SIEMENS

Corporate Technology

Module Testing with TTCN-3: Does it pay off?

Joachim Fröhlich

Siemens Corporate Technology
CT SE 12
Otto-Hahn-Ring 6
81739 München, Germany

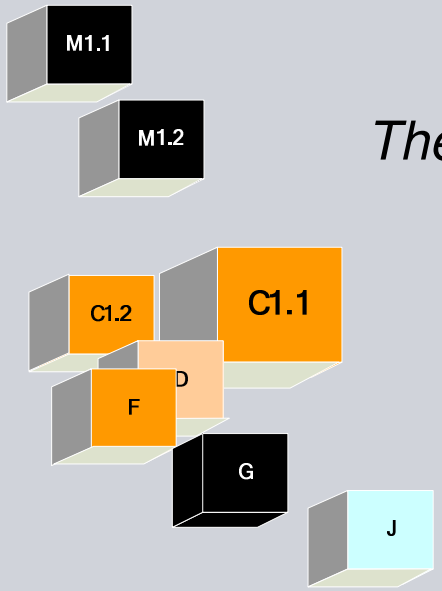
froehlich.joachim@siemens.com

6th TTCN-3 User Conference 2009
3 – 5 June 2009 – ETSI,
Sophia Antipolis, France

SIEMENS

Module testing with TTCN-3: does it pay off?

Can you expect anything new from this presentation?



The problems of TTCN-3 in this domain {module testing} are well known as a matter of fact.

The capabilities of TTCN-3 enable efficient ways of testing SW modules in a standardized way with standardized test system interfaces.

About the modules under test

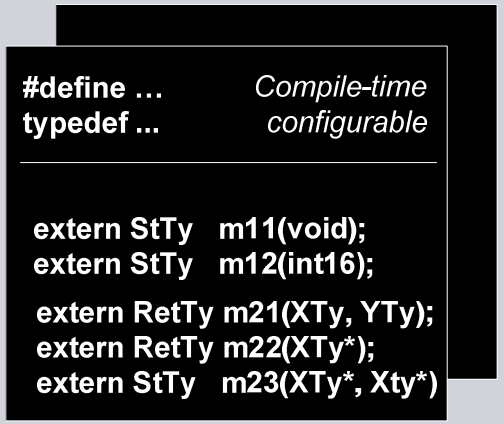
Module testing with TTCN-3: does it pay off?

SIEMENS

- Modules implemented in C90, MISRA
- Embedded platform software
- Interfaces: Operations, protocol
- Configurable: Data types, behavior, interfaces
- Module specifications evolve
- Comprehensive verification of the sequential behavior of module families

> 40 modules
> 600 operations
> 200 types
> 10k requirements
Several variants

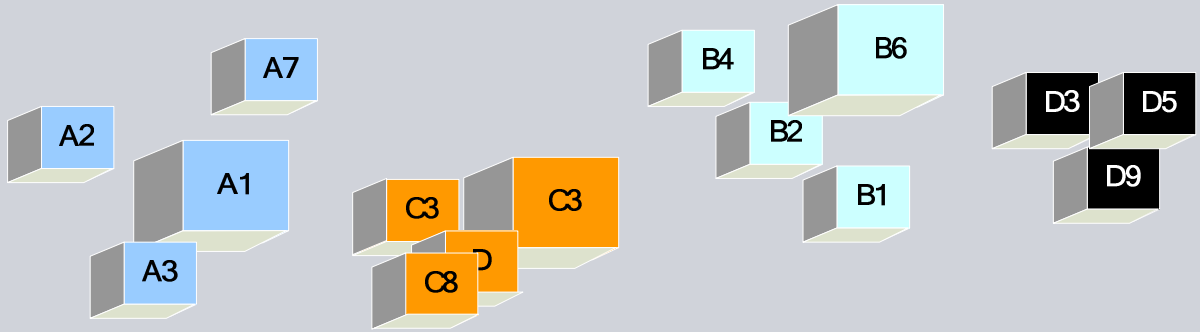
Multi-file C-module



SIEMENS

Module testing with TTCN-3: does it pay off?

How would you test embedded C-modules without TTCN-3?



TTCN-3 is not particularly the option for module testing but rather for integration and system testing.

Language complexity

Module testing with TTCN-3: does it pay off?

SIEMENS

- Tests shall find errors in the test object
- Simple, effective, efficient Tests, short-running
- Map no. of keywords to languages:
  - 74, 32, 130
  - TTCN-3 (v3.4.1), C90, ANSI C++ (C++ 98)

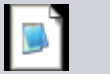
- TTCN-3 is a component-based, domain-specific language
- How does a varying C-module fit into TTCN-3's component world?

Testing approach: 1st iteration

Module testing with TTCN-3: does it pay off?

SIEMENS

- TTCN-3 components implement test doubles
- Procedure-based communication
- Test cases probe C-module variants for test points
- Test cases run to completion, always
- Final verdict calculated only once
- Custom verdict mechanism
- MISRA-C stylish test code
- Pointers? Treated elsewhere: Nyberg, 2006, LNCS 3964


Example1.ttcn

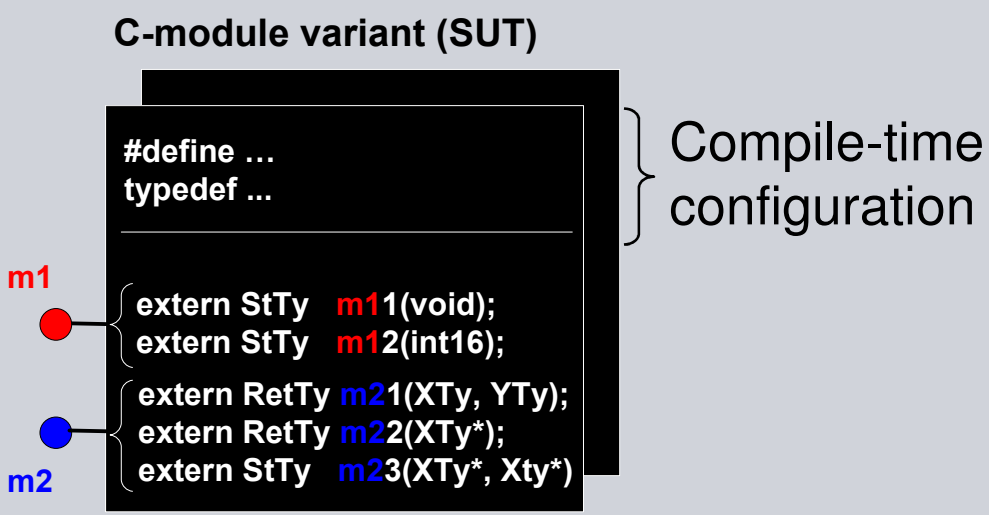

Example2.ttcn

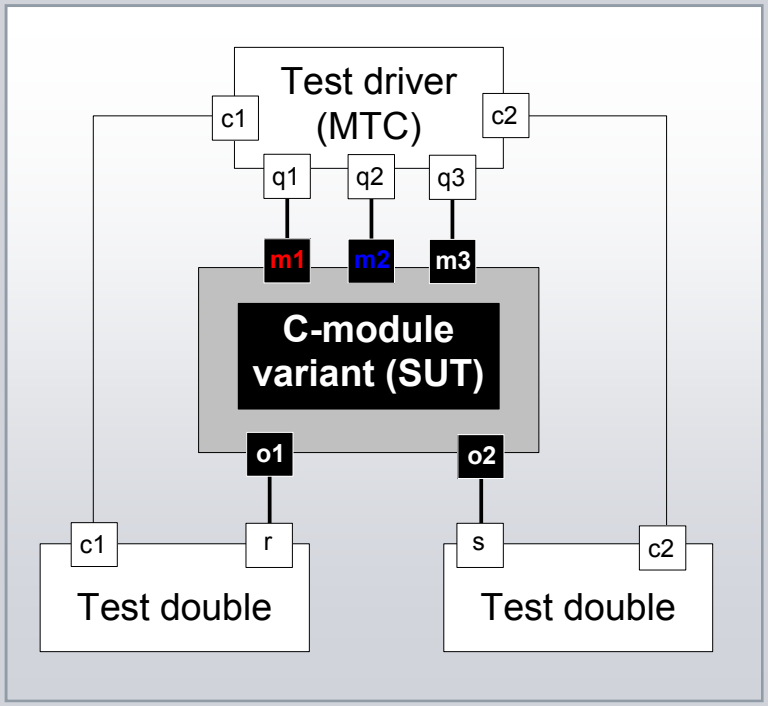
TTCN-3 components implement test doubles

Module testing with TTCN-3: does it pay off?

SIEMENS

- Parallel, distributed test components checking sequential behavior



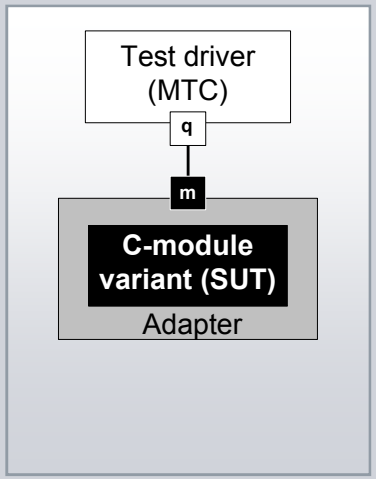


TTCN-3 components implement test doubles

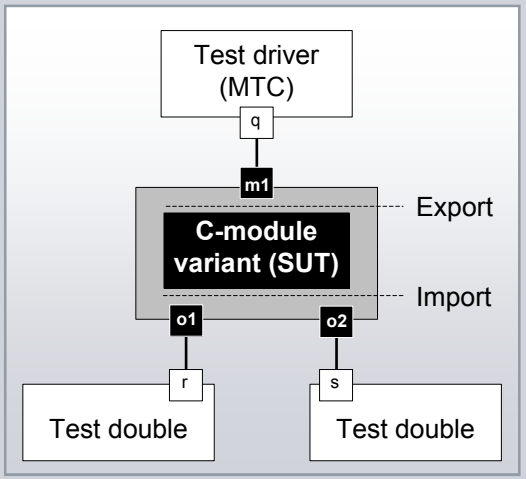
Module testing with TTCN-3: does it pay off?

SIEMENS

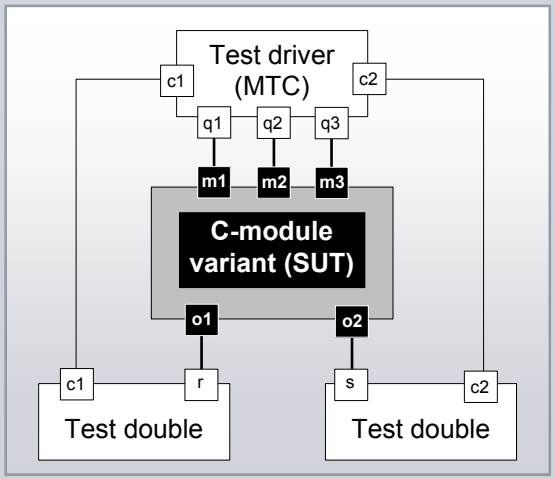
- C-module variant, 3 collaborators, sequential behavior
- Options for test system configurations:



(a): Self shunt



(b): Mock



(c): Mock

- Consider: Module variants, coordination, verdict calculation, test stop
- Processes / threads, resource needs

Tests probing for test points: A try

Module testing with TTCN-3: does it pay off?

SIEMENS

- Parameters describe C-module variants

```
01: import from ModuleDesc
02:   { function CheckX, MaxX … }
10: testcase TC1() runs on TestDriver {
20:   for (y := 0; y < MaxY(); y := y + 1) { // y groups x
21:     if (CheckY(y) == false) { setverdict(none); }
22:     else {
23:       for (x := 0; x < MaxY(); x := x + 1) {
24:         if (CheckX(x) == false) { setverdict(none); }
25:         else {
30:           // Test C-module at hot spot x, y
40:         }
41:       }
42:     }
43:   }
50:   stop;
60: }
```

Warning!

Other TCs similar

```
module ModuleDesc
{
  modulepar
  {
    ConfigTy CNF :=
    {
      __ := __,
      __ := __,
      __ :=
      {
        __ :=
        {
          ...
        },
        __ := omit,
      }
    }
  }
}
```

Test data similar

Tests probing for test points: Another try

Module testing with TTCN-3: does it pay off?

SIEMENS

- Data Selector moves N-dimensional cursor over an abstract aggregate

```
01: import from ModuleDesc { function Init, Select, Increment }
20: var FilterTy f := <<pseudo code: filter for selecting the test points>>;
21: var IndexTy  x, y; // x grouped by y
22: Init(x, y);
23: while (Select(f, x, y)) { // 1D filter (f), 2D cursor (x, y)
30:   // Test C-module at hot spot x, y
40:   Increment(x, y);
41: }
```

Application pattern
Reusable, but not generic

Filter	Cursor	*Init*	*Select*	*Increment*
1D	1D	**inout** x	**in** f, **inout** x	**inout** x
2D, and	1D	**inout** x	**in** f1, **in** f2, **inout** x	**inout** x
2D, or	1D	**inout** x	**in** f1, **in** f2, **inout** x	**inout** x

1D = 1-dimension: 1 filter criterion or 1 cursor (index)

Tests probing for hot spots: Another try

SIEMENS

- Data Selector can be applied in **`control`** or **`testcase`** parts

```
01: type integer IndexTy (0 .. infinity);

10: function Init(inout IndexTy pX, inout IndexTy pY) { pX := 0; pY := 0; }

20: function Select(in FilterTy pf, inout IndexTy pX, inout IndexTy pY)
21:   return boolean
22: {
23:   for (pY := pY; pY < cMaxY; pY := pY + 1) {
24:     if (CheckY(pY)) {
25:       for (pX := pX; pX < cMaxX; pX := pX + 1) { // x grouped by y
26:         if (CheckX(pF, pX)) { return true; } // Hot spot at x, y
27:       }
28:       pX := 0;  // Reset x, take next y
29:     }
30:   }
31:   return false; // No more hot spots
32: }

40: function Increment(inout IndexTy pX, inout IndexTy pY) { pY := pY + 1; }
```

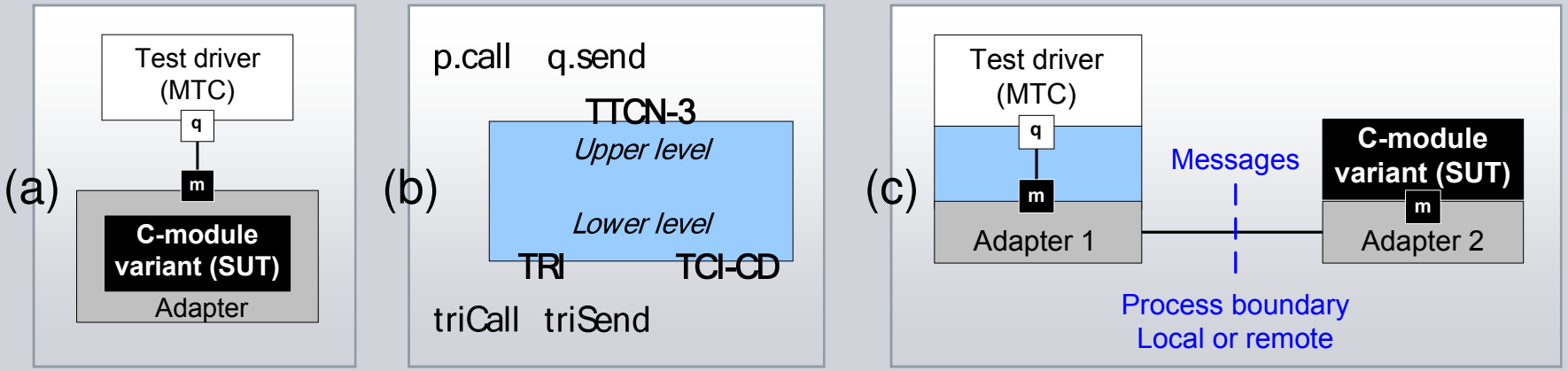
Implementation pattern
Exchangeable, but not generic

Linking C-modules and the test system

Module testing with TTCN-3: does it pay off?

SIEMENS

- Procedure-based communication: A natural choice?
- Use message-based communication if
  - Operation name is compile-time configurable
  - Expected/actual outputs shall be matched simply
  - C-module and test system run in separate processes
- Earli*er* binding (procedures) vs. lat*er* binding (messages)



Linking C-modules and the test system

Module testing with TTCN-3: does it pay off?

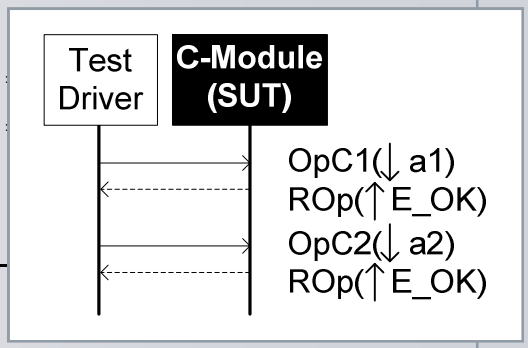
SIEMENS

- Call sequences and SUT (API) Encapsulation

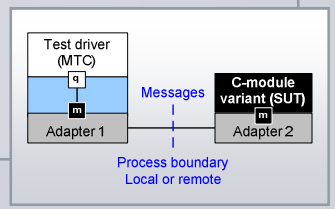
```
01: import from TestObject { function Call }
10: var template OpCallTy  OpC1 := { Id := cOp1, Par1 :=
11: var template OpCallTy  OpC2 := { Id := cOp1, Par1 :=
12: var template OpReplyTy OpR  := { ret := E_OK };

21: TestObject.Call(OpC1, OpR);
22: TestObject.Call(OpC2, OpR);

30: import from TestStrategy { function SetVerdict }
31: function Call(in template OpCallTy c, in template OpReplyTy r)
32:   runs on TestDriver
33: {
34:   q.send(c);
35:   alt {
36:     [] q.receive(r) { TestStrategy.SetVerdict(pass); }
37:     [] q.receive(*) { … }
38:   }
39: }
```



Implementation pattern not generic



Linking C-modules and the test system

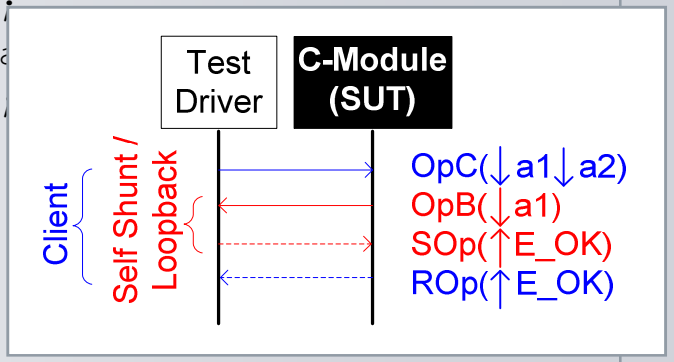
Module testing with TTCN-3: does it pay off?

SIEMENS

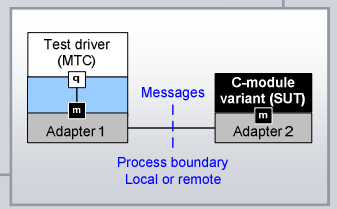
- Localizing tests with Self Shunts and SUT (API) Encapsulation

```
01: import from TestObject { function Call, GetReply, GetCall, Reply }
10: var template OpCallTy1 OpC := { Id := cOp1, Par1 := a1, Par2 := a2 };
11: var template OpReplyTy ROp := { ret := E_OK };
12: var template OpCallTy2 OpB := { Id := cOp2, Pa
13: var template OpReplyTy SOp := { ret := E_OK },
```

```
21: TestObject.Call(OpC);       // Client
22:   TestObject.GetCall(OpB); // Self Shunt
23:   TestObject.Reply(SOp);    // Self Shunt
24: TestObject.GetReply(ROp);   // Client
```




```
30: import from TestStrategy { function SetVerdict }
31: function Call(in template OpCallTy1 c) runs on TestDriver { q.send(c); }
32: function GetReply(in template OpReplyTy r) runs on TestDriver {
33:   alt {
34:     [] q.receive(r) { TestStrategy.SetVerdict(pass); }
35:     [] q.receive(*) { … }
36:   }
37: }
```



More points to consider

Module testing with TTCN-3: does it pay off?

SIEMENS

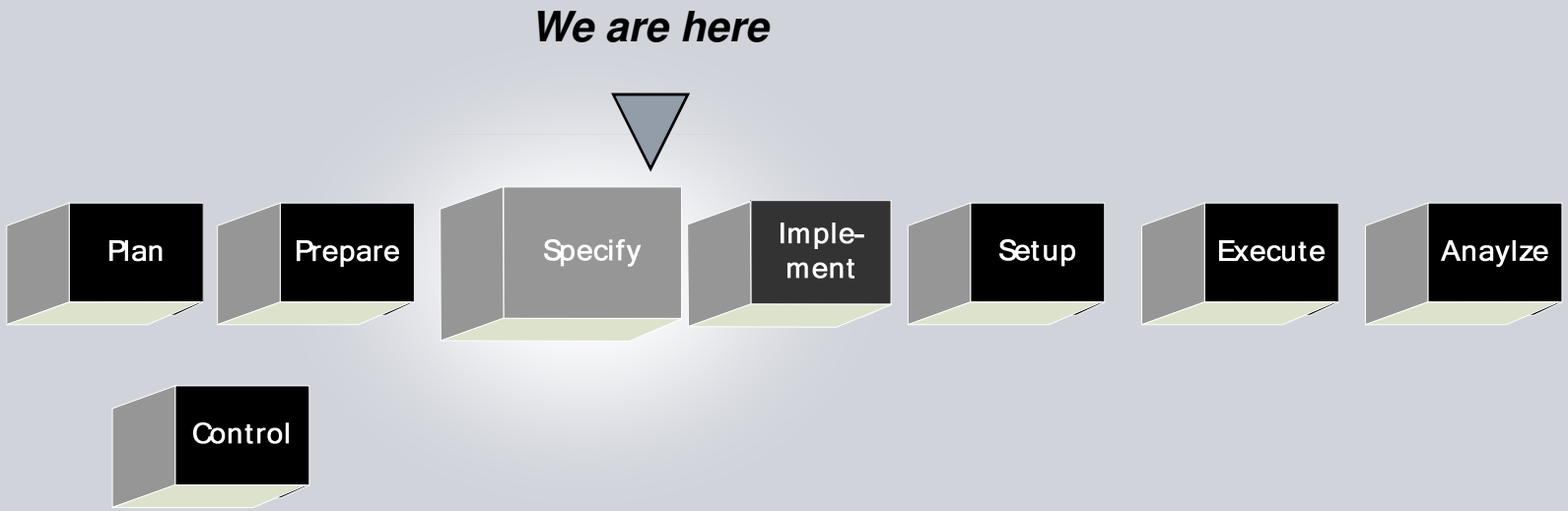
- Copy/paste/modify, parameters, generators, generics?
- C-module might call back various operations?

  … do the tests control the SUT?
- Varying type definitions / value ranges and type safety?
- Test reporting: What, when, where to log in TTCN-3, how?
- Pointers, messages, remote calls of configurable procedures
- Deployment does not matter?
- Folders and files do not matter?

SIEMENS

Module testing with TTCN-3: does it pay off?

Lost in translation?



Much more to do!

SIEMENS

Module testing with TTCN-3: does it pay off for …

… testing sequential, procedural behavior of C-module families?

Not yet

- TTCN-3 is a component-based language, including test concepts
- Complex and powerful, yet abstractions missing
- High up-front investments necessary

Will TTCN-3 pay off in the future?

It depends

- Tool chain, libraries (xUnit like), language improvements needed
- Test patterns (TUnit) and deployment patterns needed

mtc.stop