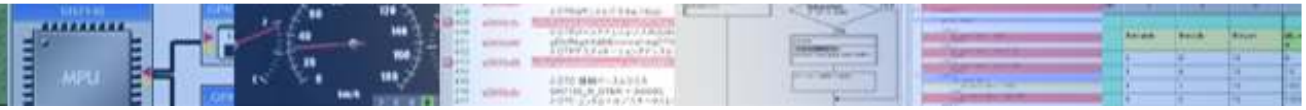
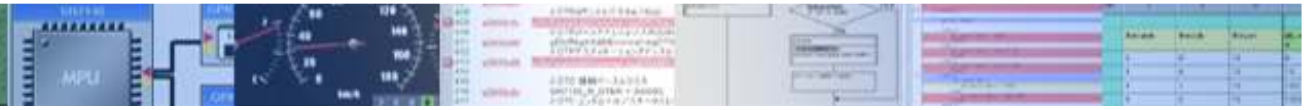


# 単体テスト設計手法「要素分析」と標準プロセス化ツールについて

## ～大規模開発向け単体テストツール「ユニットマスター」の紹介～

GAIO TECHNOLOGY



# ガイオテクノロジーのプロフィール

# ガイオ・テクノロジーとは？

- 組み込み開発ツールの開発および販売
  クロスコンパイラ/テストツール/仮想検証ツール/プロトタイピングツール/解析ツール
- テスト・検証に関わるサービスやコンサルテーション
- 受託開発

ガイオ・テクノロジー株式会社
GAIO TECHNOLOGY CO., LTD.

- 設立　1980年
- 資本金　2億9800万円
- 従業員　80名
- 本社:　横浜
- 事業所:　東京、松山
- 子会社:　GAIO INC(USA)



## 2009年度の事業計画で挙げた事業ドメイン

- 省力・省資源
  - 自動テストで省力化（自動テストツール/環境）
  - 試作レス・実機レスで省資源化（試作レス/実機レス開発環境）
  - 部品IP提案で省力・省資源化（部品IP提案）
- 品質証明 信頼性証明
  - ツール&コンサルテーションで品質証明のお手伝い
  - ツール&コンサルテーションで信頼性証明のお手伝い
  - 安全認証に対応するツールと開発品質
- 開発サービス
  - MBD能力&実績を有するHW/SW開発サービス
  - レガシーからモデルベースへ資産移行サービス（RM代行）
  - HMI開発ツール/コンパイラ/解析ツールなど専用ツールの開発

GAIO TECHNOLOGY

# ガイオの商材2009

New
Ver.Up

## カバレッジマスター
・テストシナリオ品質向上
・SW品質指標が欲しい
カバレッジ測定ツール

## CasePlayer2
・静的解析したい
・プログラムを可視化したい
・MISRA-Cチェックしたい

## 単体テスト代行サービス
・第3者認証により品質を高めたい
・検証工数の確保

## MC-Checker
・モデルとコードが一致しているか確認したい

## 仮想テストツール
SCS No.1 システムシミュレータ
・仮想テスト装置を構築したい

## ユニットマスター
・大規模ソフトウェアの単体テストを実現したい
・SWテストを効率的に実施したい

## リバースモデリング代行
・レガシーコードをモデルに移行したい
・モデルベース開発に移行したいが担当がいない

## VECU-G
・HILSコストを低減したい
・プラントモデルと連携してシミュレーション
ECUシミュレータ
・複数ECUの連携ロジックを確認したい
・IGNオン時の振る舞いのデバッグ

## クロスコンパイラ
クロスコンパイラ

## 経営視点のSW品質改善診断
・単体テストを中心に導入効果を経営視点で示したい、計測したい
・品質観点で単体テストの効果を明示したい

## MBD開発サービス

## VMPF-G
・検証を自動化したい
・ECU間通信部の確認をしたい
複数ECUシミュレータ

## G-VPM
・搬送路メカの動きをビジュアルに見たい

## デジタル制御電源向け開発ツール
・デジタル電源制御のツールを導入したい
・デジタル電源開発の分野に参入したい

## ガイオプロセッサコア
・廉価なマイコンIPでコストを圧縮したい
・IPを使ってFPGAベースの開発をしたい

## VSFS-G
FlexRayシミュレータ

## ASLS-G
・ASICの検証にマイコンの出力を使いたい

## RtFT
・実機を使った検証を自動化したい

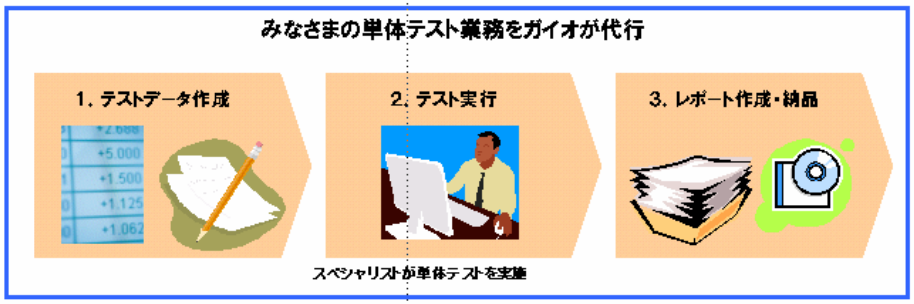
## VESS-G
プリンターエンジン向けシミュレーション

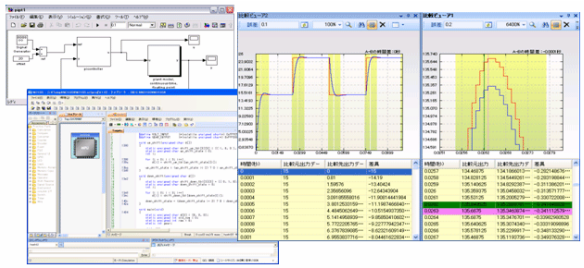
## MFSS-G
システム全体(メカ/エレキ/ソフト)をシミュレーションしたい

GAIO TECHNOLOGY

# ガイオの単体テストツールとソリューション

- **カバレッジマスター**
  - マイコンシミュレータ搭載型単体テスト自動化ツール
- **単体テスト代行サービス**
  - カバレッジマスターをベースにした単体テスト委託サービス
- **MC-Checker**
  - MATLAB/Simulink連携型モデルベース開発(MBD)用単体テストツール
- **単体テストセミナー**
  - 単体テストスキル向上セミナー
  - eラーニング
- **新単体テストツール「ユニットマスター」**
  - **大規模アプリ向け単体テスト標準化プロセスツール**

本日の講演のメイン製品

# カバレッジマスターwinAMSの導入実績/事例

## ■ 導入実績

- 自動車業界、鉄道、デバイス関連、通信機器、空調関連、メカトロ機器、プリンター複写機、FA機器、計器関連、エネルギー関連、交通機器関連、等々

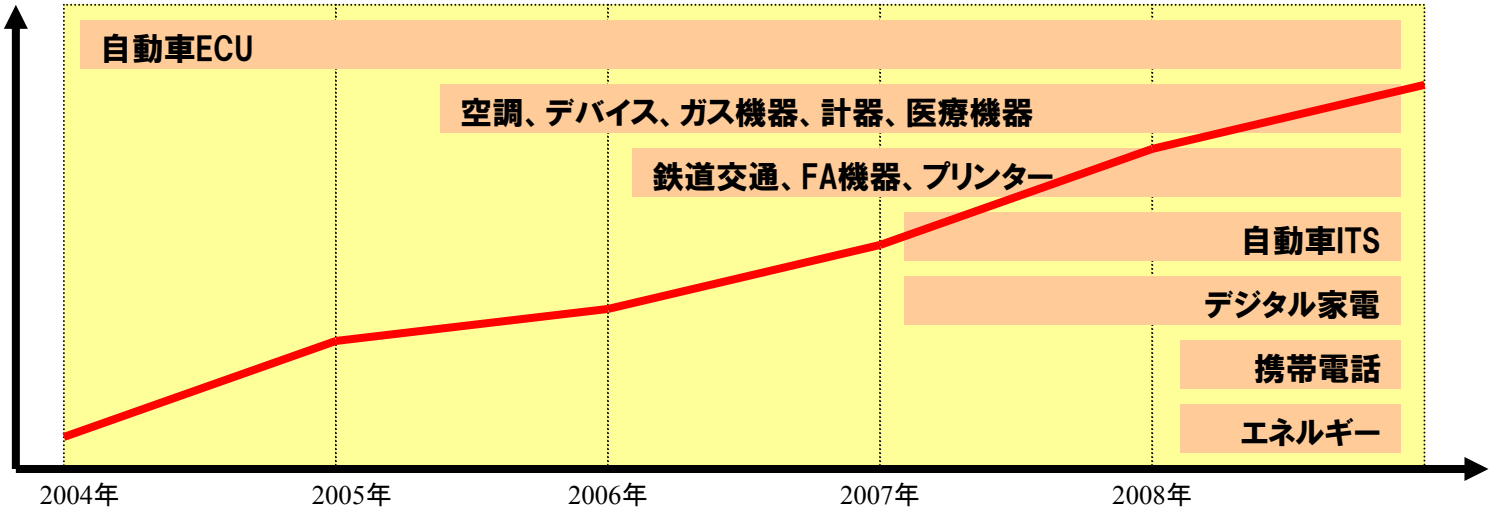
## ■ 多数のユーザ事例を発表

- フェリカネットワーク(株)　:2005/12 組込み情報誌「ガイオ倶楽部」記事にて発表
- アイシン精機(株)　:2006/08「組込みソフト単体テスト成功事例セミナー」にて発表
- 日産自動車(株)　:2006/08「組込みソフト単体テスト成功事例セミナー」にて発表
- ソニーLSIデザイン(株):2007/11「単体試験の賢い選択・活用セミナー」にて発表
- パイオニア(株)　:2008/02 組込み情報誌「ガイオ倶楽部」記事にて発表
- パナソニック(株)　:2009/06「ガイオ品質向上セミナー」にて発表


GAIO TECHNOLOGY

# winAMSの販売実績

## ■ リリースからの販売伸び率と主な販売サイト



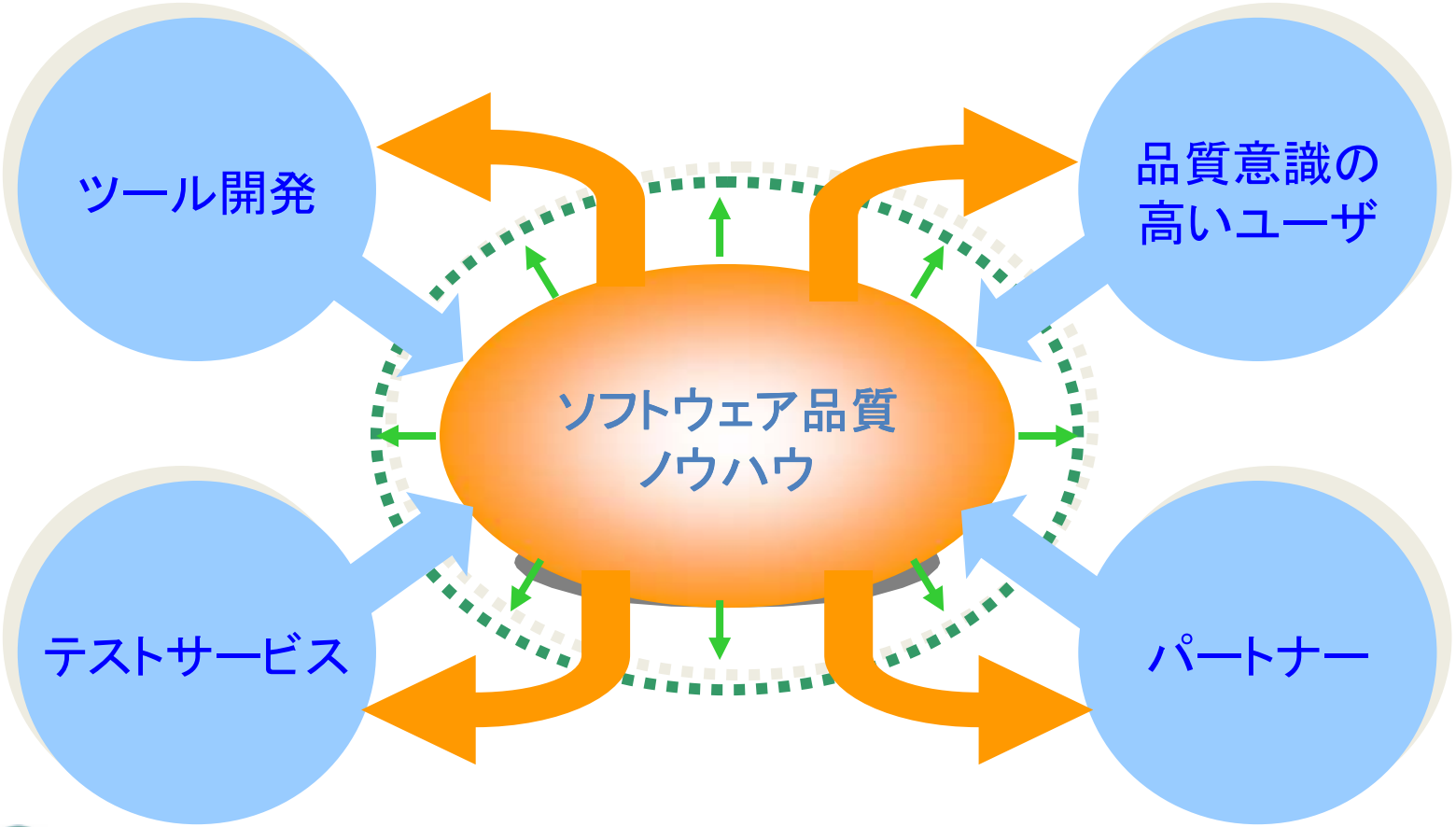

## ■ 2004年～2007年と2008年以降の業界毎の販売比率

| | 2004年～2007年 | 2008年～現在 |
|---|---|---|
| 自動車ECU | 48.3% | 30.6% |
| 自動車ITS | 4.5% | 19.1% |
| 鉄道/交通 | 3.7% | 3.5% |
| エネルギー | 2.3% | 4.7% |
| デジタル家電 | 3.2% | 22.6% |
| 携帯電話 | 0.2% | 1.2% |

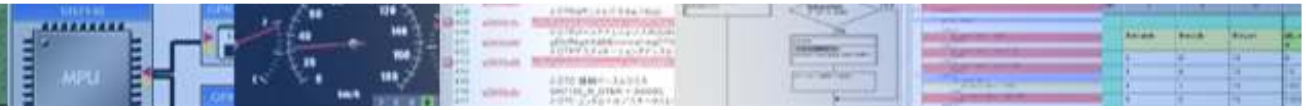
GAIO TECHNOLOGY

# ガイオの単体テストの強み

- ツール開発からテストサービスまでを提供可能な稀有なベンダー
- いろんな確度からソフトウェアの品質向上を見つめ直しています。




GAIO TECHNOLOGY

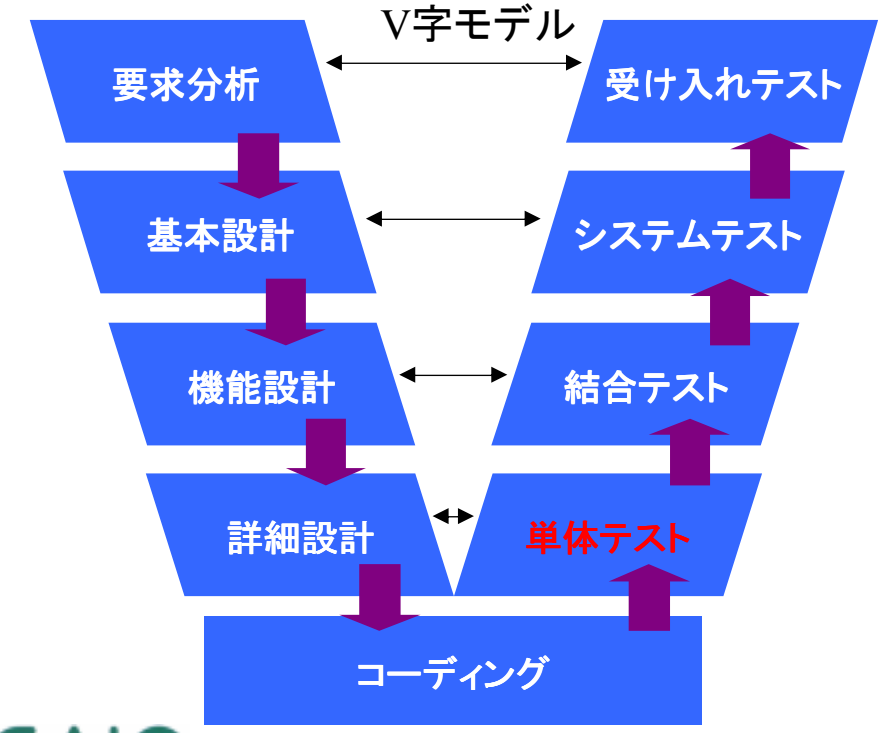


# 単体テストとソフトウェア

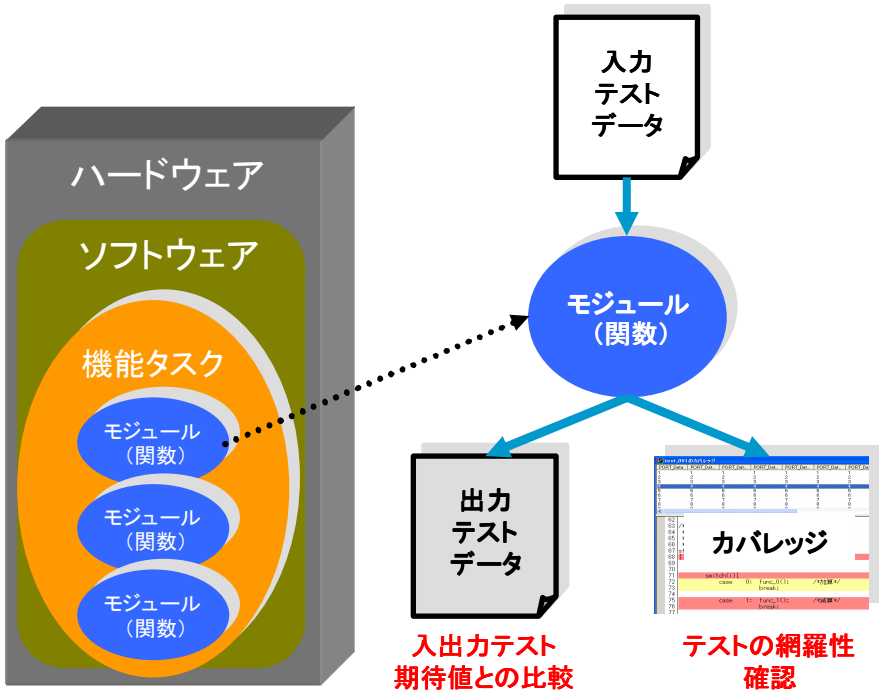
GAIO TECHNOLOGY

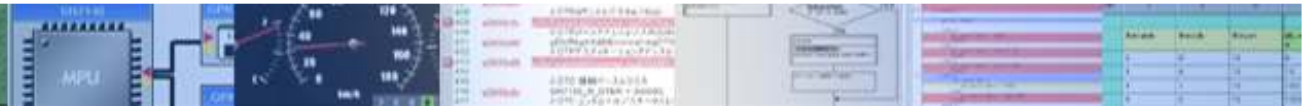
# モジュール単体テストは「最初」で「最小」

- コーディングしたソフトウェアの**最初**の品質確認テスト
- コーディングしたソフトウェアの**最小**単位における品質確認テスト




モジュール単体テストのイメージ





# 単体テストの意味と目的

- 検証の前倒しで手戻りの無い効率的な検証が実現可能
- 後工程では発見困難な潜在不具合を無くすことができる
- 定量的、客観的な単体テストを実施し、プログラマー個人依存部分を排除
- モジュール再利用時の品質確保を行うため
  - 最近の不具合傾向として、モジュールの再利用時、部分修正で発生することが多い
- 国際認証に向けたテストエビデンス(カバレッジ基準)への対応

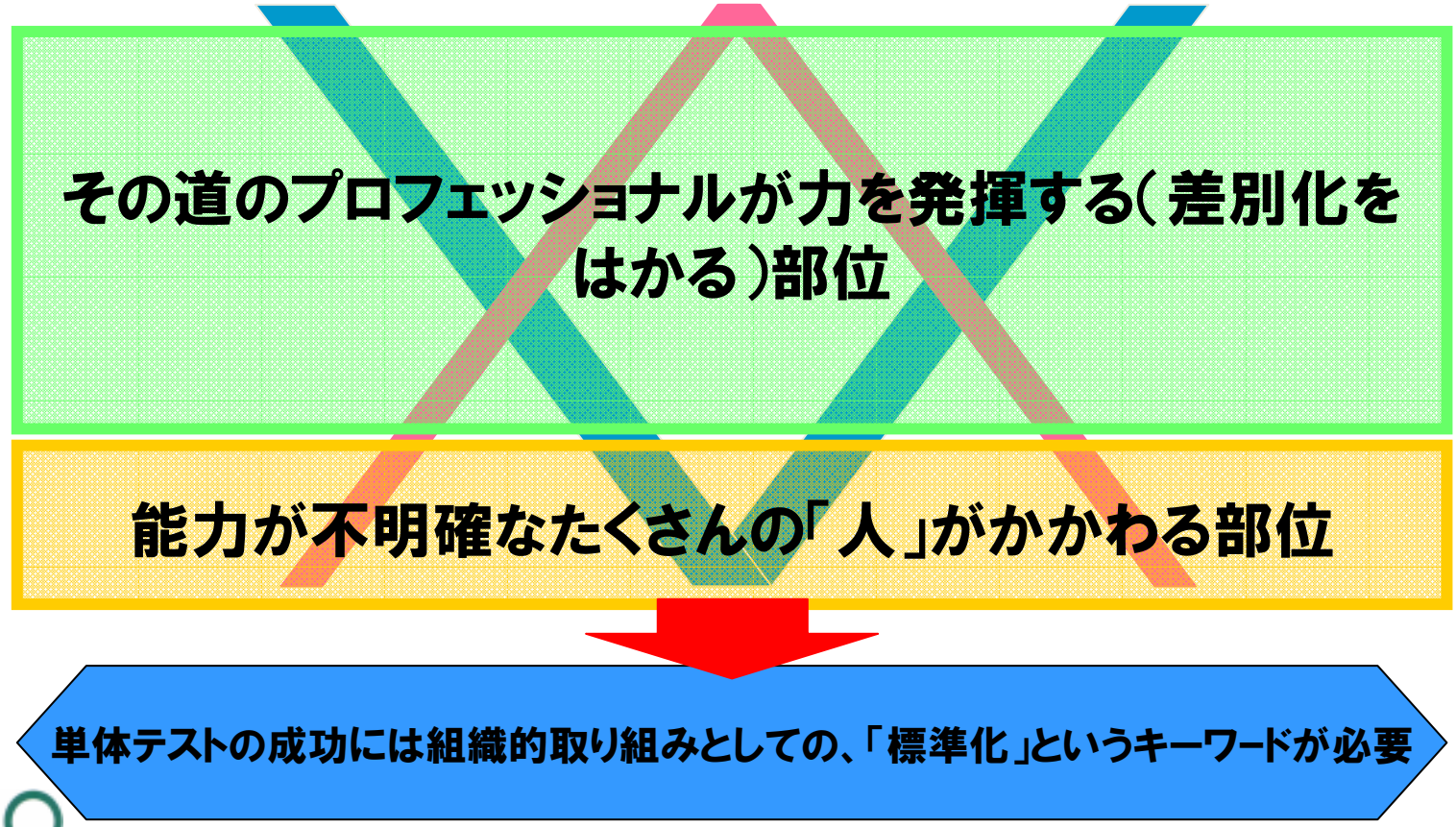
**単体テストは、全てのコードを動作させることができる**
**最初で最後の工程!!**

GAIO TECHNOLOGY

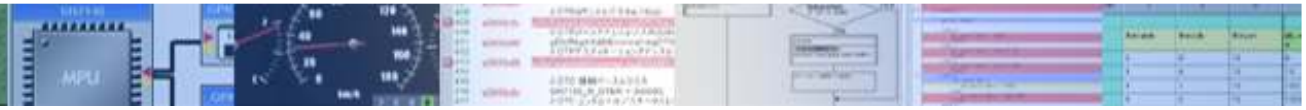
# ソフトウェア品質と単体品質の位置づけ

■ V字モデルの携わる開発者のスキルと人数について

- V字モデルの下位層ほど携わる人が多い
- V字モデルの下位層ほどスキルのバラツキが大きい




GAIO TECHNOLOGY

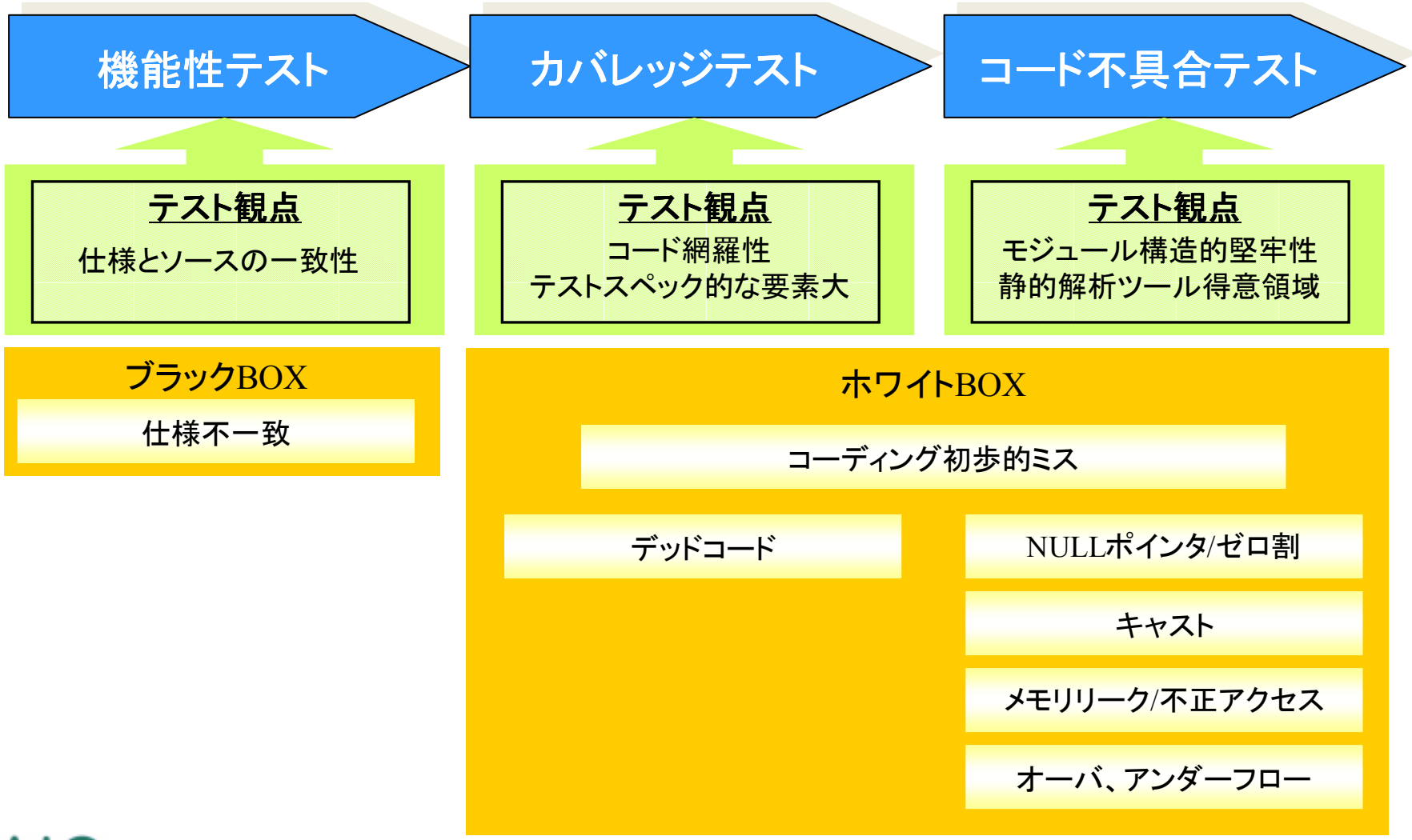


# 一般的な単体テストの課題

## ーコードカバレッジのみのテスト指針ー

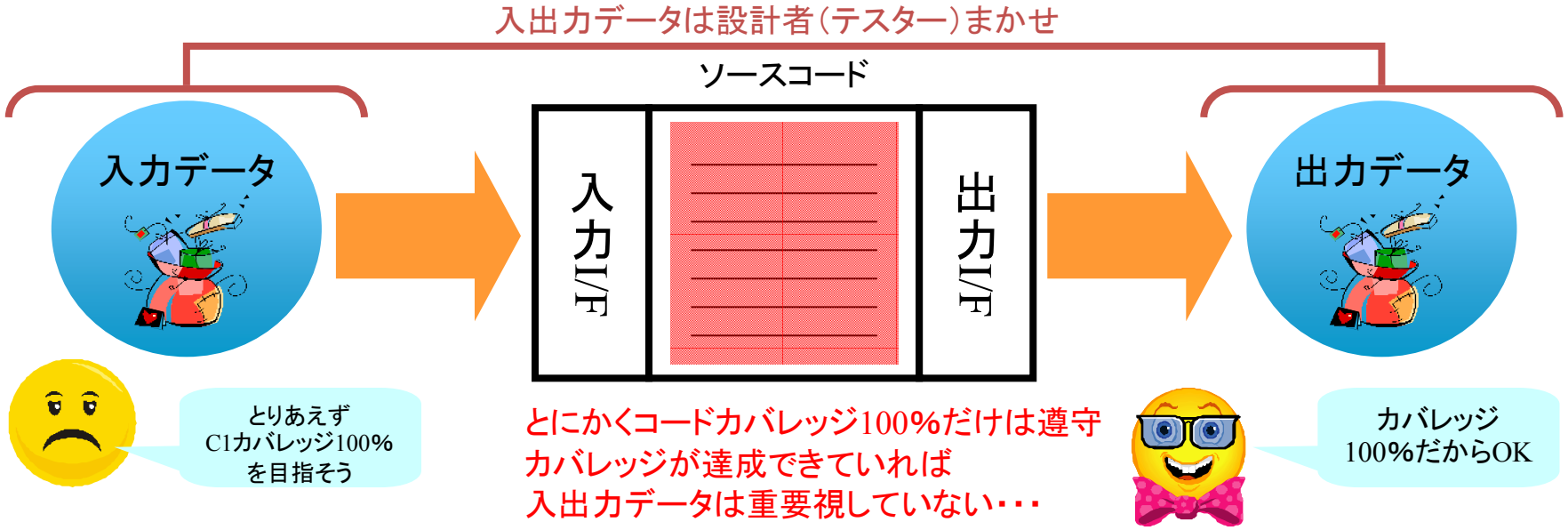
GAIO TECHNOLOGY

# 単体テストで必要なテスト観点と検出可能な不具合



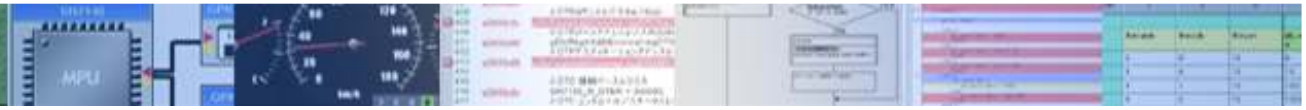
# 単体テストとコードカバレッジ100%

- 単体テストにおけるカバレッジ指標を手段では無く、目的にしてしまい、工数に見合った効果が出ていない




- 不具合を検出するためのテストデータの内容(精度)については担当者任せ
  - コードカバレッジを100%にして必ず発見できる不具合は「デッドコード」のみ
  - 単体テストに必要な組織的な標準化というキーワードとは縁遠い
  - 単体テストをしても不具合の検出ができないというユーザはこの傾向が強い

GAIO TECHNOLOGY



# 単体テスト品質を向上させるためには

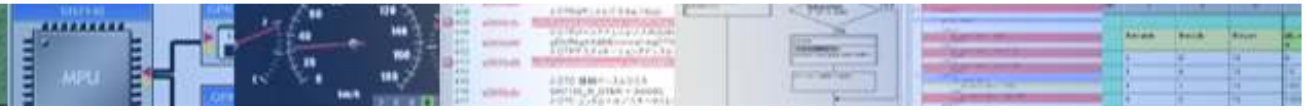
- テストデータの中身をカバレッジ重視型からモジュール品質確認重視型へ移行
  - 機能性(仕様一致性)のテスト
  - カバレッジのテスト
  - コード不具合(モジュール構造的堅牢性)のテスト

設計者依存になりがち

- 開発者スキルに依存させないために、テストデータ設計をプロセス化する
  - ① モジュール入出力要因の抽出
    - 引数や参照変数、リターン値などのモジュール入出力要因を抽出する
  - ② モジュール入出力要因への要素(付与値/期待値)の洗い出し
    - ブラック/ホワイトBOX視点におけるテスト付与値を漏れなく設定する。
  - ③ 要素を組み合わせてモジュールテストパターンを作成
    - モジュールやモジュール入出力要素の特性、テスト観点や指針に応じて、設定した各入出力要素の付与値を組み合わせて、モジュールテスト用のデータを作成する。
- 単体テスト設計の各プロセスでのレビューを容易にする

**単体テスト設計手法「要素分析」を用いて、実現!!**

GAIO TECHNOLOGY

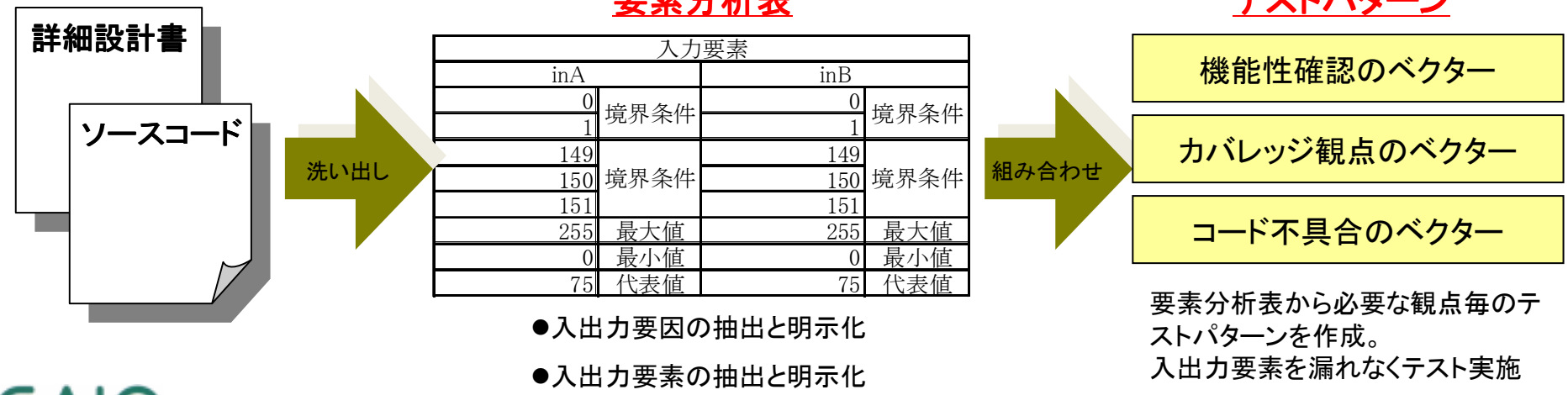


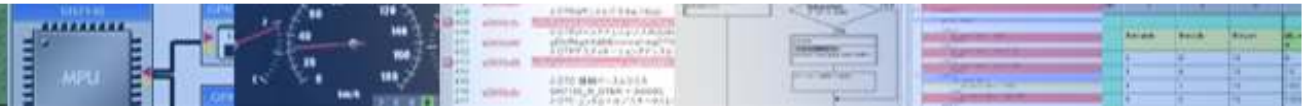
# 開発技術者の「スキル」に依存しない
# 単体テスト設計を可能にする
# 「要素分析」手法

GAIO TECHNOLOGY

# 要素分析とは？

- モジュールに対する入出力やテスト要素を漏れなく抽出して 明確にすること
  - 入出力の種類ごとのテスト要因を漏れなく抽出することによって関数に対しての入出力データの不足をなくすことを目的とする
  - 単体テストに含めなければならない「要素」を明確にする
- 単体テスト設計の内容を示す「要素分析表」を作成
  - ① 引数、外部変数、戻り値などの入出力要因を抽出
  - ② 入出力要因に対して、与えるべき要素（付与値/期待値）を抽出
  - ③ 要素を組み合わせてテストパターンを作成





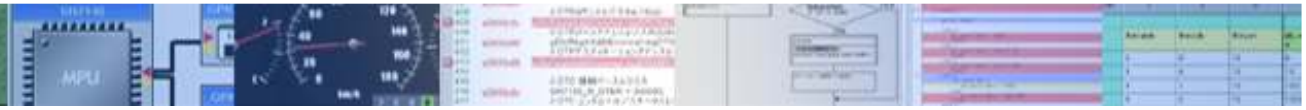
# ガイオの提案する「要素分析手法」のメリット

■ 設計時に見逃してしまう入力条件を 確実にテストに含める
- テスト担当者の判断によらないテスト設計が可能
- 仕様に関わる不具合検出を容易にする
  - ― 例：誤って参照すべきグローバル変数とは違う変数を参照する不具合

■ テスト担当者のスキルや判断に依存しない テストの標準化を実現する
- テスト設計指針書（データの設計方法）を明確に定めることができる
- テスト設計は、指針書に従い「機械的に」行うことができる

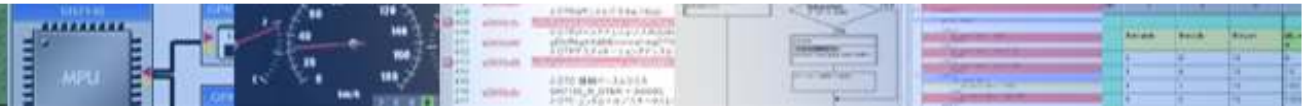
■ 要素分析表を参照することで テストのレビューを容易にする
- 実施した単体テストに含まれているテスト要因が明確になる
- テストの品質が「客観的」に判断するエビデンスとして利用できる

GAIO TECHNOLOGY



# 要素分析表を使ったテスト設計例

GAIO TECHNOLOGY



# （参考）単体テスト設計： サンプル関数仕様とコード

## ■ func5 () 関数仕様

- 入力仕様:
  - ー unsigned char型 引数 inA, inB
  - ー 入力値レンジ 0～150
- 戻り値 unsigned char型:
  - ー 2入力inA, inBの加算値
  - ー 条件:
    入力値がレンジ外の時は0
    加算値が200以上の時は200

テスト対象関数

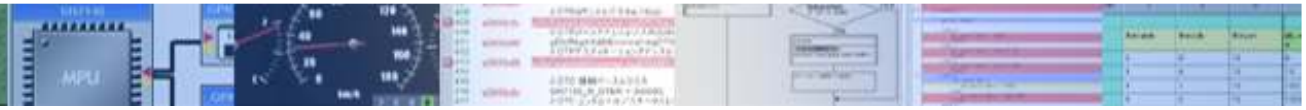
```
unsigned char
func5( unsigned char inA, unsigned char inB )
{
    unsigned char tmp;

    // 入力範囲の確認
    if( inA > 150 )
    {
        return 0; // 入力レンジエラー
    }
    if( inB > 150 )
    {
        return 0; // 入力レンジエラー
    }

    // 加算処理
    tmp = inA + inB;

    // 加算結果が200以上なら 200 を返す
    if( tmp >= 200 )
    {
        return 200;
    }
    // それ以外は加算結果を返す
    return tmp;
}
```

GAIO TECHNOLOGY

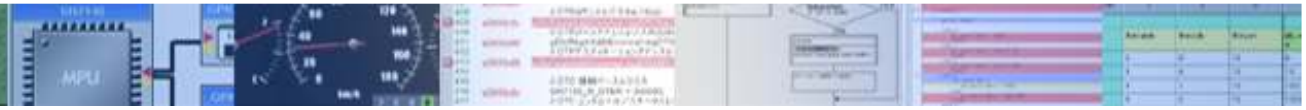


# （参考）単体テスト設計例： 入力分析表を作成

## ■ 入力変数に関する テスト指針の例↓

- 変数が境界条件を持つ場合、境界値に対して、[境界値−1]，[境界値]，[境界値+1] の3値を付与すること
- 変数の型に対して、[最大値]，[最小値] を付与すること
- 同値分割により範囲が定められたときは、有効なレンジの中心値を代表値として付与する

| | 入力要素 | | | |
|---|---|---|---|---|
| | inA | | inB | |
| 変数レンジの下限値0への境界条件→<br>型がunsignedのため [境界値-1]は設定しない | 0 | 境界条件 | 0 | 境界条件 |
| | 1 | | 1 | |
| 変数レンジの上限値150への境界条件→ | 149 | 境界条件 | 149 | 境界条件 |
| | 150 | | 150 | |
| | 151 | | 151 | |
| 変数の型（unsigned char）の最大最小値→ | 255 | 最大値 | 255 | 最大値 |
| | 0 | 最小値 | 0 | 最小値 |
| 変数レンジの中間値→ | 75 | 代表値 | 75 | 代表値 |



# （参考）単体テスト設計例： 入力要素組み合わせ作成

## ■ 入力要素の組合せ方法は「テスト指針書」に従う

- どの様な組合せをテストに含めるかは 指針書に明記する

## ■ 入力要素の組み合わせに関する テスト指針の例↓

- 境界条件値は、対象付与値の全数組み合わせを作成する
  - ― 全パスを網羅するため
- 最大値、最小値は、対象付与値の全数組み合わせを作成する
  - ― 演算のデータカバレッジを網羅するため
- 代表値がある場合は、少なくとも1度はテストパターンに含まれるようにする

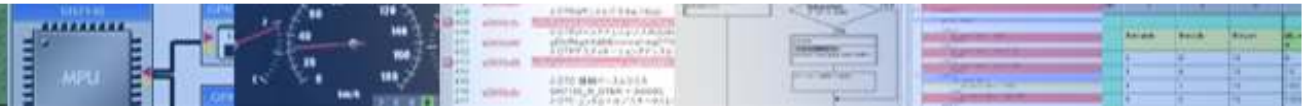
| 入力要素 | | | |
|---|---|---|---|
| inA | | inB | |
| 0 | 境界条件 | 0 | 境界条件 |
| 1 | | 1 | |
| 149 | 境界条件 | 149 | 境界条件 |
| 150 | | 150 | |
| 151 | | 151 | |
| 255 | 最大値 | 255 | 最大値 |
| 0 | 最小値 | 0 | 最小値 |
| 75 | 代表値 | 75 | 代表値 |

境界条件は全数組み合わせを作成

最大最小値は全数組み合わせを作成

代表値は少なくとも1度使われるように

GAIO TECHNOLOGY



# （参考）単体テスト設計例： 出力分析表を作成

## ■ 出力として確認すべき期待値の表を作成

- 期待値として確認しておくべき値を 表にまとめておく
  - ― 例外条件として規定されている出力値
  - ― 特異な演算結果など

## ■ この期待値が 組合せデータの期待値として得られるかを確認

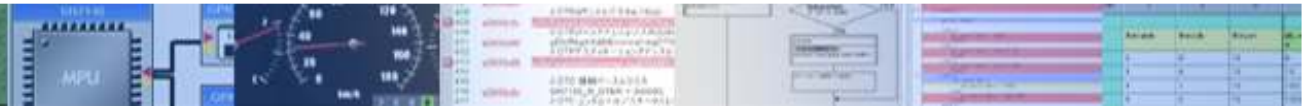
- 結果がこの期待値となるテストが行えるかを確認する
- 組み合わせたテストデータから 想定した期待値が得られるかを確認する

一条件:
入力値がレンジ外の時は0
加算値が200以上の時は200

→

| 出力要素 | |
|---|---|
| 戻り値 | |
| 0 | 下限値 |
| 200 | 上限値 |

GAIO TECHNOLOGY



# （参考）単体テスト設計例： 組み合わせを作成

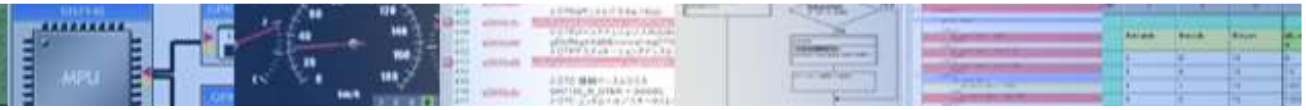
## ■ テスト指針書に従って正しく組み合わせを作成

| | COMMENT | 1 | 2 |
|---|---|---|---|
| COMMENT | | | |
| NAME | コメント | @inA | @inB |
| 1 | | ;境界条件組み | |
| 2 | | 0 | 0 |
| 3 | | 0 | 1 |
| 4 | | 0 | 149 |
| 5 | | 0 | 150 |
| 6 | | 0 | 151 |
| 7 | | 1 | 0 |
| 8 | | 1 | 1 |
| 9 | | 1 | 149 |
| 10 | | 1 | 150 |
| 11 | | 1 | 151 |
| 12 | | 149 | 0 |
| 13 | | 149 | 1 |
| 14 | | 149 | 149 |
| 15 | | 149 | 150 |
| 16 | | 149 | 151 |
| 17 | | 150 | 0 |
| 18 | | 150 | 1 |

| | COMMENT | 1 | 2 |
|---|---|---|---|
| COMMENT | | | |
| NAME | コメント | @inA | @inB |
| 19 | | 150 | 149 |
| 20 | | 150 | 150 |
| 21 | | 150 | 151 |
| 22 | | 151 | 0 |
| 23 | | 151 | 1 |
| 24 | | 151 | 149 |
| 25 | | 151 | 150 |
| 26 | | 151 | 151 |

| | | | |
|---|---|---|---|
| 29 | | ;最大最小値組み | |
| 30 | | 255 | 255 |
| 31 | | 255 | 0 |
| 32 | | 0 | 255 |
| 33 | | 0 | 0 |
| 34 | | ;代表値,,, | |
| 35 | | 75 | 75 |
| 36 | | | |

GAIO TECHNOLOGY



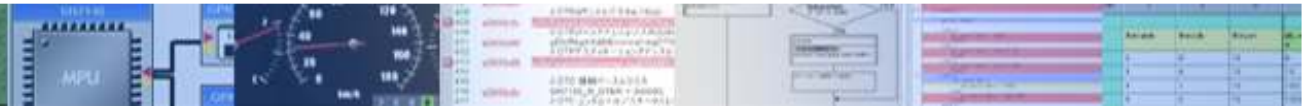
# （参考）単体テスト設計例： 期待値設定、出力要素確認

- **組み合わせたテストベクターに対する期待値を設定**
  - 期待値は関数仕様を元に設定（ソースコードから逆算してはならない）
- **出力要素分析表の期待値が全て含まれていることを確認**

| | COMMENT | 1 | 2 | 3 |
|---|---|---|---|---|
| COMMENT | | | | |
| NAME | コメント | @inA | @inB | func5@@ |
| 1 | ;境界条件組み合わせ,,,, | | | |
| 2 | | 0 | 0 | 0 |
| 3 | | 0 | 1 | 1 |
| 4 | | 0 | 149 | 149 |
| 5 | | 0 | 150 | 150 |
| 6 | | 0 | 151 | 0 |
| 7 | | 1 | 0 | 1 |
| 8 | | 1 | 1 | 2 |
| 9 | | 1 | 149 | 150 |
| 10 | | 1 | 150 | 151 |
| 11 | | 1 | 151 | 0 |
| 12 | | 149 | 0 | 149 |
| 13 | | 149 | 1 | 150 |
| 14 | | 149 | 149 | 200 |
| 15 | | 149 | 150 | 200 |
| 16 | | 149 | 151 | 0 |
| 17 | | 150 | 0 | 150 |
| 18 | | 150 | 1 | 151 |

| | COMMENT | 1 | 2 | 3 |
|---|---|---|---|---|
| COMMENT | | | | |
| NAME | コメント | @inA | @inB | func5@@ |
| 19 | | 150 | 149 | 200 |
| 20 | | 150 | 150 | 200 |
| 21 | | 150 | 151 | 0 |
| 22 | | 151 | 0 | 0 |
| 23 | | 151 | 1 | 0 |
| 24 | | 151 | 149 | 0 |
| 25 | | 151 | 150 | 0 |
| 26 | | 151 | 151 | 0 |
| 29 | ;最大最小値組み合わせ,,, | | | |
| 30 | | 255 | 255 | 0 |
| 31 | | 255 | 0 | 0 |
| 32 | | 0 | 255 | 0 |
| 33 | | 0 | 0 | 0 |
| 34 | ;代表値,,, | | | |
| 35 | | 75 | 75 | 150 |
| 36 | | | | |

GAIO TECHNOLOGY



# （参考）単体テスト設計例： 実行後 期待値比較を行うと・・・

## ■ 2つの入力要因の最大値同士の組合せで 期待値と異なる結果になる

- No14のデータ [inA=149、inB=149] で結果が200にならない

| | COMMENT | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| COMMENT | | | | | |
| NAME | コメント | @inA | @inB | func5@@ | 合否 |
| 1 | ;境界条件組み合わせ,,,, | | | | |
| 2 | | 0 | 0 | 0 | OK |
| 3 | | 0 | 1 | 1 | OK |
| 4 | | 0 | 149 | 149 | OK |
| 5 | | 0 | 150 | 150 | OK |
| 6 | | 0 | 151 | 0 | OK |
| 7 | | 1 | 0 | 1 | OK |
| 8 | | 1 | 1 | 2 | OK |
| 9 | | 1 | 149 | 150 | OK |
| 10 | | 1 | 150 | 151 | OK |
| 11 | | 1 | 151 | 0 | OK |
| 12 | | 149 | 0 | 149 | OK |
| 13 | | 149 | 1 | 150 | OK |
| 14 | | 149 | 149 | 42?(200) | NG |
| 15 | | 149 | 150 | 43?(200) | NG |
| 16 | | 149 | 151 | 0 | OK |
| 17 | | 150 | 0 | 150 | OK |
| 18 | | 150 | 1 | 151 | OK |

| | COMMENT | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| COMMENT | | | | | |
| NAME | コメント | @inA | @inB | func5@@ | 合否 |
| 19 | | 150 | 149 | 43?(200) | NG |
| 20 | | 150 | 150 | 44?(200) | NG |
| 21 | | 150 | 151 | 0 | OK |
| 22 | | 151 | 0 | 0 | OK |
| 23 | | 151 | 1 | 0 | OK |
| 24 | | 151 | 149 | 0 | OK |
| 25 | | 151 | 150 | 0 | OK |
| 26 | | 151 | 151 | 0 | OK |
| 27 | | 151 | 255 | 0 | OK |
| 28 | | 151 | 0 | 0 | OK |
| 29 | ;最大最小値組み合わせ,,, | | | | |
| 30 | | 255 | 255 | 0 | OK |
| 31 | | 255 | 0 | 0 | OK |
| 32 | | 0 | 255 | 0 | OK |
| 33 | | 0 | 0 | 0 | OK |
| 34 | ;代表値,,, | | | | |
| 35 | | 75 | 75 | 150 | OK |
| 36 | | | | | |

GAIO TECHNOLOGY

# （参考）単体テスト設計例： 潜在不具合が発見された！

■ 「最大値同士の組合せ」の入力要因がなければ この不具合は発見できない

- この要因は技術者の判断でなく、「テスト指針」により機械的に設計されている
- テスト設計者の判断やスキルに依存することなく 不具合を検出できる

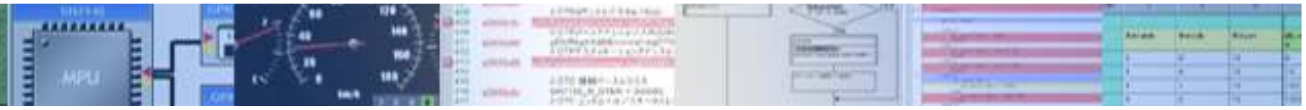
```
unsigned char func5( unsigned char inA, unsigned char inB )
{
    unsigned char tmp;

    // 入力範囲の確認
    if( inA > 150 )
    {
        return 0; // 入力レンジエラー
    }
    if( inB > 150 )
    {
        return 0; // 入力レンジエラー
    }

    // 加算処理
    tmp = inA + inB;

    // 加算結果が200以上なら 200 を返す
    if( tmp >= 200 )
    {
        return 200;
    }
    // それ以外は加算結果を返す
    return tmp;
}
```

変数tmpがunsigned char のため、255以上の演算結果がオーバーフローしてしまう

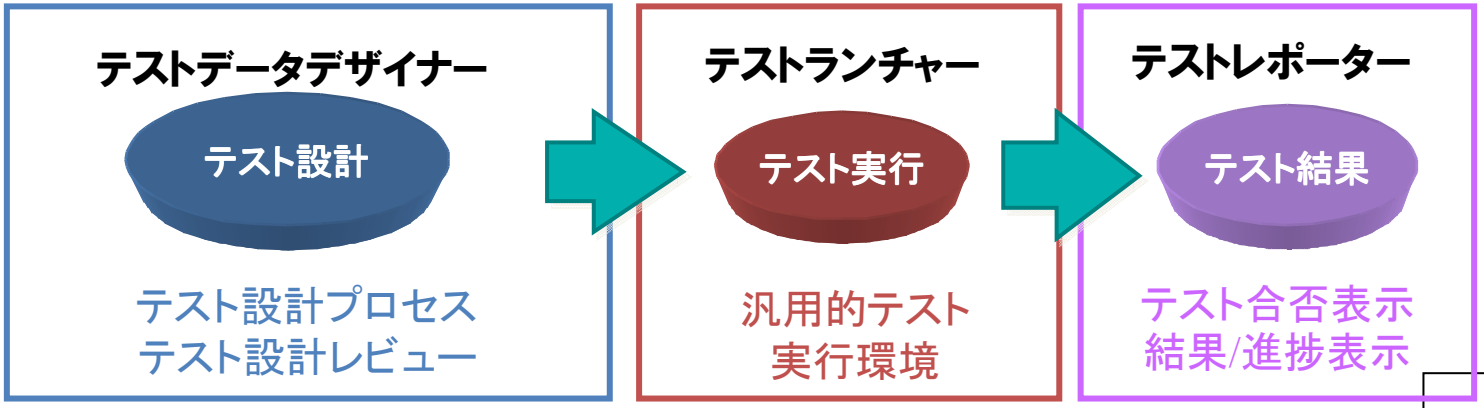


# ユニットマスター
# ツール構成・機能紹介

GAIO TECHNOLOGY

# ユニットマスターは3つの機能を個別に提供

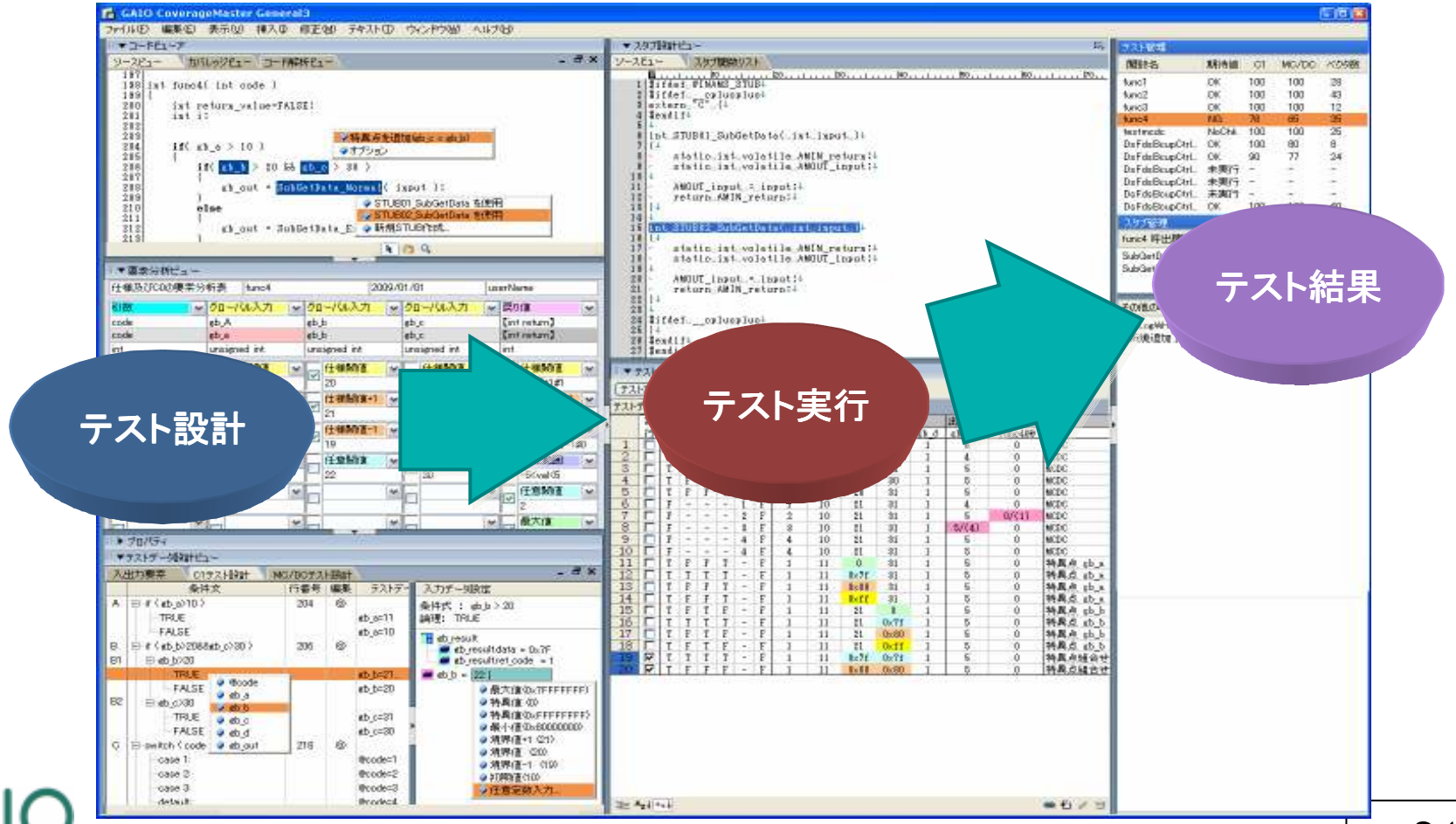
- C/C++コードの評価に対応した単体テストスイート
- 設計・実行・結果の3つの工程を 個別のアプリケーションとして提供
- 個別に機能を導入可能
  - データ設計・作成機能（テストデータデザイナー）
    - 不具合検出目的のテスト設計プロセス
    - テスト設計レビュー
  - テスト実行（テストランチャー）
    - 汎用的（PCアプリケーション、シミュレータ、実機等）な環境におけるテスト実行環境
  - テスト結果（テストレポーター）
    - 単体テスト実行結果を、テスト実施者と管理者が閲覧/管理できるビューを搭載



# ユニットマスター　画面イメージ図

## ■ 統合化された環境で効率的な単体テスト作業が可能

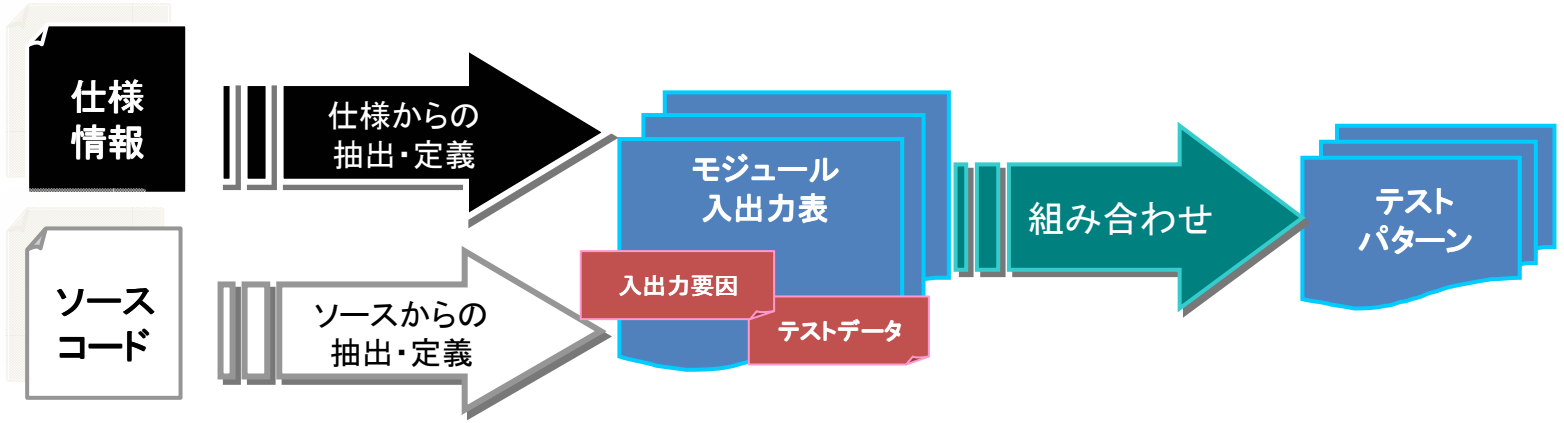
- 組込み以外のユーザー層のユースケースに合わせたGUI構成に



# テストデータデザイナーとは

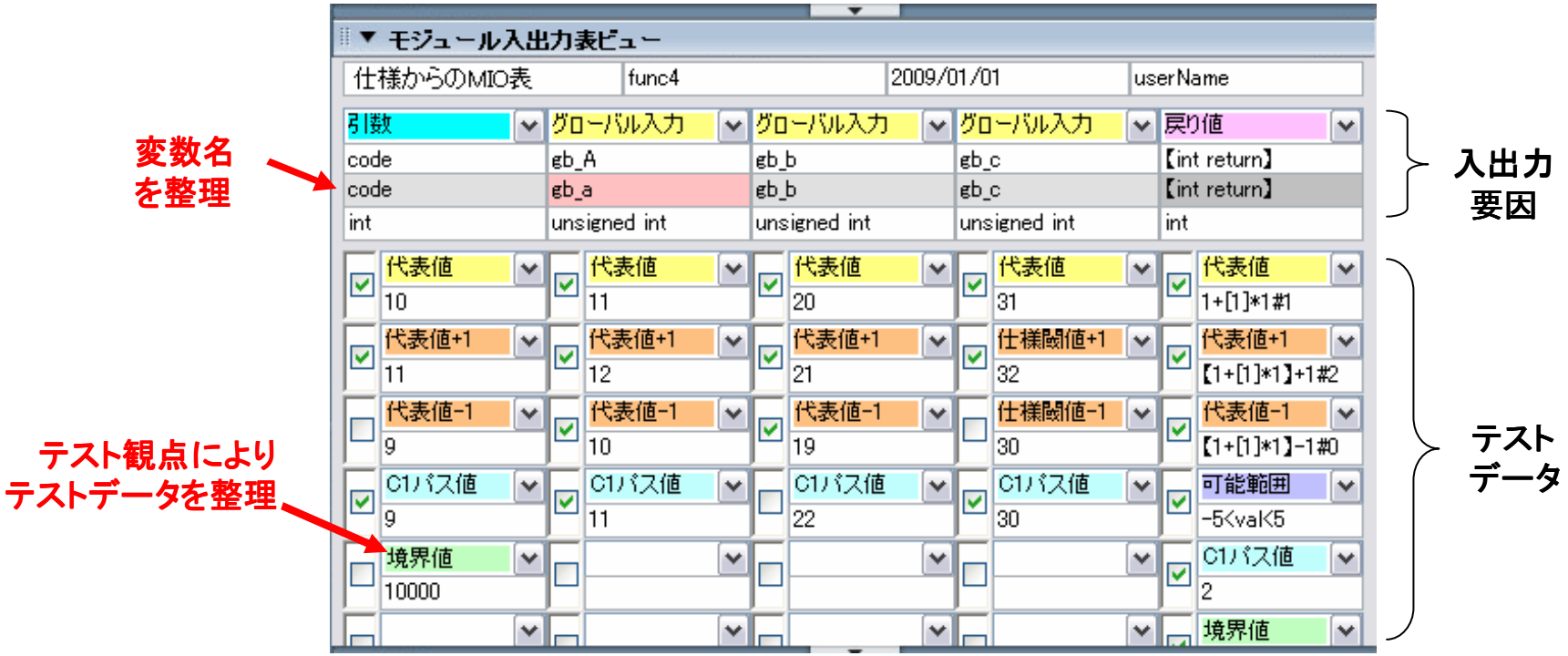
- **開発者スキルに依存しない標準化されたテストデータ作成支援アプリ**
  - 単体テストの品質を明確にする「モジュール入出力表（要素分析表）」を利用
  - テスト設計過程を記録し、客観的なテストデータ設計を実現

- **不具合検出のためのテスト観点を整理しデータ化**
  - 仕様書情報からブラックボックス観点のテストデータを整理し設定
  - ソースコードからホワイトボックス観点のテストデータを整理し設定
  - 組合せパターン（テストベクター）の生成と網羅性管理



# テストデータデザイナー：モジュール入出力表のイメージ

- モジュール（関数）への入出力変数に与えるテストデータの管理
  - 変数毎にテストの観点とテストデータを整理して設定
    - ― 境界条件値、最大最小値、オーバーフロー要因など、その他不具合の要因
- テストの品質を客観的に示すエビデンスとなる

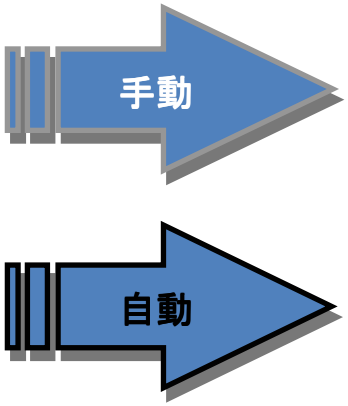


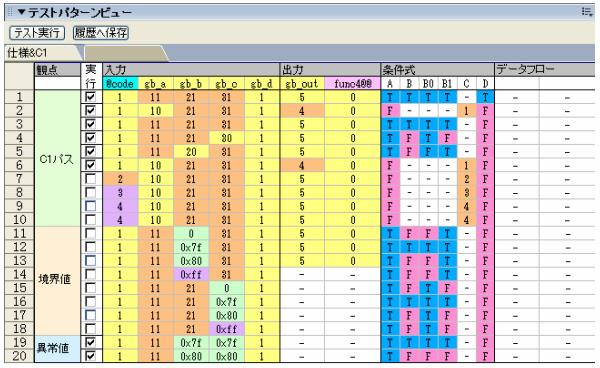
# テストデータデザイナー：テストパターンの作成

## ■ モジュール入出力表で設計したテストデータからテストパターンを生成

- 総組み合わせ
- デシジョンテーブル
- 直交表
- 最大値/最小値におけるアンダー/オーバフローの確認データ
- カバレッジ網羅データ

モジュール入出力表



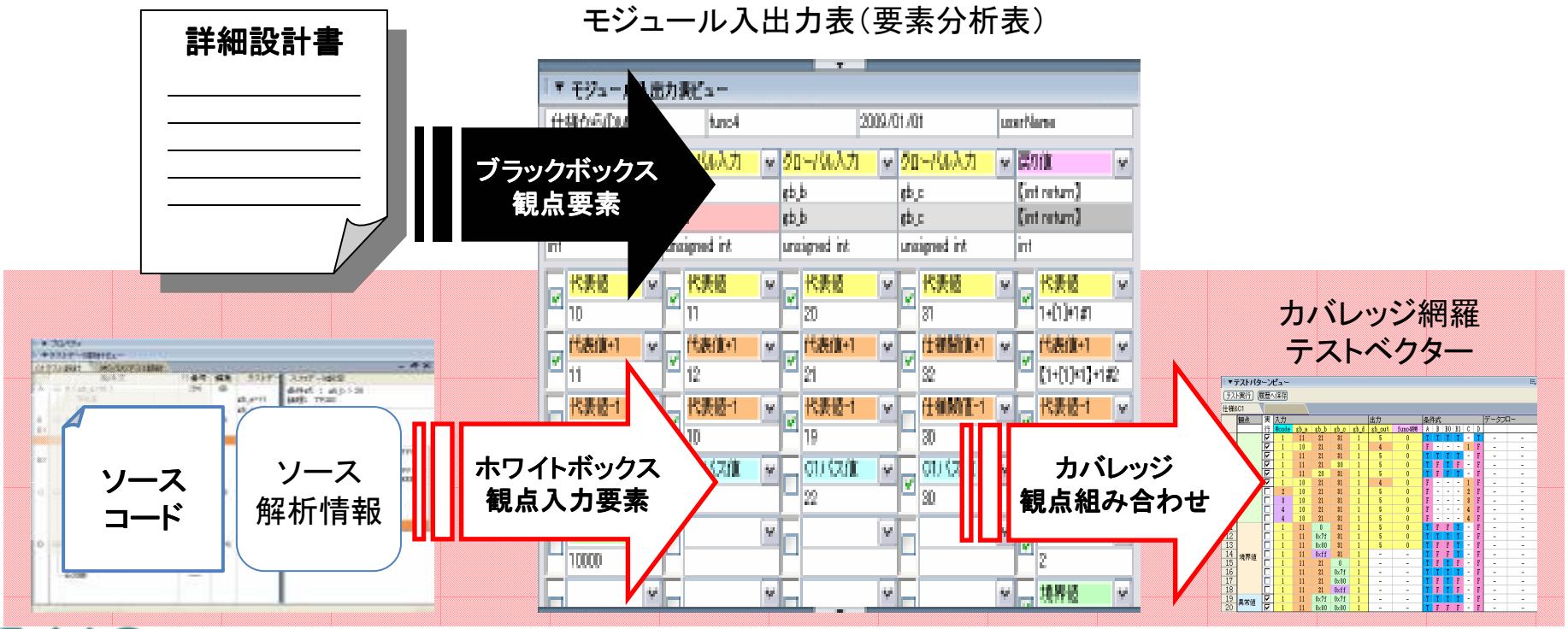



テストパターン

# テストデータデザイナー：カバレッジ網羅データの作成支援

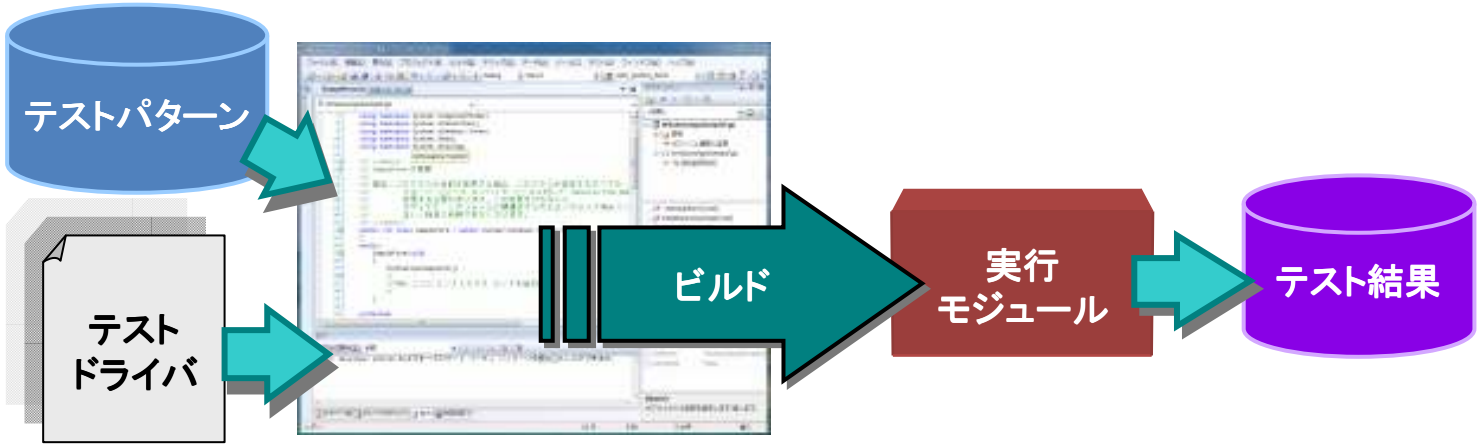
## ■ ソースコードの解析による モジュール入出力表作成の支援機能サポート

- モジュールの入出力要因を自動抽出
- ソースコードからテストデータをリバース作成
- ソースパス網羅（カバレッジ）に着目したテストデータ作成を支援



# テストランチャーとは

- **組込みプラットフォームに依存しない単体テスト実行を実現**
  - ネイティブアプリ、WindowsCEエミュレーション、
    カバレッジマスター等実機コード などのテスト実行環境を汎用的にサポート
  - ソースコードベースのテスト環境を実現
  - Visual Studioなどの開発プロジェクトからのビルド情報を取得
- **テストドライバ自動作成機能**
  - ソースコードへのカバレッジ計測用等フックコードの自動埋め込み
  - 実行モジュールを作成




GAIO TECHNOLOGY

# テストレポーターとは

## ■ テスト実施者、テスト管理者個別にテスト結果、進捗を確認する機能

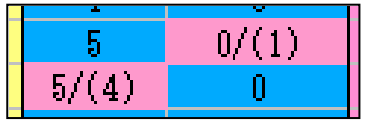
- 出力結果の合否判定
  - ― 期待値と出力値をチェックして合否結果表示
  - ― 様々なアウトプット形式
- HTML, SQLDB, XML, テキスト形式で出力可能
  - ― 差分結果の表示
  - ― 過去のテスト結果との差分表示

▼テストパターンビュー

テスト実行　履歴へ保存

仕様&C1

| | 観点 | 実行 | 入力 | | | | | 出力 | | 条件式 | | | | | | データフロー | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | @code | gb_a | gb_b | gb_c | gb_d | gb_out | func400 | A | B | B0 | B1 | C | D | | |
| 1 | C1パス | ☑ | 1 | 11 | 21 | 31 | 1 | | | T | T | T | T | - | T | - | - |
| 2 | | ☑ | 1 | 10 | 21 | 31 | 1 | 4 | 0 | F | - | - | - | 1 | F | - | - |
| 3 | | ☑ | 1 | 11 | 21 | 31 | 1 | 5 | 0 | T | | | | | | | |
| 4 | | ☑ | 1 | 11 | 21 | 30 | 1 | 5 | 0 | T | F | T | F | - | F | - | - |
| 5 | | ☑ | 1 | 11 | 20 | 31 | 1 | 5 | 0 | T | F | F | T | - | F | - | - |
| 6 | | ☑ | 1 | 10 | 21 | 31 | 1 | 4 | 0 | F | - | - | - | 1 | F | - | - |
| 7 | | ☐ | 2 | 10 | 21 | 31 | 1 | 5 | 0/(1) | F | - | - | - | 2 | F | ○ | ×(0) |
| 8 | | ☐ | 3 | 10 | 21 | 31 | 1 | 5/(4) | 0 | F | - | - | - | 3 | F | ×(255) | ○ |
| 9 | | ☐ | 4 | 10 | 21 | 31 | 1 | 5 | 0 | F | - | - | - | 4 | F | ○ | ○ |
| 10 | | ☐ | 4 | 10 | 21 | 31 | 1 | 5 | 0 | F | - | - | - | 4 | F | ○ | ○ |
| 11 | 境界値 | ☐ | 1 | 11 | 0 | 31 | 1 | | | T | F | F | T | - | F | - | - |
| 12 | | ☐ | 1 | 11 | 0x7f | 31 | 1 | 5 | 0 | T | T | T | T | - | F | - | - |
| 13 | | ☐ | 1 | 11 | 0x80 | 31 | 1 | 5 | 0 | T | F | F | T | - | F | - | - |
| 14 | | ☐ | 1 | 11 | 0xff | 31 | 1 | 5 | 0 | T | F | F | T | - | F | - | - |
| 15 | | ☐ | 1 | 11 | 21 | 0 | 1 | 5 | 0 | T | F | T | F | - | F | - | - |
| 16 | | ☐ | 1 | 11 | 21 | 0x7f | 1 | 5 | 0 | T | T | T | T | - | F | - | - |
| 17 | | ☐ | 1 | 11 | 21 | 0x80 | 1 | 5 | 0 | T | F | T | F | - | F | - | - |
| 18 | | ☐ | 1 | 11 | 21 | 0xff | 1 | 5 | 0 | T | F | T | F | - | F | - | - |
| 19 | 異常値 | ☑ | 1 | 11 | 0x7f | 0x7f | 1 | 5 | 0 | T | T | T | T | - | F | - | - |
| 20 | | ☑ | 1 | 11 | 0x80 | 0x80 | 1 | 5 | 0 | T | F | F | F | - | F | - | - |

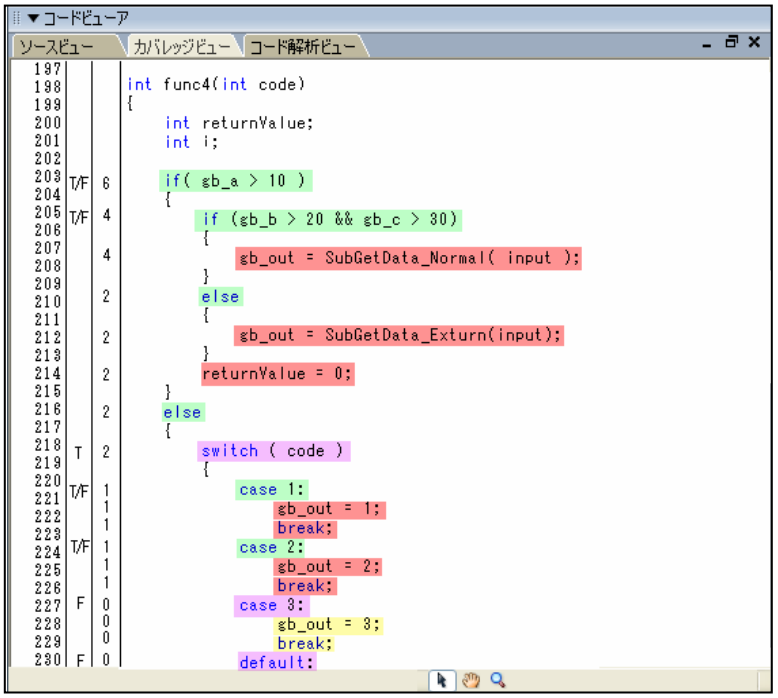



期待値「5」に対して
出力値が「4」の為
エラーの結果が表示されている

# テストレポーター：カバレッジ結果表示

## ■ テストランチャーで取得した カバレッジ結果の表示

- C0,C1,MC/DCのカバレッジ結果の表示
- 複数モジュールのカバレッジ結果の表示も可能




テスト管理

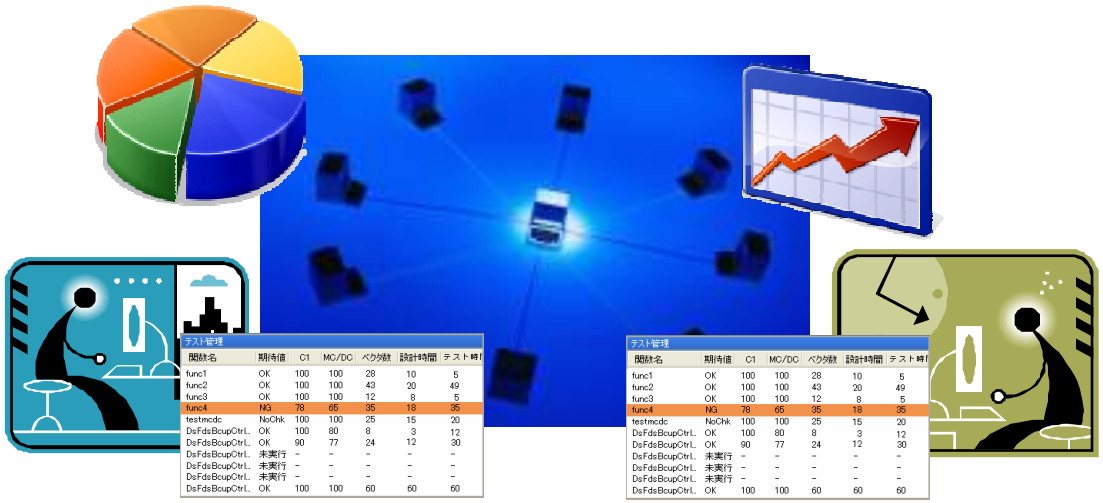
| 関数名 | 期待値 | C1 | MC/DC | ベクタ数 | 設計時間 | テスト時 |
|---|---|---|---|---|---|---|
| func1 | OK | 100 | 100 | 28 | 10 | 5 |
| func2 | OK | 100 | 100 | 43 | 20 | 49 |
| func3 | OK | 100 | 100 | 12 | 8 | 5 |
| func4 | NG | 78 | 65 | 35 | 18 | 35 |
| testmcdc | NoChk | 100 | 100 | 25 | 15 | 20 |
| DsFdsBcupCtrl... | OK | 100 | 80 | 8 | 3 | 12 |
| DsFdsBcupCtrl... | OK | 90 | 77 | 24 | 12 | 30 |
| DsFdsBcupCtrl... | 未実行 | - | - | - | - | - |
| DsFdsBcupCtrl... | 未実行 | - | - | - | - | - |
| DsFdsBcupCtrl... | 未実行 | - | - | - | - | - |
| DsFdsBcupCtrl... | OK | 100 | 100 | 60 | 60 | 60 |

GAIO TECHNOLOGY

# テストレポーター：管理者向け機能

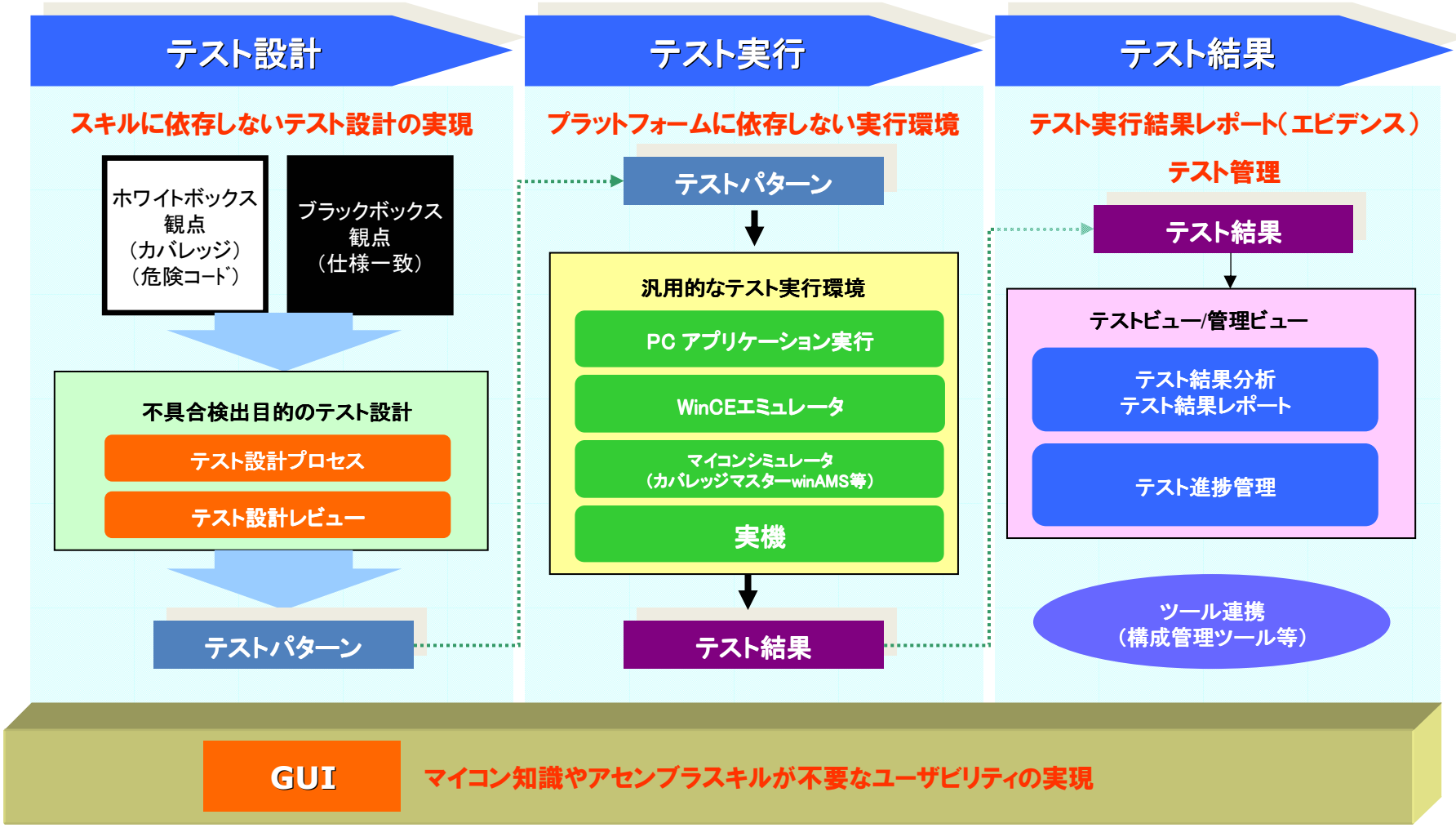
## ■ プロジェクト全体、各テスト実施者の進捗、テスト結果を集中管理

- リレーショナルDBをサポート
  - ― 複数の担当者が作成した情報を1箇所で集約し
  - ― プロジェクト全体の管理業務をサポート可能
- 全ての情報をRDBへ送信、プロジェクト単位で保存
  - ― 進捗管理やテスト結果の収集に利用可能
  - ― 構成管理ツールでの管理が可能




GAIO TECHNOLOGY

# ユニットマスターの全体図　まとめ



# END

## 最新の製品情報は
## http://www.gaio.co.jp/


ガイオ・テクノロジー株式会社



ガイオ・テクノロジー株式会社
日本橋事業所　営業部
〒103　東京都中央区日本橋人形町3-12-8

TEL.(03)3662-3041
FAX.(03)3662-3043
Email　info@gaio.co.jp　・・ご質問はこちらにお願いします。

GAIO TECHNOLOGY