PSE support Center Test

SIEMENS

B2 – Dynamic testing and test- case design

PSE support Center Test

SIEMENS

Overview

- Problem
- Classification
- Black box technique
- White box technique
- Intuitive TC-identification
- Summary

PSE support Center Test

Dynamic Testing

SIEMENS

Dynamic testing =

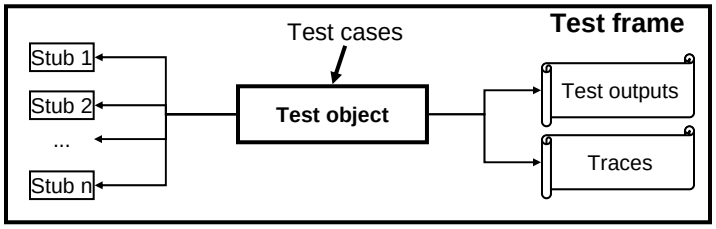
Testing by executing test objects

Prerequisites:

an executable program

Component testing:

No executable program so far, therefore:
Test frame, test driver (test bed)
Placeholder (stub, mock object)



PSE support Center Test

Procedure for test case determination

SIEMENS

- Classification in accordance with what is known of the test object:
  - Black-box methods:
    - Do not make use of any knowledge of internal structure of the test object
    - Designated also as “functional testing technique“
  - White-box methods
    - Make use of the knowledge of internal structure
    - Designated also as “structural testing technique"
- Classification in accordance with the procedure
  - Methodical identification of test cases
    - Based on defined methods (see above)
  - Intuitive identification of test cases
    - Makes use of the experiences of the testers

PSEsupportCenter Test

Classification of test techniques

SIEMENS

- Black box technique
  - Establishment of equivalence partitions
  - Boundary value analysis
  - State transition testing
  - Cause-effect analysis
  - Classification tree technique
- White box technique
  - Statement coverage
  - Branch coverage
  - Test of conditions
  - Path coverage

PSEsupportCenter Test

Black box technique - general

SIEMENS

- Internal set-up of test object unknown
  - ==> test cases derived only from the specification
- Test of all possible input combinations
  - impossible even theoretically
  - number of test cases has to be minimized
- Possibility for minimizing:
  - equivalence partitions
  - boundary values
- State-transition testing
  - test-object behavior depends not only on inputs
  - ==> internal states have to be taken into account

PSE support Center Test

Equivalence partitions

SIEMENS

Definition of “Equivalence partition":

- A contiguous value range per input variable
- Test data of one equivalence partition cause equal ouputs or execute equal program functions

Procedure:

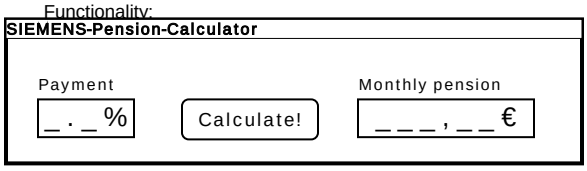
1. Determine definition ranges per variable
2. Refine equivalence partitions
3. Implement positive test cases
4. Implement negative test cases

PSE support Center Test

Equivalence partition - example

SIEMENS

Functionality:



Description:

Input: Payment in percent of the gross salary.
Output: Pension according to works agreement of Siemens.

Conditions:

Percentages have to be between 0.0 and 5.0

Expected output:

0.0	No pension
0.1 - 2.0	Calculation based on model A of the agreement
2.1 - 5.0	Calculation based on model B of the agreement

PSE support Center Test

Application of the method

SIEMENS

1.Determination of definition ranges per variable

Input variable:	Percentage
Definition range:	Decimal number
Possible values:	0.0 to 9.0
Invalid values:	Values > 5.0 and other characters

2. Refining of equivalence partitions

Equivalence partitions derive from the expected outputs:
- EP1: 0.0
- EP2: 0.1 - 2.0
- EP3: 2.1 - 5.0

Invalid equivalence partitions:
- IP1: 5.1 - 9.9
- IP2: "a,b", and similar invalid characters

3. Implementation of positive test cases

3 test cases in accordance with the 3 equivalence partitions
e.g.: with representatives "0.0", "1.1", 4.2"

4. Implementation of negative test cases

2 test cases, in accordance with 2 equivalence partitions
e.g.: with representatives "6,7", "a,b"

PSE support Center Test

Boundary value method

SIEMENS

- Weak point of equivalence partitioning method:
  - Typical values work in most cases
  - Errors occur frequently at the boundaries of equivalence partitions
- Procedure for the boundary value method:
  - As with equivalence partitioning
  - However option to choose at the boundaries of EP
    - Value exactly on the boundary
    - Value above/below the boundary
- Example "Pension calculator"
  - Positive test cases: "0.0", "0.1", "2.0", "2.1", "5.0"
  - Negative test cases: "5.1", "a,b"
    - "5.1" is a boundary value resulting in a negative test case!

PSE support Center Test

State-transition testing

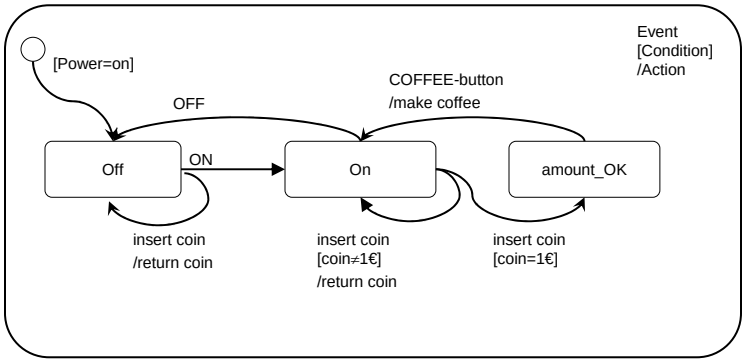
SIEMENS

System behavior

not dependent on inputs only

==> internal states

Example “coffee machine"



PSE support Center Test

Test case design

SIEMENS

- Different degrees of coverage:
  - each state reached at least once
  - each edge passed at least once
  - each path passed once
  - testing for rejection of prohibited events
- In the example:
  - Coverage of all nodes and edges:
    - **TC1: button ON - Insert 1€ - key COFFEE - key OFF**
    - **TC2: button ON - Insert 2€ - key OFF**
    - **TC3: insert 1€**
  - Rejection of prohibited events
    - TCP1: button ON - button COFFEE
    - TCP2: button ON - insert 2€ - button COFFEE
    - TCP3: button ON - insert 1€ - key OFF (!)

PSE support Center Test

White box technique - general

SIEMENS

- Internal set-up of the test object is known
- Basic idea:
  - execute all program parts at least once

PSE support Center Test

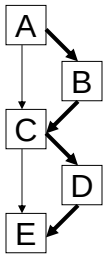
C0 - coverage

SIEMENS

Each statement is executed once

```
void CoverMe (int a, int b)
{
    printf("A");
    if (a < 1)
        printf("B");
    printf("C");
    if (b < 2)
        printf("D");
    printf("E");
}
```

a=0, b=1 => ABCDE



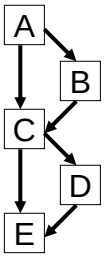
PSE support Center Test

C1 - Coverage

SIEMENS

Each branch is executed once

```
void CoverMe (int a, int b)
{
    printf("A");
    if (a < 1)
        printf("B");
    printf("C");
    if (b < 2)
        printf("D");
    printf("E");
}
```



a=0, b=1 => ABCDE

a=1, b=2 => ACE

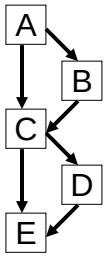
PSE support Center Test

C2 - coverage

SIEMENS

Each path is executed once

```
void CoverMe (int a, int b)
{
    printf("A");
    if (a < 1)
        printf("B");
    printf("C");
    if (b < 2)
        printf("D");
    printf("E");
}
```



a=0, b=1 => ABCDE

a=1, b=2 => ACE

a=0, b=2 => ABCE

a=1, b=1 => ACDE

PSE support Center Test

Condition coverage

SIEMENS

- Single condition coverage
  - Each sub-condition has to be at least true once and false once ((a>1) && (b<2)) requires 2 test cases
- Multiple condition coverage
  - Each combination of sub-conditions taken into account
  - ((a>1) && (b<2)) requires 4 test cases
- Minimum multiple condition coverage
  - Only the combinations of sub-conditions where the modification of the logical value of a condition changes the logical value of the entire condition.
  - Considerable savings effect as compared to the previous method
  - Nonetheless good error detection rate

PSE support Center Test

Minimum multiple condition coverage

SIEMENS

((a>1) && (b<2)) 3 test cases required

(a > 1)	(b < 2)	Total	Test case
F	**F**	**F ==> Test case**	**TC1**
F	T	F - nothing changed	-
T	**F**	**F ==> Test case**	**TC2**
T	**T**	**T** Result changed ==>	**TC3**

PSE support Center Test

Intuitive identification of test cases

SIEMENS

- Motivation:
  - Intuition of experienced testers
  - Ideas based on past errors
  - Assumptions about future errors
- Procedure:
  - Cannot be specified
  - Test cases derive from intuition, experience, assumptions, ...
- Use:
  - At all test levels
  - By no means as the only/primary procedure!
  - Only to complement the systematic methods!

PSE support Center Test

Summary

SIEMENS

- An executable program is the precondition for dynamic testing
- Classification of test techniques in accordance with the knowledge about the test object
- Combination of test methods often make sense (equivalence partitions, state-transition testing)
- C0, C1-coverage in component testing