
それは、

- より少ないテストケースで、
- カバレッジ(網羅)を確保しつつ、
- より多くのバグを摘出し、
- 整理された(説明できる、変更に対応可能)

テストケースを作成するため！！

# テスト設計では

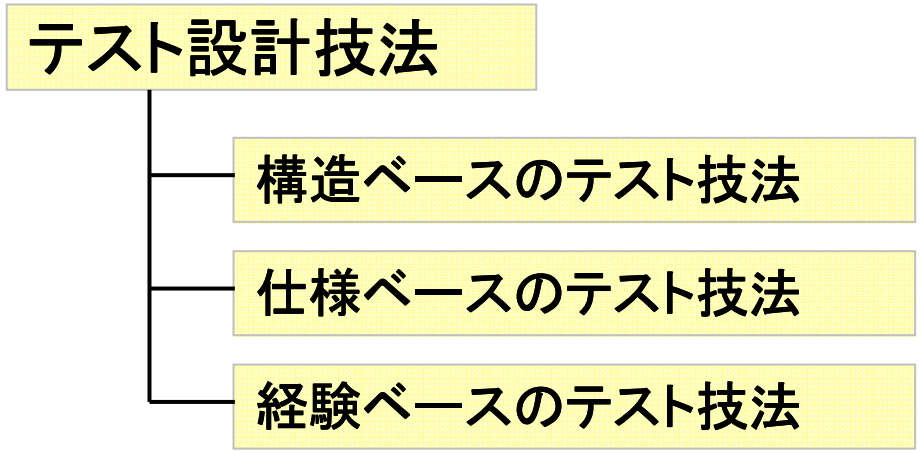
テスト設計にて行うこと



# 3. テスト設計技法

- コードカバレッジ(制御フローテスト)
- 同値分割法
- 境界値分析
- All-Pair法

# テスト設計技法のカテゴリ　1/3




## 構造ベースのテスト技法

コンポーネントやシステムの内部構造を分析して、テストを設計する技法。
ホワイトボックス技法とも呼ばれる。

主なテスト技法)
コードカバレッジ(制御フロー)、データフローテスト、コールフローテスト
など

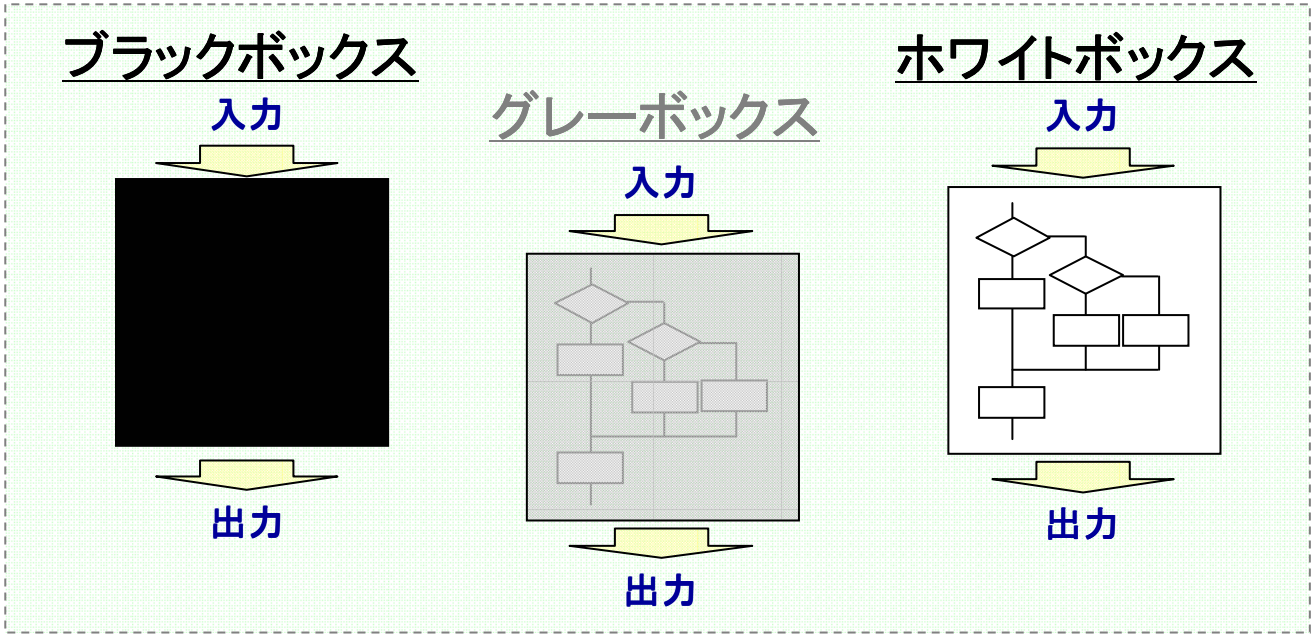
# テスト設計技法のカテゴリ　2/3

## 仕様ベースのテスト技法

内部構造は参照せず、ドキュメントをベースにして、テスト条件やテストケースを導き出し、選択する技法。ブラックボックス技法とも呼ばれる。

主なテスト技法)

同値分割法、境界値分析、デシジョンテーブル、状態遷移テスト、ユースケーステスト、All-Pair法、HAYST法、原因結果グラフ　など



# テスト設計技法のカテゴリ　3/3

## 経験ベースのテスト技法

テスト担当者のスキルと直感の他、同様のアプリケーションまたはテクノロジーにおける経験からテストを抽出する。

経験ベースの技法は、テスト担当者の経験に大きく依存する。

主なテスト技法)

エラー推測、探索的テスト　　など

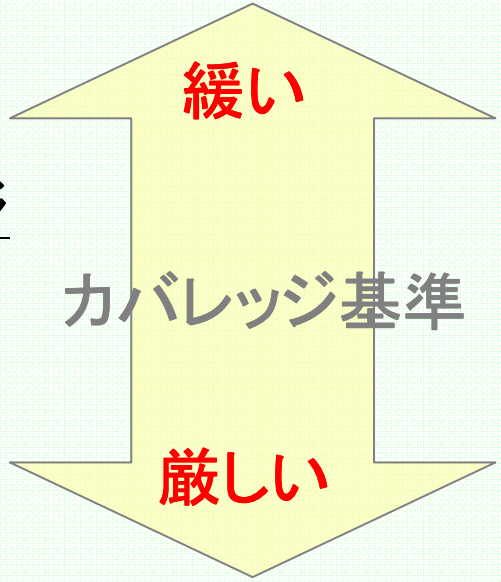
# コードカバレッジ　1/5　～ カバレッジ基準の種類 ～

コードカバレッジは、構造ベースのテスト技法でありソースコードを対象にどれくらい実行されたかを評価する技法。

カバレッジする対象基準によっていくつかの種類がある。

## カバレッジの種類

- ステートメントカバレッジ
- デシジョン(ブランチ)カバレッジ
- 条件 カバレッジ
- 複合条件 カバレッジ

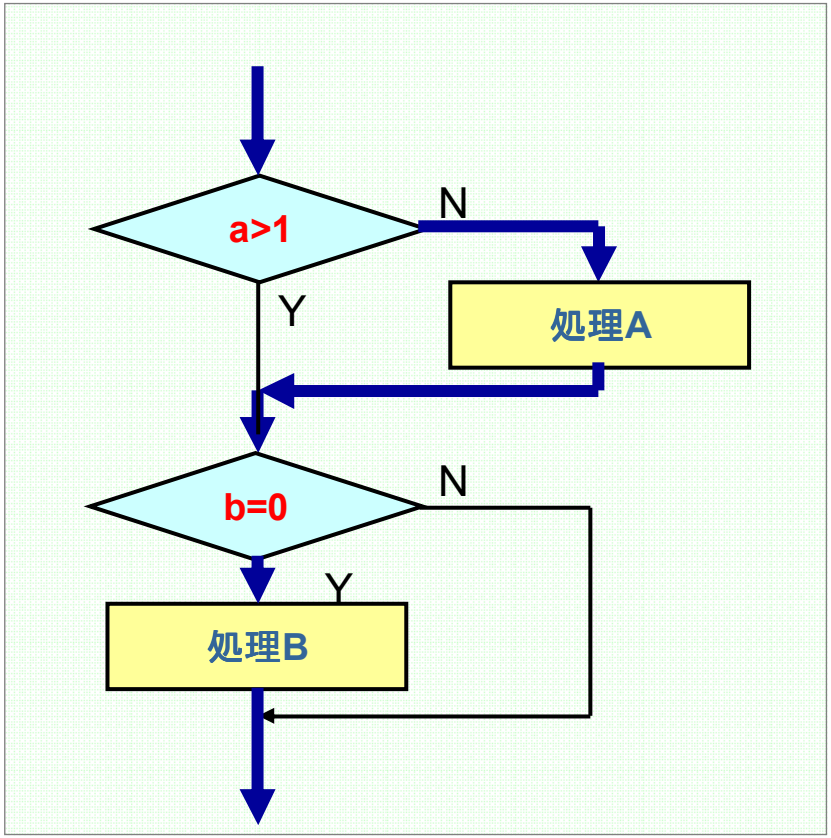


# コードカバレッジ　2/5　～ ステートメントカバレッジ ～

## ステートメントカバレッジ

テスト対象のコード中のステートメント(命令文)がどれくらい実行されたかを確認する。

「C0」などと呼ばれている。

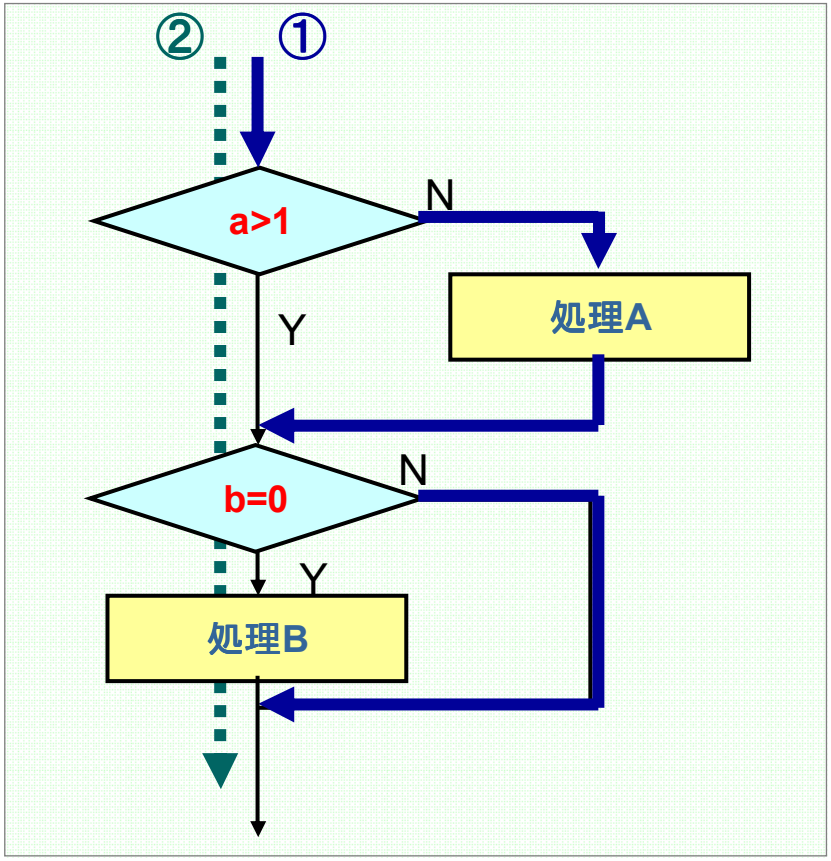



この処理に対して、例えばa=0、b=0を与えることで100%のステートメントカバレッジとなる。

# コードカバレッジ　3/5　～ デシジョンカバレッジ ～

## デシジョンカバレッジ

テスト対象のコード中の**判定の結果**がどれくらい実行されたかを確認する。
「C1」などと呼ばれている。



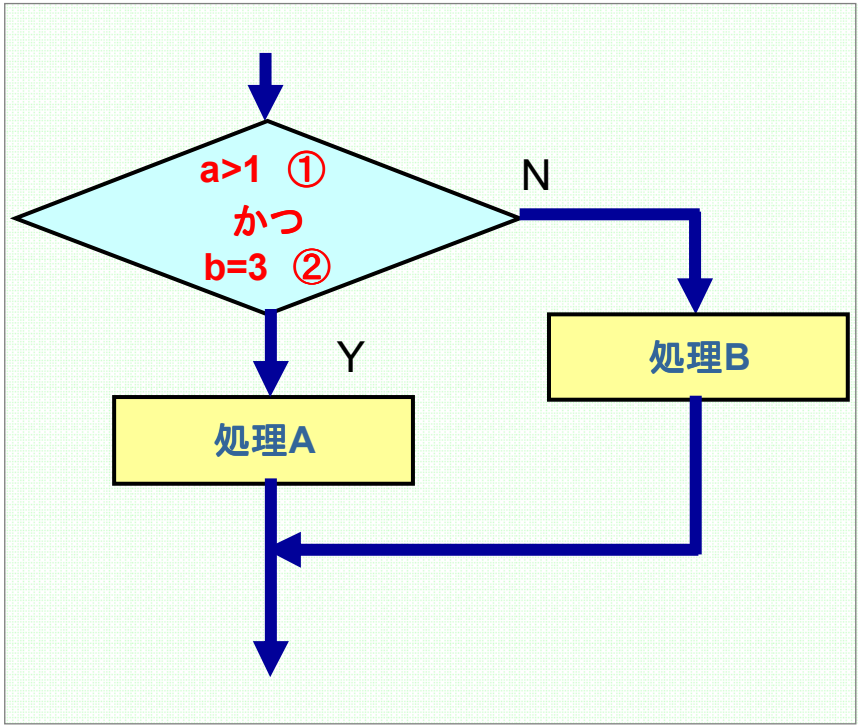

この処理に対して、例えば
①(a=0、b=1)
②(a=3、b=0)
を与えることで100%のデシジョンカバレッジとなる。

# コードカバレッジ　4/5　～ 条件カバレッジ ～

## 条件カバレッジ

テスト対象のコード中の個々の判定(条件)の結果がどれくらい実行されたかを確認する。

※複数の条件がANDやORで結合されている場合




この場合、
「a>1」①、「b=3」②のそれぞれの条件に対して、
- Y(真)となる場合
- N(偽)となる場合

を評価するため、例えば
- a=2、b=3
- a=0、b=0

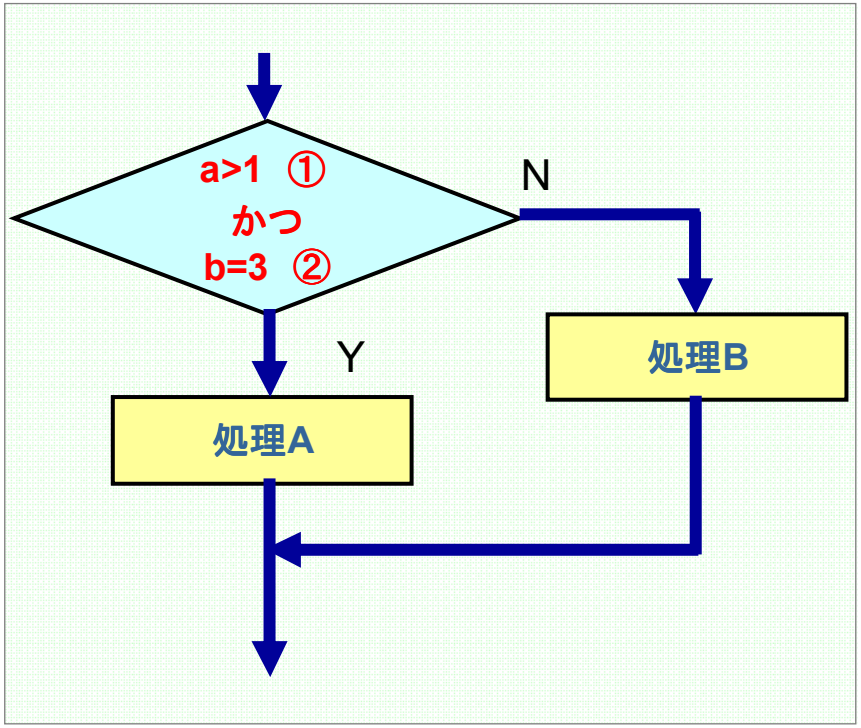
を与えることで100%の条件カバレッジとなる。

# コードカバレッジ　5/5　～ 複合条件カバレッジ ～

## 複合条件カバレッジ

条件カバレッジをさらに厳しくした基準であり、コード中の**複数の条件文をどのように判定したか**を確認する。

※複数の条件がANDやORで結合されている場合




この場合、
「a>1」①、「b=3」②のそれぞれの判定結果に対して
- ①:Y、②:Y　で判定結果Y
- ①:Y、②:N　で判定結果N
- ①:N、②:Y　で判定結果N
- ①:N、②:N　で判定結果N

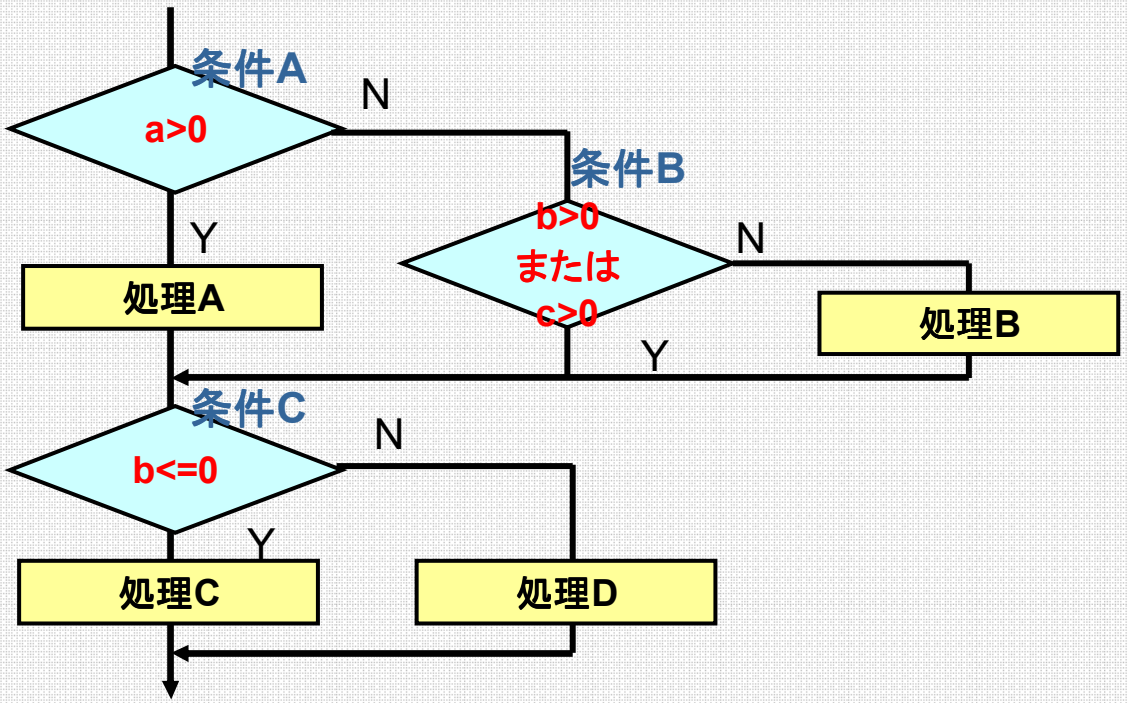
を評価するため、例えば
- a=2、b=3
- a=2、b=0
- a=0、b=3
- a=0、b=0

を与えることで100%の複合条件カバレッジとなる。

# コードカバレッジ　演習

次の処理フローに対して、
- ステートメントカバレッジ
- デシジョンカバレッジ
- 条件カバレッジ

をそれぞれ100%にして、かつ少ない回数で確認可能な入力(a,b,cの組み合わせ)を求めよ。



# コードカバレッジ　演習～回答例～

ステートメントカバレッジの場合：

・a=0、b=0、c=0　・・・①
・a=1、b=1、c=任意　・・・②

の2通り

デシジョンカバレッジの場合：

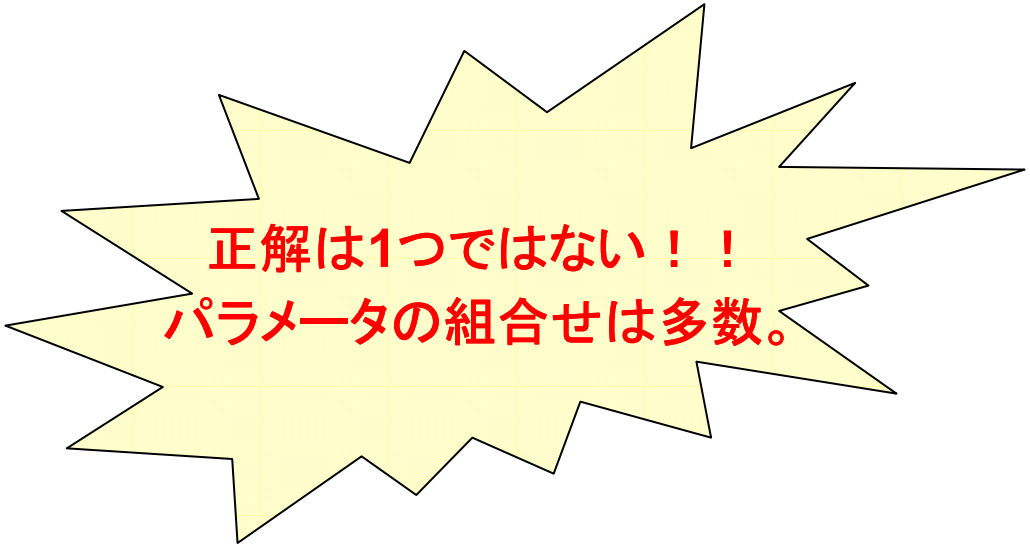
・a=0、b=0、c=0　・・・①
・a=1、b=1、c=任意　・・・②
・a=0、b=1、c=任意　・・・③

の3通り

条件カバレッジの場合：

・a=0、b=0、c=0　・・・①
・a=0、b=1、c=1　・・・②
・a=1、b=1、c=任意　・・・③

の3通り



# コードカバレッジ　演習～解説～

考え方：それぞれ以下のポイントを網羅するよう考える。

ステートメントカバレッジの場合：

| ポイント | チェック |
|---|---|
| 処理A | ・・② |
| 処理B | ・・① |
| 処理C | ・・① |
| 処理D | ・・② |

・a=0、b=0、c=0　・・・①
・a=1、b=1、c=任意　・・・②
の2通り

デシジョンカバレッジの場合：

| ポイント | チェック |
|---|---|
| 条件A(Y) | ・・② |
| 条件A(N) | ・・① |
| 条件B(Y) | ・・③ |
| 条件B(N) | ・・① |
| 条件C(Y) | ・・① |
| 条件C(N) | ・・② |

・a=0、b=0、c=0　・・・①
・a=1、b=1、c=任意　・・・②
・a=0、b=1、c=任意　・・・③
の3通り

条件カバレッジの場合：

| ポイント | チェック |
|---|---|
| 条件A(Y) | ・・③ |
| 条件A(N) | ・・① |
| 条件B(bがY) | ・・② |
| 条件B(bがN) | ・・① |
| 条件B(cがY) | ・・② |
| 条件B(cがN) | ・・① |
| 条件C(Y) | ・・① |
| 条件C(N) | ・・② |

・a=0、b=0、c=0　・・・①
・a=0、b=1、c=1　・・・②
・a=1、b=1、c=任意　・・・③
の3通り

# 同値分割法　1/3

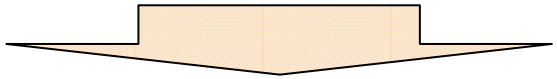
## 同値分割法とは

ソフトウェアやシステムへの入力を同じ処理をするグループに分割し、グループ内の入力を同等に扱えるようにする技法である。

正常データ(有効データ)だけではなく、不正データ(無効データ)も考慮する。

テストでは、同値分割したグループを網羅するように設計する。

同値分割法は、あらゆるレベルのテストで適用できる。



●同じ結果が得られるグループに分割し、代表を選ぶ技法

●少ないテストケースで効率的にテストが可能

**Keyword:**

同値クラス:　同じ処理をするグループ

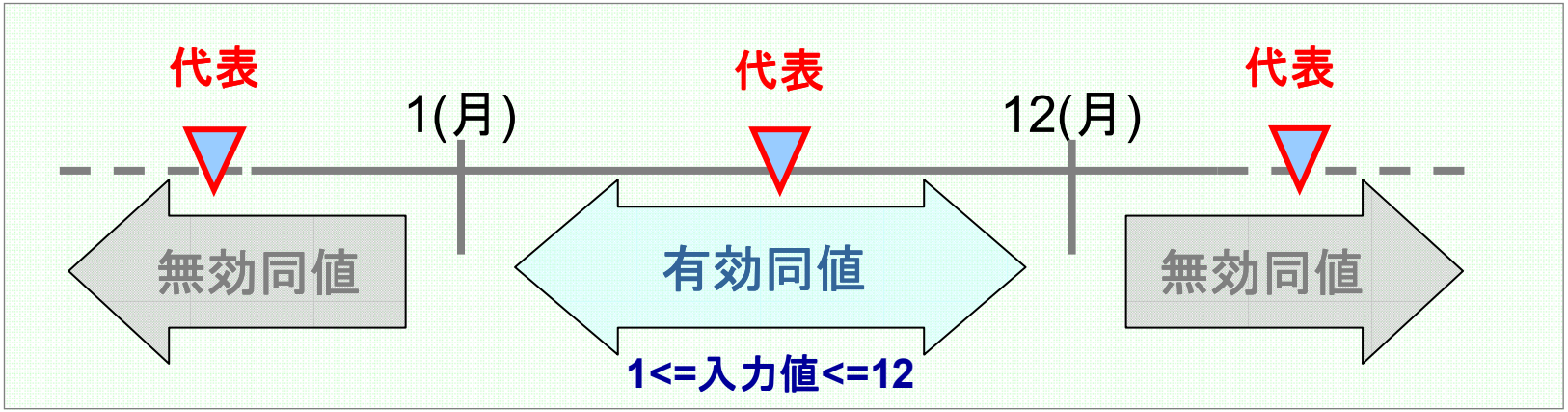
有効同値(クラス):　有効となる場合の同値クラス

無効同値(クラス):　無効となる場合の同値クラス

# 同値分割法　2/3

## 例1)　月の整数入力エリアに関する同値分割

※数値範囲に着目した場合

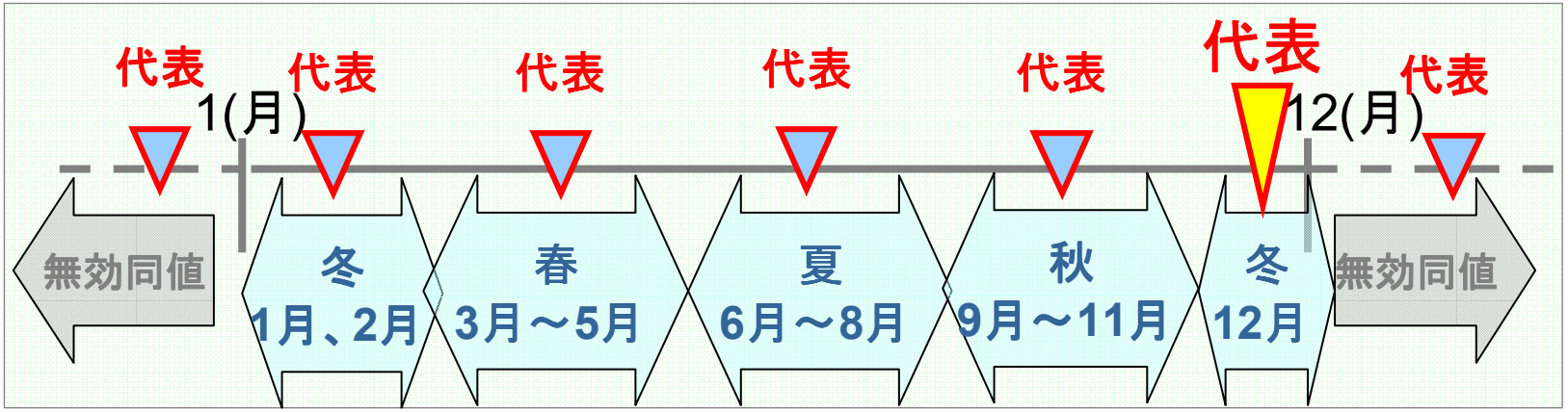



この場合、例えば
「0」、「6」、「20」の入力で同値クラスを網羅できる。

# 同値分割法　3/3

例2)　月の入力に対して季節を判定するソフトウェアの場合

※数値範囲と季節の種類に着目した場合




この場合、例えば
「0」、「1」、「4」、「7」、「10」、「12」、「20」の入力で
同値クラスを網羅できる。

## 同値分割法のポイント

- ◆グループ化(同値分割)する着目点(観点) が重要
- ◆有効同値だけでなく無効同値も考える
- ◆各グループ(同値クラス)から漏れなく代表を選ぶ

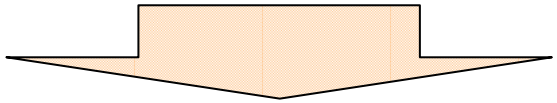
# 境界値分析　1/2

## 境界値分析とは

同値分割したグループの境界上の動作は正しくないことが多く、境界には多くの欠陥が潜んでいる可能性が大きい。

正常な領域の境界値は、正常な境界値となり、不正な領域の境界値は不正な境界値になる。

境界の内側と外側の両方をカバーするようにテストを設計する。



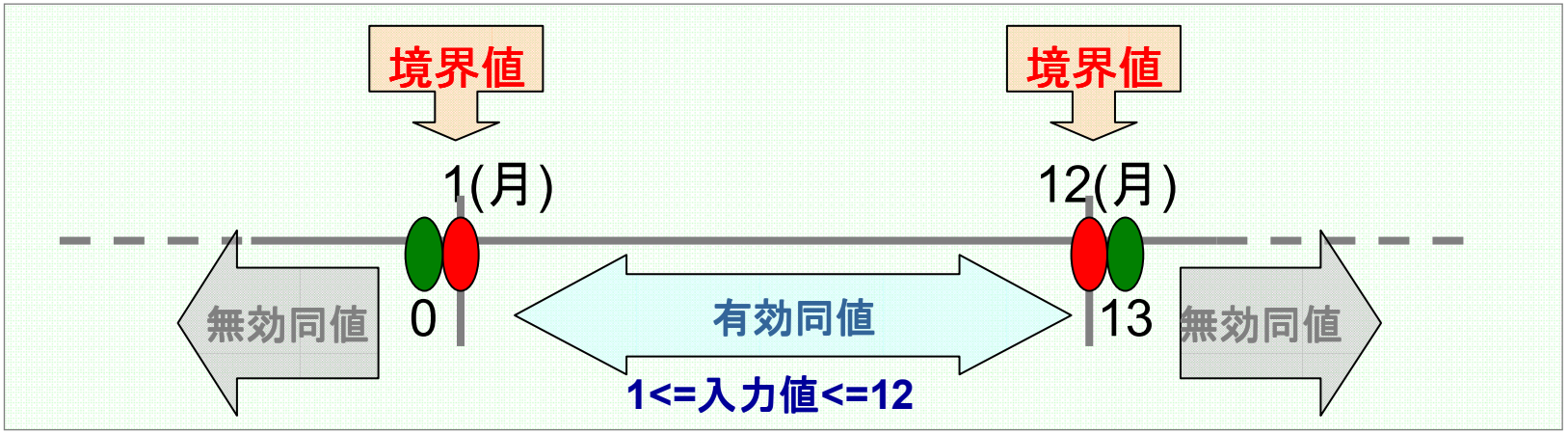
- 同値クラスの境界値に注目してテスト設計する技法
- 境界値付近を狙うことで効果的にテストすることが可能

**Keyword:**

**境界値**:　同値分割した領域の端、あるいは端のどちらか側で最小の増加的距離にある入力値または出力値

# 境界値分析　2/2

例)　月の整数入力エリアに関する境界値分析




この場合、
「0」、「1」、「12」、「13」が境界値分析を利用したテストケースとなる。

## 境界値分析のポイント

- ◆同値分割を利用する
- ◆境界値の範囲に含まれる値、含まれない値をテストする

# 境界値分析 演習1

以下のような荷物配達の料金表があるとする。

この料金判定を行うソフトウェアについて、同値分割、境界値分析を利用してテストケースを作成せよ。

尚、期待する結果も考えること。

| | | |
|---|---|---|
| 定形 | 25gまで | 80円 |
| | 50gまで | 90円 |
| 定形外 | 50gまで | 120円 |
| | 100gまで | 140円 |
| | 500gまで | 390円 |
| | 2kgまで | 850円 |
| | 4kgまで | 1,150円 |

補足：

※入力は整数とし、g単位(少数点は切り上げ)とする。

※マイナス値の入力は考慮しなくてよい。

※50gを超過する定型郵便は定型外として扱われる。

※料金表の範囲を超える場合にはエラーとして処理する。

※形式は定型/定型外以外は考えないでよい(必ずどちらかとなるものとする)。

# 境界値分析 演習1 ～回答例～

| No. | テストケース | 期待結果 | 補足 |
|---|---|---|---|
| 1 | 定型で0g | エラー | |
| 2 | 定型で1g | 80円 | |
| 3 | 定型で25g | 80円 | |
| 4 | 定型で26g | 90円 | |
| 5 | 定型で50g | 90円 | |
| 6 | 定型で51g | 140円 | 定型外扱い |
| 7 | 定型で100g | 140円 | 定型外扱い |
| 8 | 定型で101g | 390円 | 定型外扱い |
| 9 | 定型で500g | 390円 | 定型外扱い |
| 10 | 定型で501g | 850円 | 定型外扱い |
| 11 | 定型で2000g | 850円 | 定型外扱い |
| 12 | 定型で2001g | 1150円 | 定型外扱い |
| 13 | 定型で4000g | 1150円 | 定型外扱い |
| 14 | 定型で4001g | エラー | 定型外扱い |

| | | | |
|---|---|---|---|
| 15 | 定型外で0g | エラー | |
| 16 | 定型外で1g | 120円 | |
| 17 | 定型外で50g | 120円 | |
| 18 | 定型外で51g | 140円 | |
| 19 | 定型外で100g | 140円 | |
| 20 | 定型外で101g | 390円 | |
| 21 | 定型外で500g | 390円 | |
| 22 | 定型外で501g | 850円 | |
| 23 | 定型外で2000g | 850円 | |
| 24 | 定型外で2001g | 1150円 | |
| 25 | 定型外で4000g | 1150円 | |
| 26 | 定型外で4001g | エラー | |

# 境界値分析 演習1 ～解説～

## 同値クラスの着目点

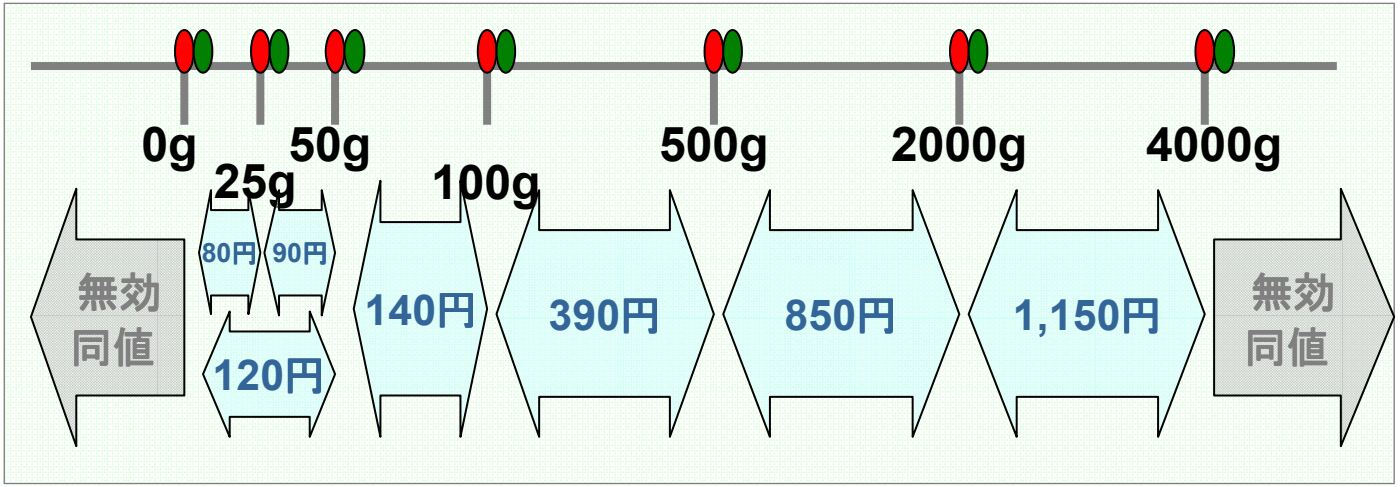
### 1. 形式

| 形式 |
| --- |
| 定型 |
| 定型外 |

### 2. 重量

| 重量 |
| --- |
| 0g |
| 25g |
| 50g |
| 100g |
| 500g |
| 2000g |
| 4000g |

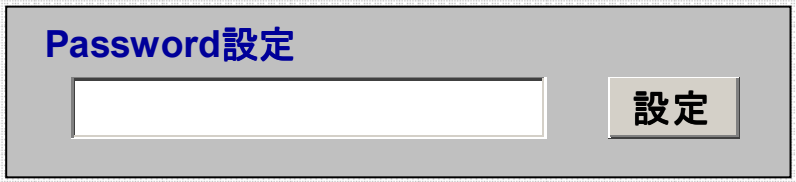
境界値がありそうだ



# 境界値分析 演習2

パスワード設定画面について、同値分割、境界値分析を利用してテストケースを作成せよ。
以下のパスワードの条件とし、これら条件以外の場合にはエラーとする。
尚、テストデータ、期待する結果も考えること。

【条件】
・パスワードは英字(a～z、A～Z)、数字(0～9)の組み合わせで8文字以上12文字以下とする。
・必ず英字と数字を含む必要がある。



# 境界値分析 演習2 ～回答例～

| No. | テストケース | データ(例) | 期待動作 |
|---|---|---|---|
| 1 | 文字数が7文字 | abcd123 | エラー |
| 2 | 文字数が8文字 | abcd1234 | 正常 |
| 3 | 文字数が12文字 | abcd12345678 | 正常 |
| 4 | 文字数が13文字 | abcd123456789 | エラー |
| 5 | 文字コードに「/」(0x29)を含む | abcd123/ | エラー |
| 6 | 文字コードに「0」(0x30)を含む | abcd0000 | 正常 |
| 7 | 文字コードに「9」(0x39)を含む | abcd9999 | 正常 |
| 8 | 文字コードに「@」(0x40)を含む | abc@1234 | エラー |
| 9 | 文字コードに「A」(0x41)を含む | AAAA1234 | 正常 |
| 10 | 文字コードに「Z」(0x5A)を含む | ZZZZ1234 | 正常 |
| 11 | 文字コードに「[」(0x5B)を含む | abc[1234 | エラー |
| 12 | 文字コードに「`」(0x60)を含む | abc`1234 | エラー |
| 13 | 文字コードに「a」(0x61)を含む | aaaa1234 | 正常 |
| 14 | 文字コードに「z」(0x7A)を含む | zzzz1234 | 正常 |
| 15 | 文字コードに「{」(0x7B)を含む | abc{1234 | エラー |
| 16 | 数字のみ | 12345678 | エラー |
| 17 | 英字のみ(大文字/小文字混在) | abcdABCD | エラー |
| 18 | 英字のみ(大文字のみ) | ABCDEFGH | エラー |
| 19 | 英字のみ(小文字のみ) | abcdefgh | エラー |
| 20 | 英数混在(英:大文字/小文字混在) | abCD1234 | 正常 |
| 21 | 英数混在(英:大文字のみ) | ABCD1234 | 正常 |
| 22 | 英数混在(英:小文字のみ) | abcd1234 | 正常 |

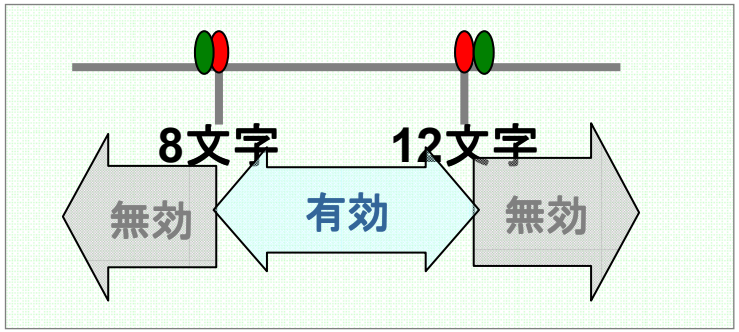
# 境界値分析 演習2 ～解説 1/2～

## 同値クラスの着目点

### 1. 文字数

| 文字数 |
| --- |
| 8文字 |
| 12文字 |

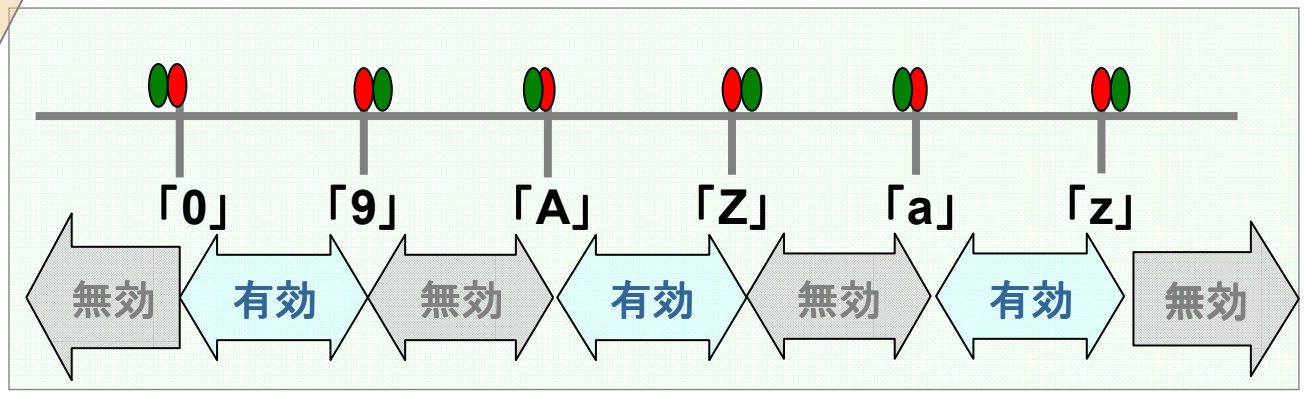
境界値




### 2. 文字コード(ASCII)

| 文字コード |
| --- |
| 0 (0x30) |
| 9 (0x39) |
| A (0x41) |
| Z (0x5A) |
| a (0x61) |
| z (0x7A) |

境界値



# 境界値分析 演習2 ～解説 2/2～

## 3. 文字の組合せ

| 文字組合せ | サブクラス |
|---|---|
| 数字のみ | – |
| 英字のみ | 大文字/小文字混在 |
| | 大文字のみ |
| | 小文字のみ |
| 英数字混在 | 大文字/小文字混在 |
| | 大文字のみ |
| | 小文字のみ |

回答例では3つの同値クラスを用いたが、観点は他にもある。

例)

・開始文字に注目して(数字始まりの場合、英字始まりの場合)

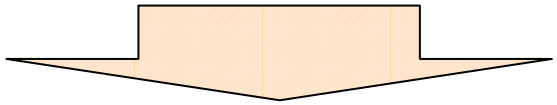
・組み合わせの割合に注目して(英字体数字が1:7や4:4)

# All-Pair法　1/4

## All-Pair法とは

組み合わせテストの設計において、すべての組み合わせを網羅するのではなく、「任意の2つの因子間ですべての水準の組み合わせが1つ以上存在する」という条件で、最小限の組み合わせ数にてテストを設計する技法。

アルゴリズムは一通りではない。



- 最小限のテストケースで、2因子間の組み合わせ網羅率を100%にする技法
- 少ないテストケースで網羅的にテストが可能

**Keyword:**

因子:　テスト対象機能(パラメータ)

水準:　パラメータが取り得る値

# All-Pair法　2/4

## なぜ2因子間なのか？

- FTFI (failure-triggering fault interaction)
  - フォールトに関係しているパラメータ数

| FTFI | エキスパートシステム | | | | OS | 組込み | Netscape N | Apache | DB |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 61 | 72 | 48 | 39 | 82 | 66 | 29 | 42 | 68 |
| 2 | 36 | 10 | 6 | 8 | * | 31 | 47 | 28 | 25 |
| 3 | *na | * | * | * | * | 2 | 19 | 19 | 5 |
| 4 | * | * | * | * | * | 1 | 2 | 7 | 2 |
| 5 | * | * | * | * | * | | 2 | 0 | |
| 6 | * | * | * | * | * | | 1 | 4 | |

*※出展:D.Richard Kuhn Software Fault Interactions and Implications for Software Testing ,IEEE Transactions on software engineering,Vol.30,2004*

**例えば「組込み」では全バグの97%までが2因子間で発生しており、3因子以上ではたったの3%。**

**であれば、3因子以上で発生する問題は大量のテストケースを用いて網羅的に行うよりも、別の技法で狙った方が効果的。**

# All-Pair法　3/4

## 例）4因子3水準の組み合わせをAll-Pair法を用いて設計

ある携帯電話では、以下のような設定値を持っており、各設定がされたときの起動を確認したい。

| 設定 | 着信音 | 待受け画面 | バイブ設定 | マナーモード |
|---|---|---|---|---|
| オプション1 | 電子音 | 標準画面 | ON | ON |
| オプション2 | 鐘の音 | カレンダー | OFF | OFF |
| オプション3 | 黒電話の音 | 動画 | オリジナル | カスタマイズ |

因子

水準

全組み合わせを考えると、
3×3×3×3=81パターン
のテストケースが必要。

All-Pair法

| No. | 着信音 | 待受け画面 | バイブ設定 | マナーモード |
|---|---|---|---|---|
| 1 | 電子音 | 標準画面 | ON | ON |
| 2 | 電子音 | カレンダー | OFF | OFF |
| 3 | 電子音 | 動画 | オリジナル | カスタマイズ |
| 4 | 鐘の音 | 標準画面 | オリジナル | OFF |
| 5 | 鐘の音 | カレンダー | ON | カスタマイズ |
| 6 | 鐘の音 | 動画 | OFF | ON |
| 7 | 黒電話の音 | 標準画面 | OFF | カスタマイズ |
| 8 | 黒電話の音 | カレンダー | オリジナル | ON |
| 9 | 黒電話の音 | 動画 | ON | OFF |

9パターンのテストケースで
2因子間の網羅100%の
テストができる。

# All-Pair法　4/4

## All-Pair法のポイント

- 2因子の組み合わせにのみ注目して設計する
- 同じ組み合わせは極力発生しないよう設計する
- 複数のアルゴリズムが存在し、それぞれ異なるテストケースが作成される場合がある
  →All-Pair法のツールは多数存在している。

※3因子以上の組み合わせは別の方法で考慮する
→ All-Pair法では考慮されないため

# All-Pair法　解説 1/5

## 例）4因子3水準の組み合わせをAll-Pair法を用いて設計

ある携帯電話では、以下のような設定値を持っており、各設定がされたときの起動を確認したい。

| 設定 | 着信音 | 待受け画面 | バイブ設定 | マナーモード |
|---|---|---|---|---|
| オプション1 | 電子音 | 標準画面 | ON | ON |
| オプション2 | 鐘の音 | カレンダー | OFF | OFF |
| オプション3 | 黒電話の音 | 動画 | オリジナル | カスタマイズ |

## All-Pair法での設計手順(例)

### 1．因子と水準を考える

| 着信音 | 待ちうけ | バイブ | マナー |
|---|---|---|---|
| A | B | C | D |
| 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 |
| 2 | 2 | 2 | 2 |

A:着信音
B:待ちうけ
C:バイブ
D:マナー
とする。

0:電子音
1:鐘の音
2:黒電話の音
とする。

# All-Pair法　解説 2/5

## 2. 2因子の組合せパターンを考える

| AとB | AとC | AとD | BとC | BとD | CとD |
|---|---|---|---|---|---|
| 00 | 00 | 00 | 00 | 00 | 00 |
| 01 | 01 | 01 | 01 | 01 | 01 |
| 02 | 02 | 02 | 02 | 02 | 02 |
| 10 | 10 | 10 | 10 | 10 | 10 |
| 11 | 11 | 11 | 11 | 11 | 11 |
| 12 | 12 | 12 | 12 | 12 | 12 |
| 20 | 20 | 20 | 20 | 20 | 20 |
| 21 | 21 | 21 | 21 | 21 | 21 |
| 22 | 22 | 22 | 22 | 22 | 22 |

Aが「0」、Bが「0」という意味

## 3. まずAとBのパターンを表に割り付ける

| 行No. | A | B |
|---|---|---|
| 1 | 0 | 0 |
| 2 | 0 | 1 |
| 3 | 0 | 2 |
| 4 | 1 | 0 |
| 5 | 1 | 1 |
| 6 | 1 | 2 |
| 7 | 2 | 0 |
| 8 | 2 | 1 |
| 9 | 2 | 2 |

# All-Pair法　解説 3/5

## 4. 次に「AとC」のパターン表よりCを割り付けてみる

※パターン表は2.で作成したものを参照

| 行No. | A | B | C |
|---|---|---|---|
| 1 | 0 | 0 | 0 |
| 2 | 0 | 1 | 1 |
| 3 | 0 | 2 | 2 |
| 4 | 1 | 0 | |
| 5 | 1 | 1 | |
| 6 | 1 | 2 | |
| 7 | 2 | 0 | |
| 8 | 2 | 1 | |
| 9 | 2 | 2 | |

| 行No. | A | B | C |
|---|---|---|---|
| 1 | 0 | 0 | 0 |
| 2 | 0 | 1 | 1 |
| 3 | 0 | 2 | 2 |
| 4 | 1 | 0 | |
| 5 | 1 | 1 | 0 |
| 6 | 1 | 2 | |
| 7 | 2 | 0 | |
| 8 | 2 | 1 | |
| 9 | 2 | 2 | |

「AとC」のパターンで上から4つ目の「10」の組合せは、そのままNo.4に「0」を割り付けると、BとCの組み合わせが、「00」(行No.1と同じ組合せ)となってしまうため、ここはBとCの組み合わせがまだ出ていないNo.5に割り付ける。
(No.5ではBとCの組合せは「10」でまだ出ていない。)

次ページへ

# All-Pair法　解説 4/5

| 行No. | A | B | C |
|---|---|---|---|
| 1 | 0 | 0 | 0 |
| 2 | 0 | 1 | 1 |
| 3 | 0 | 2 | 2 |
| 4 | 1 | 0 | 2 |
| 5 | 1 | 1 | 0 |
| 6 | 1 | 2 | 1 |
| 7 | 2 | 0 | 1 |
| 8 | 2 | 1 | 2 |
| 9 | 2 | 2 | 0 |

同様に、他の組み合わせ(ここでは「AとC」の組み合わせ割付時の「BとC」の組み合わせ)」がなるべく同じにならないように割り付けていく。

## 5. 次に「AとD」のパターン表よりDを割り付けてみる

| 行No. | A | B | C | D |
|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 |
| 2 | 0 | 1 | 1 | 1 |
| 3 | 0 | 2 | 2 | 2 |
| 4 | 1 | 0 | 2 | 1 |
| 5 | 1 | 1 | 0 | 2 |
| 6 | 1 | 2 | 1 | 0 |
| 7 | 2 | 0 | 1 | 2 |
| 8 | 2 | 1 | 2 | 0 |
| 9 | 2 | 2 | 0 | 1 |

この時も、「AとB」、「AとC」、「BとC」の組み合わせがなるべく同じにならないように割り付ける。

**これで2因子間網羅100%の無駄のない表が出来あがった!!**

# All-Pair法　解説 5/5

## 6.「A～C」、「0～2」を元の名前に戻してみる

| 行No. | 着信音 | 待受け画面 | バイブ設定 | マナーモード |
|---|---|---|---|---|
| 1 | 電子音 | 標準画面 | ON | ON |
| 2 | 電子音 | カレンダー | OFF | OFF |
| 3 | 電子音 | 動画 | オリジナル | カスタマイズ |
| 4 | 鐘の音 | 標準画面 | オリジナル | OFF |
| 5 | 鐘の音 | カレンダー | ON | カスタマイズ |
| 6 | 鐘の音 | 動画 | OFF | ON |
| 7 | 黒電話の音 | 標準画面 | OFF | カスタマイズ |
| 8 | 黒電話の音 | カレンダー | オリジナル | ON |
| 9 | 黒電話の音 | 動画 | ON | OFF |

## 7. テストケース風にすると

| No. | テストケース |
|---|---|
| 1 | 着信音:「電子音」、待受け画面:「標準画面」、バイブ設定:「ON」、マナーモード:「ON」中に着信した際の起動確認 |
| 2 | 着信音:「電子音」、待受け画面:「カレンダー」、バイブ設定:「OFF」、マナーモード:「OFF」中に着信した際の起動確認 |
| 3 | 着信音:「電子音」、待受け画面:「動画」、バイブ設定:「オリジナル」、マナーモード:「カスタマイズ」中に着信した際の起動確認 |
| 4 | 着信音:「鐘の音」、待受け画面:「標準画面」、バイブ設定:「オリジナル」、マナーモード:「OFF」中に着信した際の起動確認 |
| 5 | 着信音:「鐘の音」、待受け画面:「カレンダー」、バイブ設定:「ON」、マナーモード:「カスタマイズ」中に着信した際の起動確認 |
| 6 | 着信音:「鐘の音」、待受け画面:「動画」、バイブ設定:「OFF」、マナーモード:「ON」中に着信した際の起動確認 |
| 7 | 着信音:「黒電話の音」、待受け画面:「標準画面」、バイブ設定:「OFF」、マナーモード:「カスタマイズ」中に着信した際の起動確認 |
| 8 | 着信音:「黒電話の音」、待受け画面:「カレンダー」、バイブ設定:「オリジナル」、マナーモード:「ON」中に着信した際の起動確認 |
| 9 | 着信音:「黒電話の音」、待受け画面:「動画」、バイブ設定:「ON」、マナーモード:「OFF」中に着信した際の起動確認 |

# 4. さいごに

# さいごに

今日はありがとうございました。

今日、覚えた技法はテスト技法のほんの一部です。

やってみた例題もすごくきれいな例題で、実際の現場ではこんなシチュエーションはなかなかないと思います。

テスト設計のポイントは、「設計技法を覚えて使えること」だけではなく「テスト設計技法適用の選球眼」です。

今後、テスト設計技法を勉強する際には「自分たちの現場でどう使おう？」を常に考えながら勉強することで、ちゃんと選球眼を養うことができます。

今日のセッションが、明日からの「ちゃんとしたテスト」への第一歩となってもらえると幸いです。

# 参考文献


・ソフトウェア・テストの技法 第2版　(Glenford J.Myers)

・はじめて学ぶソフトウェアのテスト技法　(Lee Copeland)

・ソフトウェアテスト技法　(Boris Beizer)

・ソフトウェア品質知識体系ガイド —SQuBOK Guide—　(SQuBOK策定部会)

・テスト技術者資格制度 Foundation Levelシラバス　日本語版
(International Software Testing Qualifications Board )

・ソフトウェアテスト標準用語集（日本語版）
(International Software Testing Qualifications Board用語集作業班 )

・JSTQB教科書 JSTQB認定テスト技術者 Foundation Level試験
(大西 建児/勝亦 匡秀/加藤 大受/佐々木 方規/鈴木 三紀夫/町田 欣史/湯本 剛/吉澤 智美)

・JaSST'09Tokyo資料「初心者向けテスト技法演習 - テスト技法の基本のキ-」
http://jasst.jp/archives/jasst09e/pdf/E4-1.pdf　(秋山浩一)