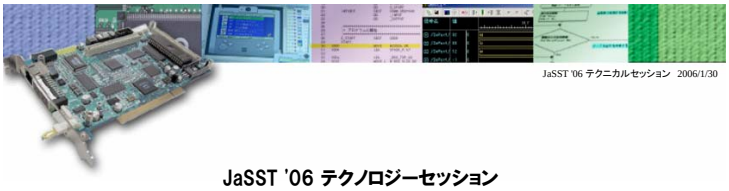


JaSST '06 テクノロジーセッション

カバレッジマスター winAMS

マイコンコードで組み込みソフト単体自動テストを実現

ガイオ・テクノロジー株式会社
テクニカルサポート マネージャ
髙橋 圭一

組み込みソフト品質向上への課題

組み込みソフト品質向上への課題

- **大規模化する組み込みソフトの品質保証**
  - 増大するソースコード量に対して 効率よくテストする方法は？
  - 不具合の原因であるテスト不足をどのように改善するか？
- **テスト不足になりがちなソフト検証プロセス**
  - ソフトの設計フェーズ毎に、体系立てたソフト検証が必要
  - 手作業によるデバッグ工程では、実行可能なテスト量に限界がある
- **過去のソースコード資産の信頼性**
  - 組み込み開発は、過去のソース資産の積み重ねが多い
  - 設計者不在の過去の資産の品質を 如何に保証するか
- **結合テストだけでは網羅できない例外条件の検証**
  - 正常系だけでなく、異常系、例外系の条件を、如何にテストするか

ソフト品質向上への解決策

- **ソフトウエアへのテストの密度（網羅率）を上げる**
  - （理想的には）ソフトウエア条件の全てのパターンをテストする
  - 開発途中でのレベルダウンを起こさない 信頼性の高い管理
- **テスト工程の自動化で テスト時間の短縮を行う**
  - 人海戦術による手作業デバッグから脱却する
  - 自動テストのメカニズム、手法を取り入れる
- **結合試験の前（上流工程）で 潜在バグを無くす**
  - ハードウエアでの結合テストでは発見できない潜在バグに対応

- **新しい検証・テストツールなど**
  既存開発環境以外の即応性の高い開発手法を取り入れる

課題解決のキーワードの1つが「シミュレータ」

潜在バグを無くし 信頼性を向上する
モジュール単体テスト

モジュール単体テストとは

- **組み込みソフトの関数単位での入出力網羅テスト**
  - 関数単位で 想定される入力に対する出力を網羅テスト
  - フィルタリング計算など 数値演算ブロックの品質保証
- **コードパス、条件分岐に対する パスカバレッジ 網羅テスト**
  - カバレッジの指標 C0, C1, C2 保証テスト
    - C0: 命令カバレッジ → ソース行を少なくとも1回は実行
    - C1: 分岐カバレジ → コード内の全ての分岐を少なくとも1回は実行
    - C2: 条件カバレッジ → 分岐の条件を全て実行
    - MCDC: Modified Condition Decision Coverage

ガイオは「マイコンシミュレータ」を使用して
モジュール単体＆カバレッジ 自動テストを実現！

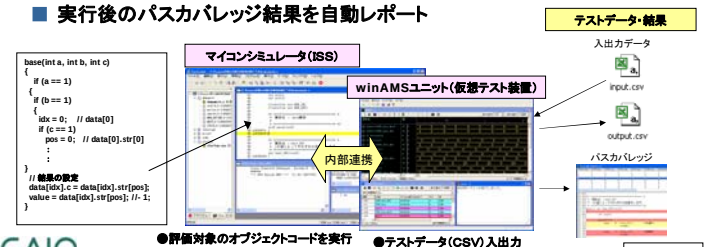
モジュール単体テストの効果

- **ソフト設計の上流工程での単体テストは 潜在バグを未然に防ぐ**
  - 上流工程での「ホワイトボックステスト」は、例外条件による潜在バグを発見しやすい
  - 後半でのシステムテストでは、「ブラックボックス」テストとなり、表面的な正常系動作のデバッグに偏るため、潜在的なバグは発見しにくい
- **関数設計者自身による「網羅率」を意識したテストは ソフト品質を向上する**
  - 関数のテストにおける設計者自身による「網羅率」を意識したテスト・デバッグは、「ソースコードレビュー」をより深く確実に行う切っ掛けとなる
- **テストの結果が、ソフト品質を示す「客観的・定量的」なデータとなる**
  - ソースコード上の網羅率は、客観性のある検証データとなる

マイコンコードで単体テストを自動化する カバレッジマスターwinAMS

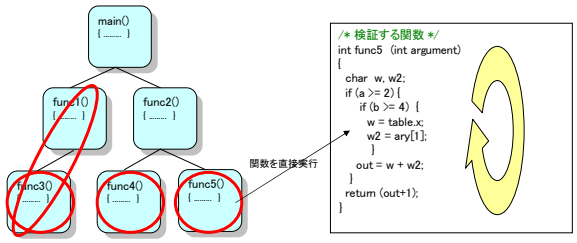
カバレッジマスター winAMS 概要

- **組み込みソフトの関数単体テストに特化した 検証システム**
- **マイコンシミュレータ(ISS)を動作エンジンに使用**
  - 実際のマイコンの組み込みオブジェクトコードで関数単体テストを実行
  - 対象の評価ソースコードの書き換えは 一切不要
- **テストデータ(関数、変数名)は全てCSVファイルで入出力**
- **実行後のパスカバレッジ結果を自動レポート**



組み込みソフトを関数毎に検証

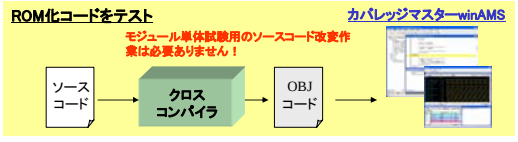
- **カバレッジマスターwinAMSの検証スタイルは関数単位**
  - 関数単独、関数ネストを含め検証可能
- **関数を直接実行、ネストの深い関数でも容易に検証可能**
- **時間 / タイミング要素は、検証の対象外**



マイコンの実コードを使用して検証

- **ソース修正なしで単体テストが可能**
- **実装ROM(マイコン)コードを使用して モジュール単体テストを実行**
  - ークロスコンパイラで生成したオブジェクトをそのまま使用
  - ーNATIVE系のテストツールにありがちなコード修正作業は一切必要なし
    - ・クロスコンパイラ依存した予約語(# pragma xx等)
    - ・インラインアセンブラ
    - ・RTOS依存部分(各種システムCALL)

カバレッジマスターwinAMSではソースコード改変作業レス



入力データと検証結果

- **入力データ・期待値は、CSVファイルで設定・管理**
  - 評価対象の関数シンボル名
  - 入力変数名と入力データ
  - 出力変数名と期待値データ
- **入力データと検証結果は同じ形式**

評価する関数名 / 入力変数名 (関数名@ は引数) / 出力変数名 (@@関数名は戻り値)

mod	func	コメント	4	2	
input	TAB.mode	array[2]	func@a	func@@	output
1	1	-1	-1	1	2
12	43~	56~58	78	1	0x4
0x10	0x23	0x66	0x11		

入力データ / 期待値データ

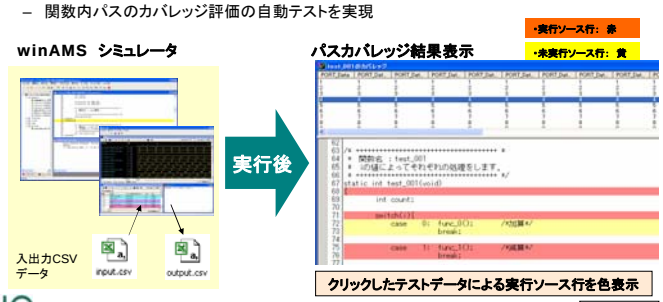
テスト関数サンプル

```
// 入力データ
char input;
struct {
    char mode;
    int flag;
} TAB;
char array[3] = { '1','2','3' };

// 結果データ
int output;
// 評価対象の関数 func
int func(int a)
{
    if (a == 1) {
        if (b == 1) {
        | 中略
    return 1;
}
```

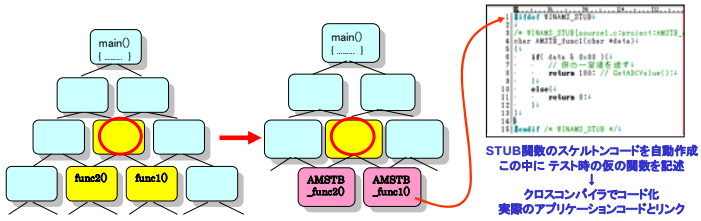
自動パスカバレッジ結果出力

■「カバレッジマスター winAMS」でパスカバレッジを自動化
- 入力データ毎に、実行パスを確認
- テスト実行後に 実行したソース行を色表示
- 関数内パスのカバレッジ評価の自動テストを実現



特長（1） 関数のSTUB機能

■ STUB関数作成と管理機能を装備
■ 元のソースコードを一切修正せず 単体自動テストが可能
- I/Oアクセス部分、RTOSシステムコール、永久ループ関数など、テスト環境では実行できない部分を代行関数へ置き換え
- 関数リスト上で STUBのON/OFFを簡単に切り替え可能
- 呼び出し側のソースは、一切修正不要



特長（2） ポインタ変数・引き数に対応

■ ポインタ変数・引数にも対応
- 変数の実体を伴わないポインタ変数・引数の場合でもそのままテスト可能
- 入力データCSVの変数名に「$」を付けるだけで メモリを動的割り当て
- 割り付けエリアは MPUのメモリモデルから 空き領域を設定

$

```
int func1( char *data )
```

```
int main()
{
    p = malloc (100);    // メモリをアロケーション
    func1 (p);
}
```

カバレッジマスターwinAMSで検証する関数

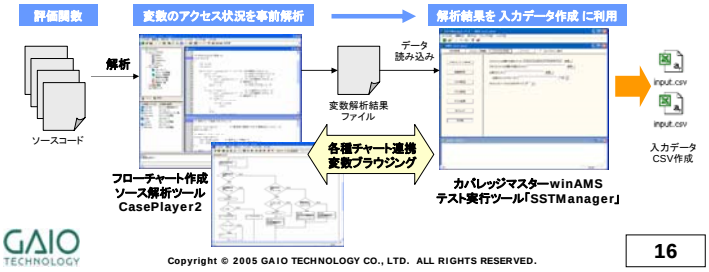
```
int func1 ( char *data )
{
    func2 (data);
}
```

	1	2	3
1	mod	func1	project_name
2	$func1 @data	out1	out2
3	1		
4	2		

ポインタ変数の実体を作るための割り当てエリアを指定

特徴（3） テストデータ作成支援

■ ソース解析ツール「CasePlayer2」と連携して テストデータ作成を支援
■ テスト対象の関数が使用する変数を事前に解析
■ テストデータ作成の効率を向上
- 条件分岐境界値（しきい値）を自動リストアップ
- 複数の入力変数データの組み合わせパターンの作成機能



導入事例

＜フェリカネットワークス様＞

フェリカネットワークス様 導入事例

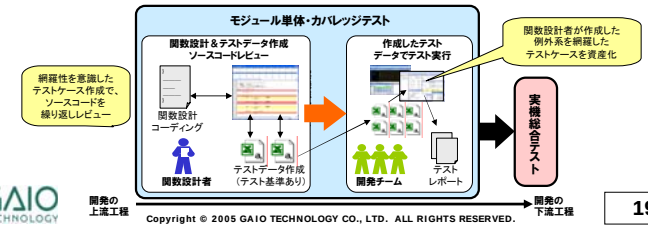
FeliCa Networks

■ 携帯電話向け組込み用 モバイルFeliCaチップ ソフト開発に利用
- 非接触ICカード技術、電子決済システムに利用

■ ガイオカバレッジマスターの選択ポイント
- 携帯電話向けの厳しい品質要求には 実機に最も近いテスト環境（ISS）がマッチ
- 多数のマイコンへの対応（マルチプラットフォーム）、環境の切換が容易であること
- 作成したテストデータ資産が 他のマイコンでの単体テストにもそのまま使用できること

フェリカネットワークス様 テスト方法

- **関数設計者自身による単体テスト実施で ソースコードを反復レビュー**
  - 「網羅性」を意識したテストケース作りで ソースの潜在バグを洗い出し
  - パスカバレッジビュー で、分岐パスを意識した ソースの見直し実施
- **開発の上流で「単体テスト」を徹底し、全体の検証プロセスを改善**
  - 結合テストでは発見できないバグを 上流の「単体テスト」で補間
- **テストデータ作成基準を設定し 担当者間の個人差を縮小**
  - 例えば、全ての分岐条件に「最大値、最小値、しきい値」の3つを必ず指定 など



フェリカネットワークス様 導入効果

- **開発プロジェクトの評価対象ソースコード**
  - マイコン ： RISC系／CSIC系 複数
  - オブジェクトコードサイズ： 約50kバイト
  - 単体評価関数の数： 850関数
  - テストケース数： 7500ケース（約9ケース/関数）
    - if文、switch文等の分岐条件に着目
- **テストスケジュール（2段階評価）**
  - 事前試行テスト： フルタイム5名 x 5日間
    - STUB設定などの基準を作成する目的で事前実行
  - テストデータ整備 ＋ 一斉テスト： 7名 x 15日間
- **テスト実行結果**
  - C0カバレッジ 100%（実行出来ないdefaultケース0.1%を除く）
  - 不具合件数 90件（開発レビュー時のものも含む）
- **不具合箇所の原因などの分析結果**
  - ほとんどの不具合は例外系で、結合テストでは発見できない可能性が高い

フェリカネットワークス（株）開発部の皆様

END

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