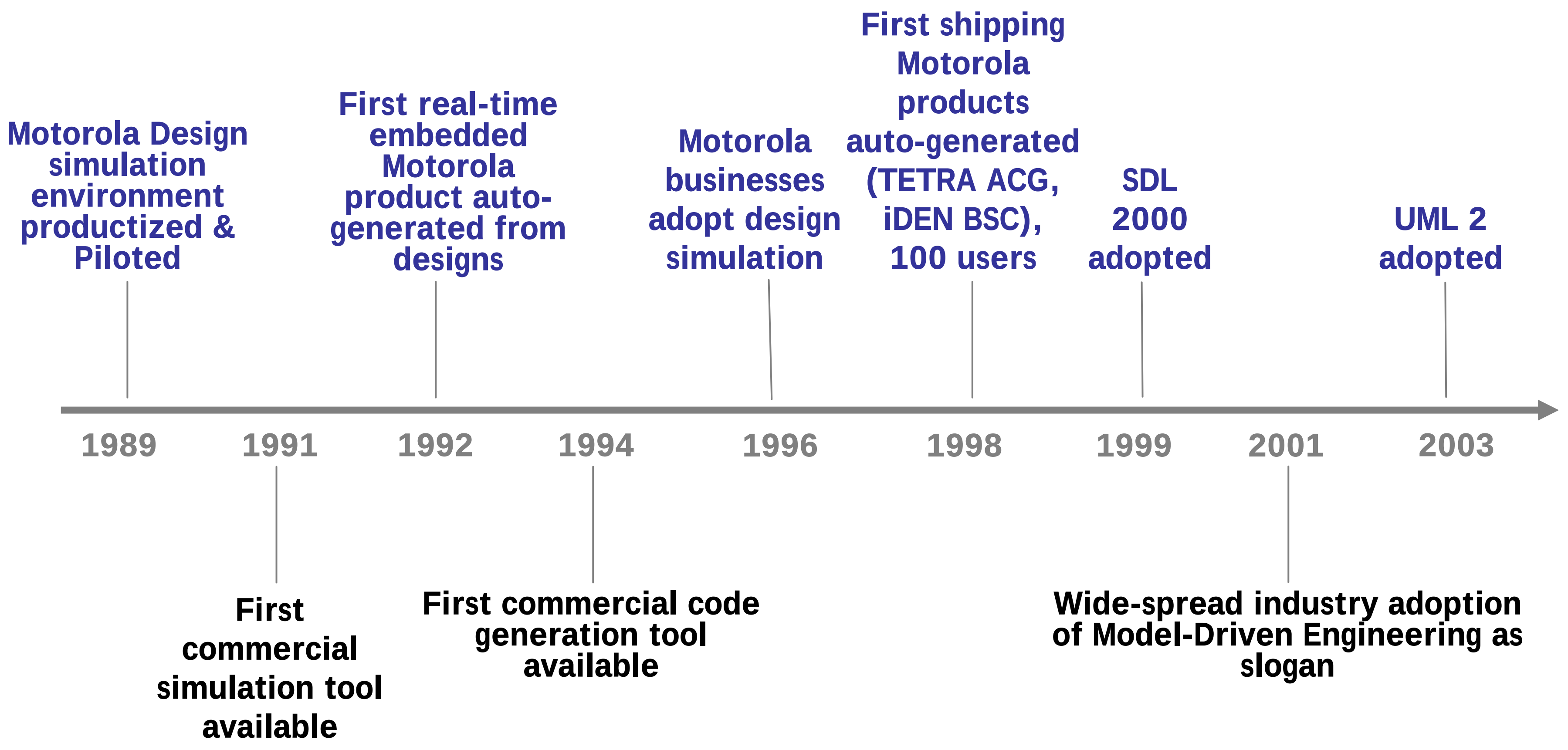
Early UML Model Testing using TTCN-3 and the UML Testing Profile

Paul Baker, Clive Jervis

Motorola - a Mature Model-Driven Engineering Company



1989

Motorola Design simulation environment productized & Piloted

1991

First commercial simulation tool available

1992

First real-time embedded Motorola product auto-generated from designs

1994

First commercial code generation tool available

1996

Motorola businesses adopt design simulation

1998

First shipping Motorola products auto-generated (TETRA ACG, iDEN BSC), 100 users

1999

SDL 2000 adopted

2001

Wide-spread industry adoption of Model-Driven Engineering as slogan

2003

UML 2 adopted

Model-Driven Engineering Impact

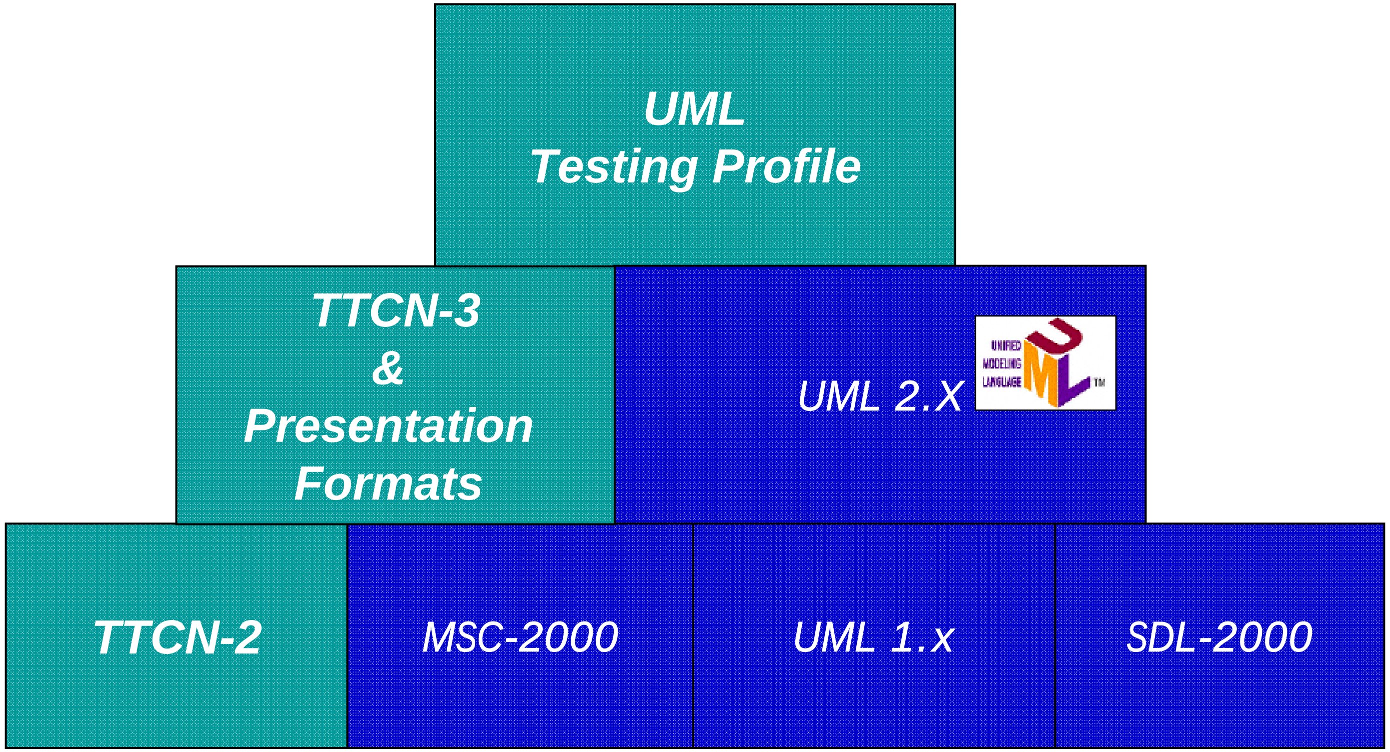
Quality

- 1.2 to 4X overall reduction in defects
- 3X improvement in phase containment of defects
- The overall Cost of Quality has also decreased due to decreased inspection and testing times

Productivity

- 2X to 8X productivity improvement - measured in terms of equivalent source lines of code

Standards Evolution



UML Testing Profile

TTCN-3 & Presentation Formats

UML 2.X

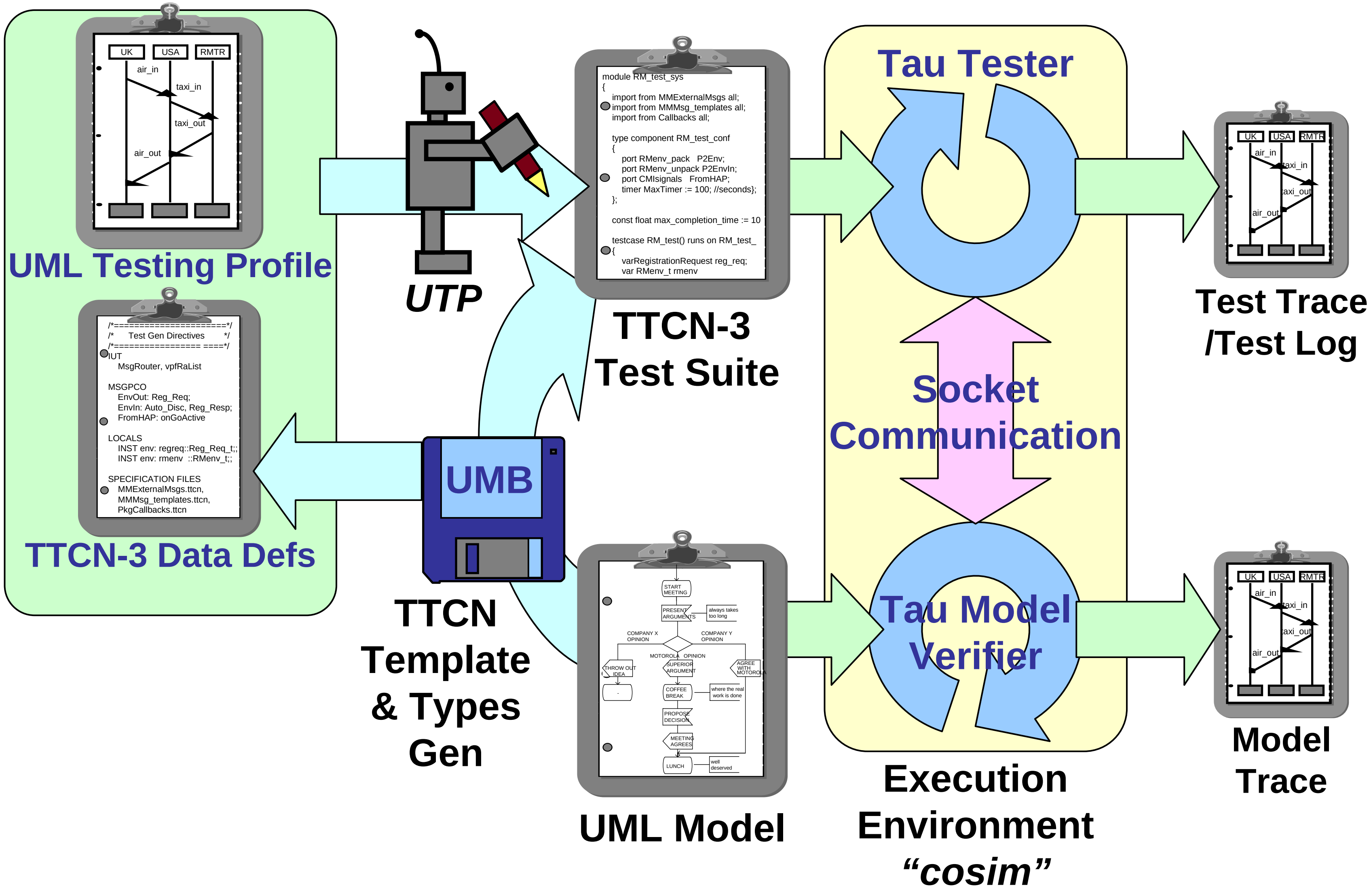
TTCN-2

MSC-2000

UML 1.x

SDL-2000

UML Model-Testing Environment



```
/*=====================*/
/*     Test Gen Directives      */
/*================== ====*/
IUT
    MsgRouter, vpfRaList

MSGPCO
    EnvOut: Reg_Req;
    EnvIn: Auto_Disc, Reg_Resp;
    FromHAP: onGoActive

LOCALS
    INST env: regreq::Reg_Req_t;;
    INST env: rmenv  ::RMenv_t;;

SPECIFICATION FILES
    MMExternalMsgs.ttcn,
    MMMsg_templates.ttcn,
    PkgCallbacks.ttcn
```

```
module RM_test_sys
{
   import from MMExternalMsgs all;
   import from MMMsg_templates all;
   import from Callbacks all;

   type component RM_test_conf
   {
      port RMenv_pack   P2Env;
      port RMenv_unpack P2EnvIn;
      port CMIsignals   FromHAP;
      timer MaxTimer := 100; //seconds};
   };

   const float max_completion_time := 10

   testcase RM_test() runs on RM_test_
   {
      varRegistrationRequest reg_req;
      var RMenv_t rmenv
```

Some Challenges

- **Development platform usually not target platform**
  - therefore no platform interface available.
  - how do we simulate platform functions?
  - including deliberately returning errors

- **Real-time aspects**
  - model testing not-testing real-time usually
  - therefore how to test timer related functionality?
  - e.g. force timer to time out, or to not have timer expire

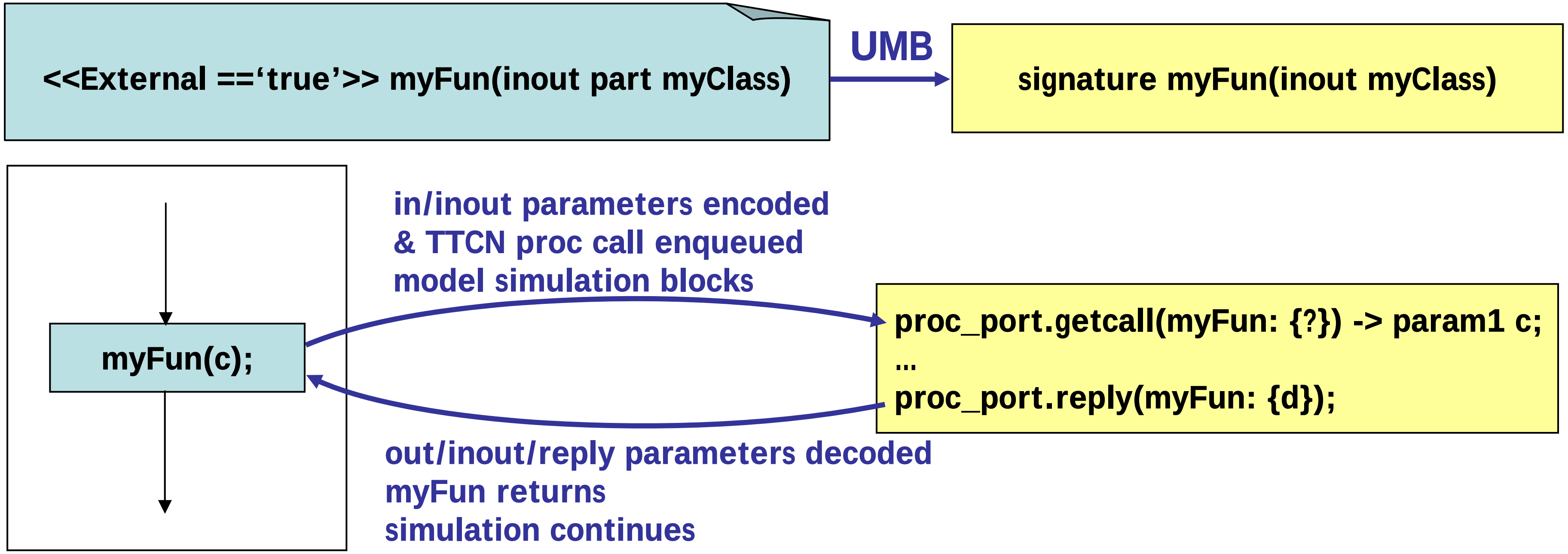
- **Using same model for model testing as target code generation**
  - avoid stubbing functionality in mode

- **How to test integration software**
  - marshalling code
  - platform interface code

External Operations

Interface to platform/system operations provided via external operations

- **for application generation user provides code/libraries**
- **for model testing cosim generates and compiles code to enable handling in TTCN**
  - UML operation body enqueues calls in TTCN TRI
  - UML operation body waits for response
  - TTCN getcall used to pick up call
  - TTCN reply to send response back to UML
  - UML operation body returns values and control to simulation run-time



<<External ==‘true’>> myFun(inout part myClass)

UMB

signature myFun(inout myClass)

myFun(c);

in/inout parameters encoded
& TTCN proc call enqueued
model simulation blocks

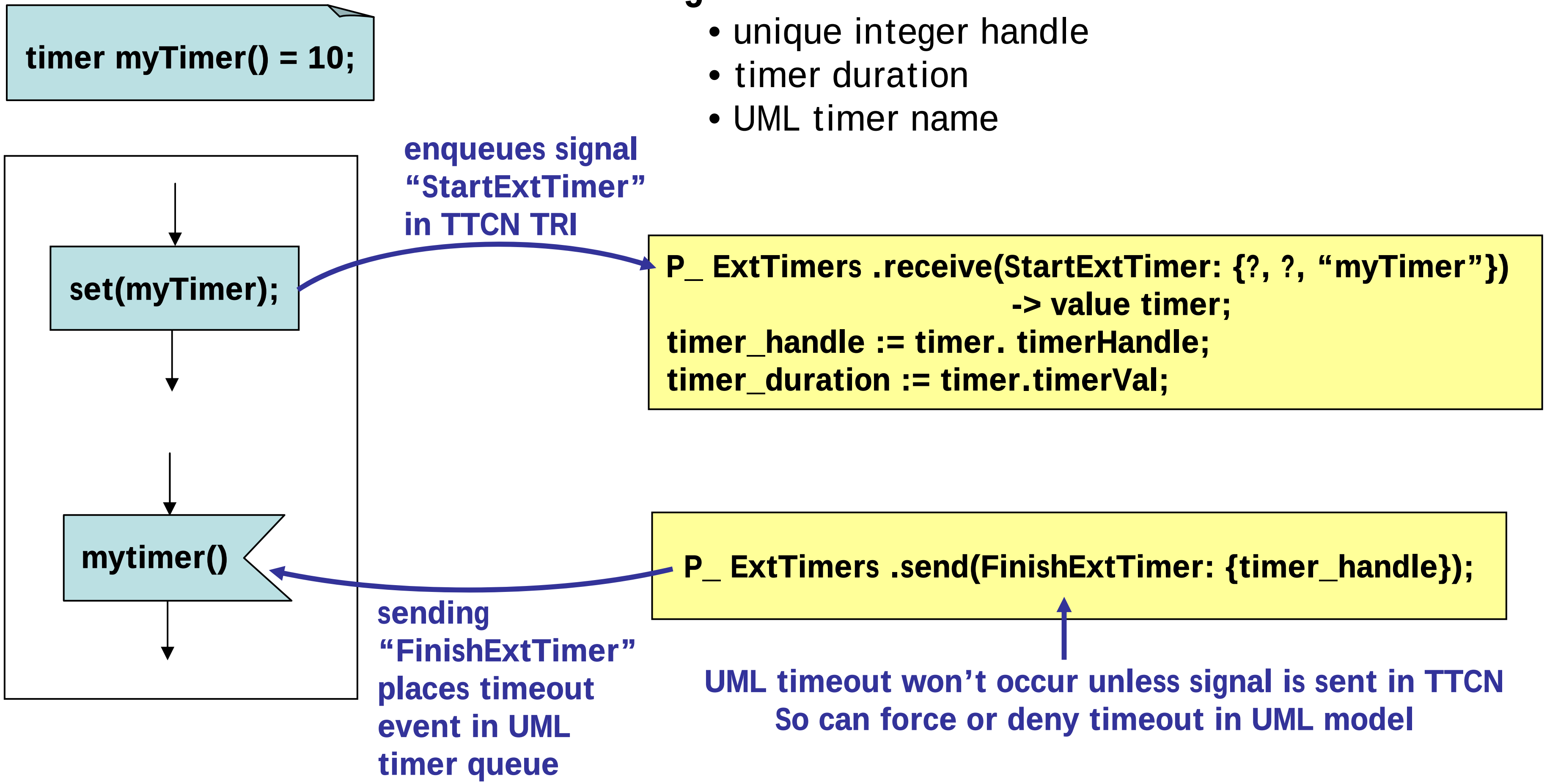
```
proc_port.getcall(myFun: {?}) -> param1 c;
…
proc_port.reply(myFun: {d});
```

out/inout/reply parameters decoded
myFun returns
simulation continues

Servicing Timers

UML Timers can be serviced by TTCN in cosim

- **Gives complete control of timers to test script**
- **UML timer events modelled as signals in TTCN**
- **signal contains:**
  - unique integer handle
  - timer duration
  - UML timer name



enqueues signal “StartExtTimer” in TTCN TRI

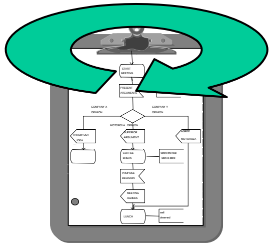
```
P_ ExtTimers .receive(StartExtTimer: {?, ?, “myTimer”})
                                        -> value timer;
timer_handle := timer. timerHandle;
timer_duration := timer.timerVal;
```

```
P_ ExtTimers .send(FinishExtTimer: {timer_handle});
```

sending “FinishExtTimer” places timeout event in UML timer queue

UML timeout won’t occur unless signal is sent in TTCN
So can force or deny timeout in UML model

cosim



cosim

Integration between UML Model & TTCN-3 execution

Supports:

- **Asynchronous signaling exchange**
- **UML External operations**
  - serviced in TTCN-3 by getCall/reply statements
  - synchronous external operation calls by Model
- **Timer handling**
  - test script is notified when timer is set via signal
  - script returns signal to expire timer in model
  - requires Tau kernel modification, supplied with cosim
- **cosim works with real-time model simulation**
- **TTCN test Execution**
  - dynamic via Tau Tester GUI for interactive use
  - static as determined by test suite
- **Batch execution via bridgeUI, supplied with cosim**
  - test plans can be used in batch mode
- **Test Management Tool Integration provided**

UML/TTCN-3 Mapping

UML Packages

- TTCN Modules
- nested UML packages -> TTCN groups

UML Signals

- TTCN record types (even if parameterless)

UML Constants

- TTCN constants

UML Port Definitions

- TTCN port type definitions
- UML signal lists are expanded out – no equivalent in TTCN
- directions used in UML are reversed for TTCN
  - signal output from UML port translates as an ‘in’ in TTCN

UML Interface Definitions

- Where used in ports, expanded out in TTCN

UML External Operation

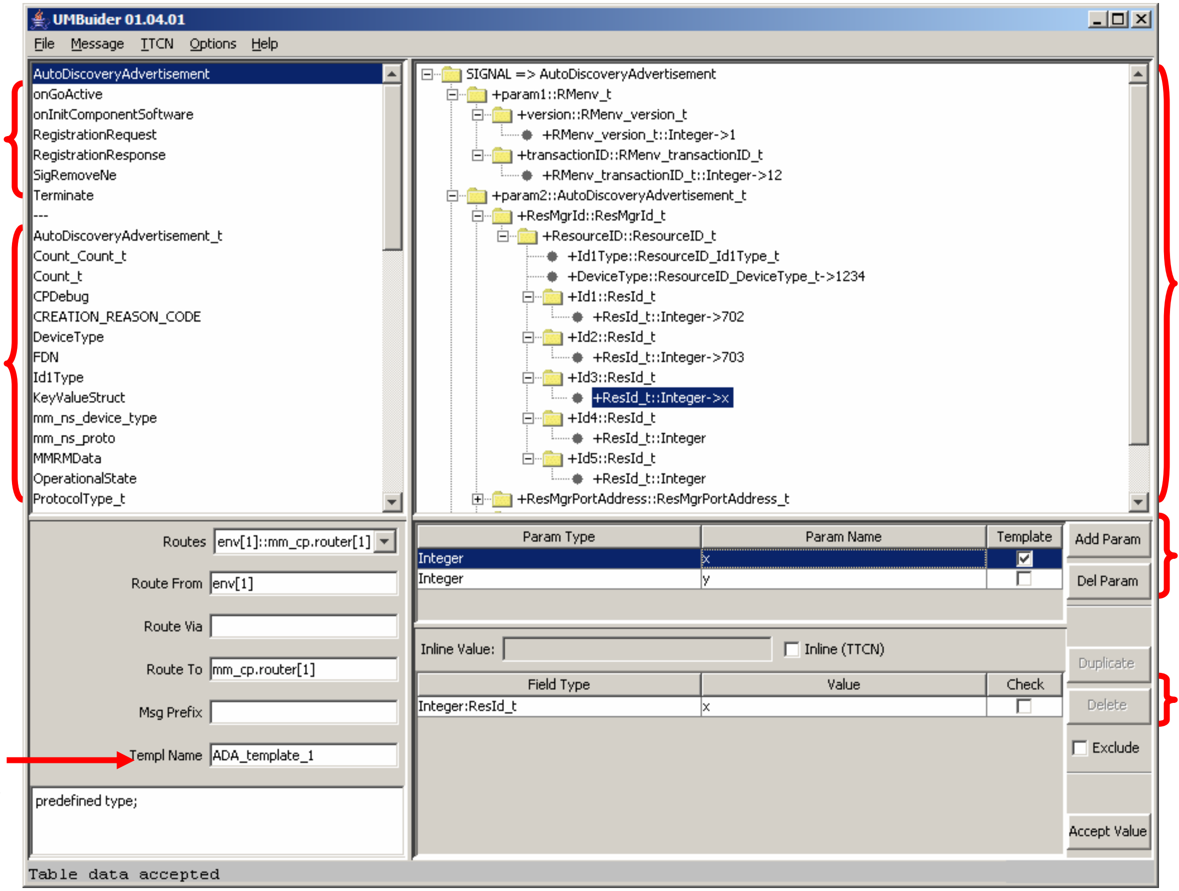
- TTCN signatures

UMB Template Generation

Signals

Types Classes

Template Name



AutoDiscoveryAdvertisement
onGoActive
onInitComponentSoftware
RegistrationRequest
RegistrationResponse
SigRemoveNe
Terminate

AutoDiscoveryAdvertisement_t
Count_Count_t
Count_t
CPDebug
CREATION_REASON_CODE
DeviceType
FDN
Id1Type
KeyValueStruct
mm_ns_device_type
mm_ns_proto
MMRMData
OperationalState
ProtocolType_t

```
SIGNAL => AutoDiscoveryAdvertisement
  +param1::RMenv_t
    +version::RMenv_version_t
      +RMenv_version_t::Integer->1
    +transactionID::RMenv_transactionID_t
      +RMenv_transactionID_t::Integer->12
  +param2::AutoDiscoveryAdvertisement_t
    +ResMgrId::ResMgrId_t
      +ResourceID::ResourceID_t
        +Id1Type::ResourceID_Id1Type_t
        +DeviceType::ResourceID_DeviceType_t->1234
        +Id1::ResId_t
          +ResId_t::Integer->702
        +Id2::ResId_t
          +ResId_t::Integer->703
        +Id3::ResId_t
          +ResId_t::Integer->x
        +Id4::ResId_t
          +ResId_t::Integer
        +Id5::ResId_t
          +ResId_t::Integer
    +ResMgrPortAddress::ResMgrPortAddress_t
```

Routes: env[1]::mm_cp.router[1]
Route From: env[1]
Route Via:
Route To: mm_cp.router[1]
Msg Prefix:
Templ Name: ADA_template_1

predefined type;

Param Type	Param Name	Template
Integer	x	☑
Integer	y	☐

Inline Value: ☐ Inline (TTCN)

Field Type	Value	Check
Integer:ResId_t	x	☐

☐ Exclude

Table data accepted

Graphical View of Template Spec

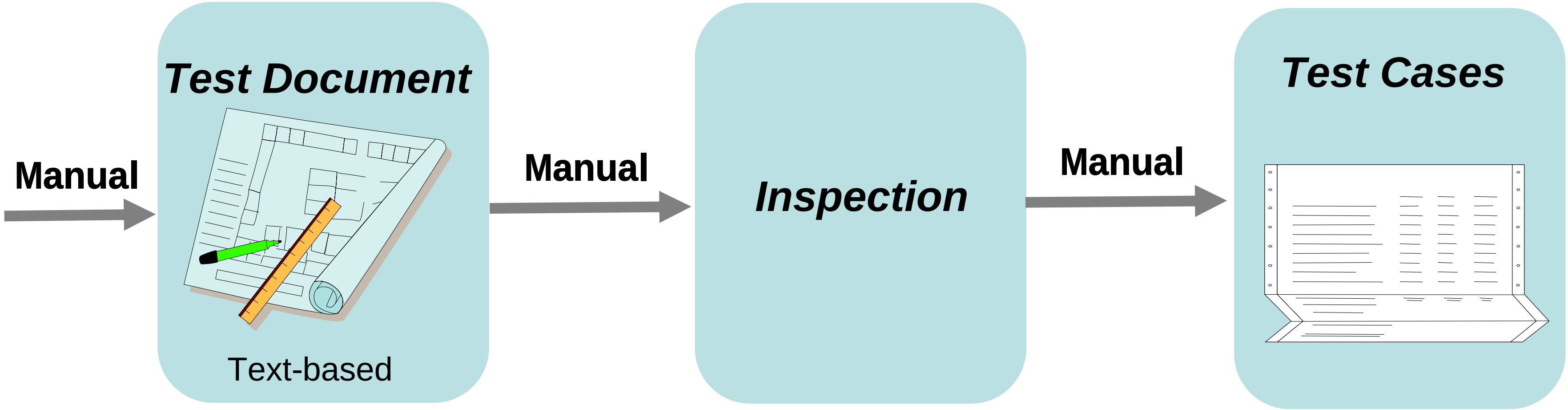
Template Param Defs

Template Field Defs

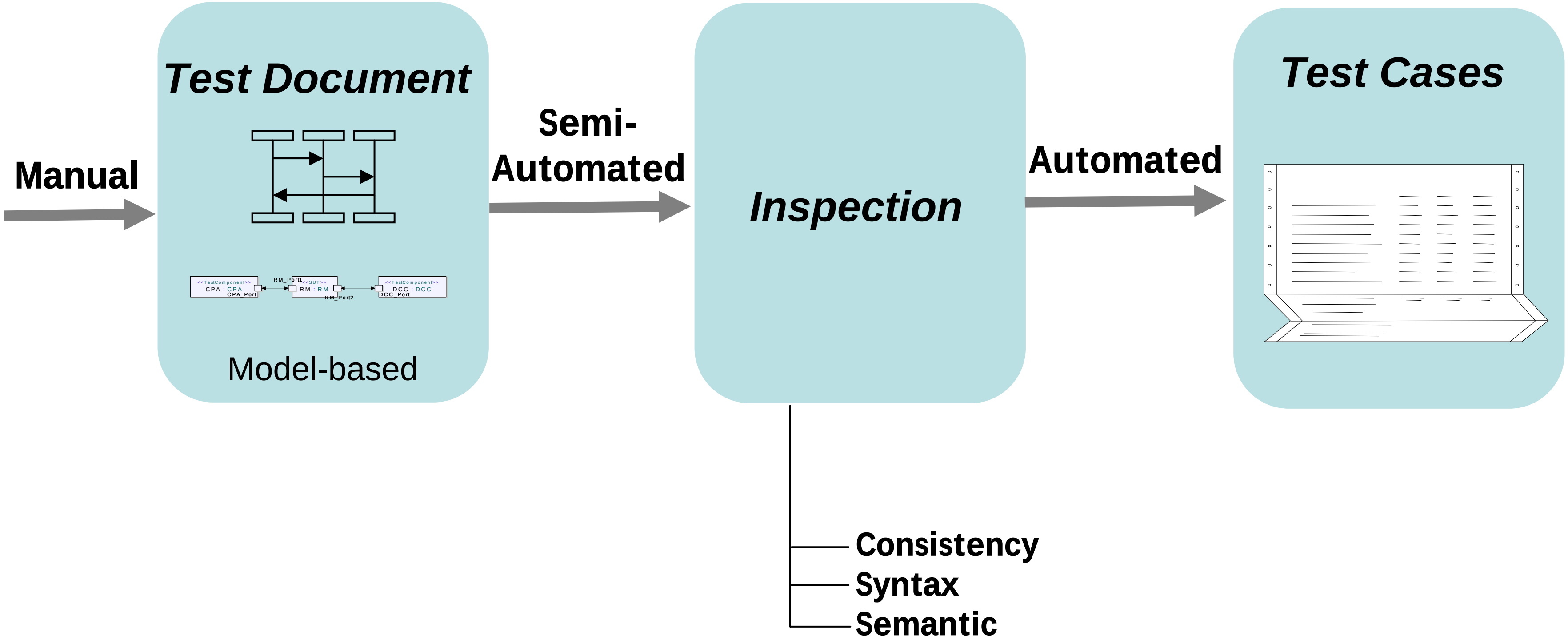
User can generate new or modify existing templates created by UMB

- **generates required import statements to type definition modules**

Test Specification - Manual Process



Semi-Automated Process



Manual

Test Document

Model-based

Semi-Automated

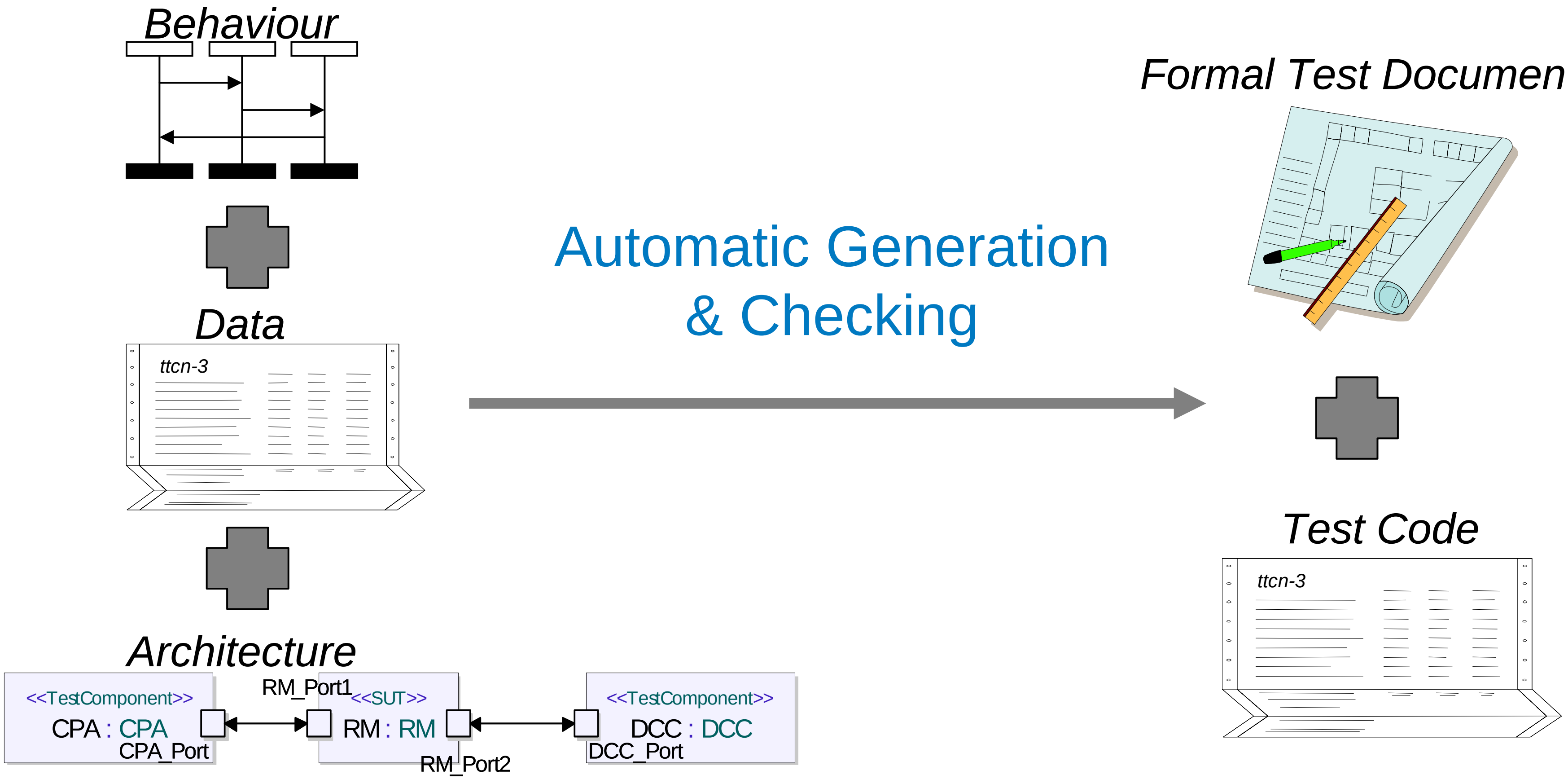
Inspection

- Consistency
- Syntax
- Semantic

Automated

Test Cases

UML Testing Profile + TTCN-3



- Defined semantics based on UML Testing Profile
- Consistency checking between architecture, behaviour, data
- Formal approach to test documentation process

Summary

Multiple projects using cosim/UMB in Motorola

- typical will involve dozens of tests per feature
- some tests exchange ~1000 signals
- applications generated from tested models are in deployed product

Added Benefits:

- Model tests reused to test generated application code running on target (using in-house Mousetrap UML code generator)
- Developing correct tests is easier in model testing (due to tracing of model it is easier to discover test defects)

Model Errors Uncovered

- Significant project reported Model Error discovery rate 1 in 10 tests
- Mostly common kinds of programming error:
  - ranges out of bounds
  - incorrect initialization
  - cut/paste errors
- Most Common UML Error:
  - passing signal parameters by reference errors
  - should normally be passed by value (part)

Further Details

Model-Driven Engineering Experience Papers

- Baker, P., Loh, S. and Weil, F. "*Model-Driven Engineering in a Large Industrial Context -- Motorola Case Study*." In L. Briand and C. Williams (Eds.) in MoDELS 2005, LNCS 3713, pp. 476--491, Springer-Verlag, 2005.

- Thomas Weigert, Frank Weil, Paul Baker, Kevin Marth, Aswin van den Berg, Thomas Cottenier, "*Practical Experiences in Using Model-Based System Engineering with UML*", Journal of Systems and Software, 2007.

TTCN-3

http://www.ttcn3.org

UML Testing Profile

- Baker, P., Dai, Z.R., Grabowski, J., Haugen, Ø., Schieferdecker, I., Williams, C. *"Model-Driven Testing Using the UML Testing Profile",* ISBN: 978-3-540-72562-6