

---

# Automated Test Case Generation

with

# TC-Gen

(an introduction)

## Testing Concept Overview

Motivation

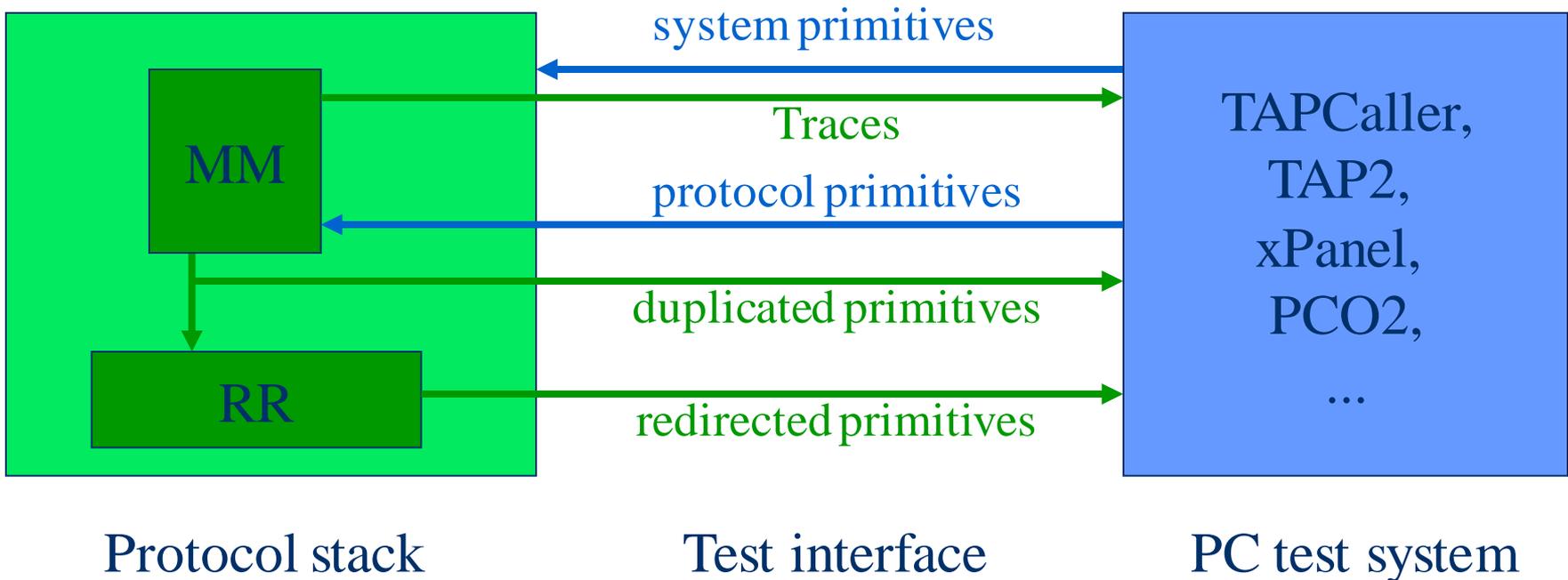
Usage

How it works

Current Limitations & Outlook

# TC-Gen ... *testing concept overview*

- Test interface approach:
  - ⇒ data interface between G23 protocol stack and a PC test system
  - ⇒ e.g. standard serial cable, COM-ports on both ends
  - ⇒ for running test cases: usually shared memory on one PC



# TC-Gen ... *testing concept overview*

---

- On stack side:
  - ⇒ test interface entity included in the FRAME
  - ⇒ uses corresponding hardware driver for communication
- On PC test system side:
  - ⇒ test interface executable using the FRAME
    - ◆ connects using standard OS drivers or via a Multiplexer
  - ⇒ used by tools like TAP, xPanel & PCO
  - ⇒ these tools finally provide GUI-stack-access for testers

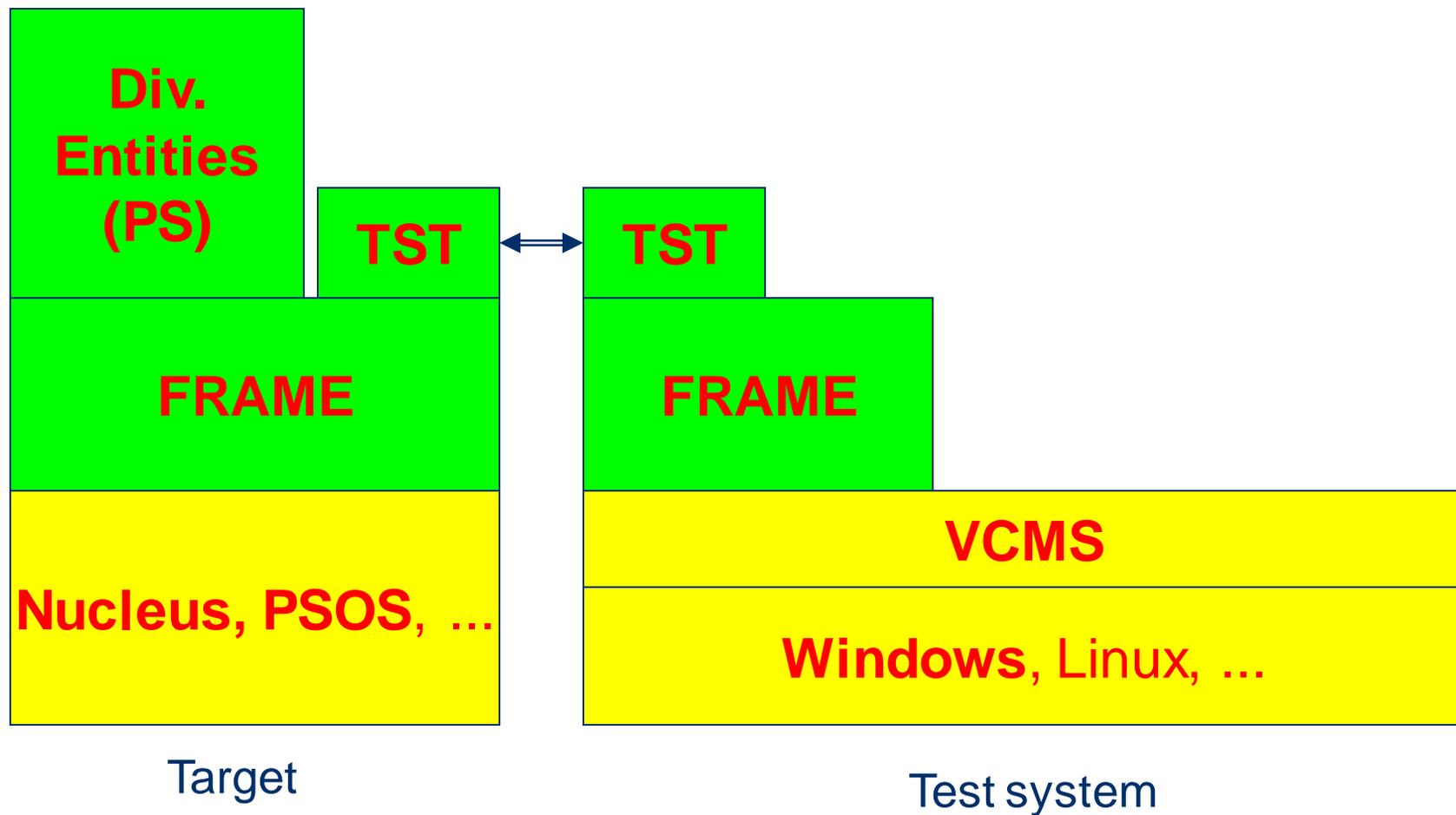
# TC-Gen ... *testing concept overview*

---

- TAP (Test Application Process) :
  - ⇒ program which executes test cases by sending, receiving and comparing primitives to/from the protocol stack
  - ⇒ TAPCaller provides GUI interface
- PCO (Point of Control and Observation):
  - ⇒ tools capable to stimulate the protocol stack and display traces and duplicated primitives
  - ⇒ used for logging and replaying test sessions
- xPanel (eXtended Panel):
  - ⇒ „virtual mobile“

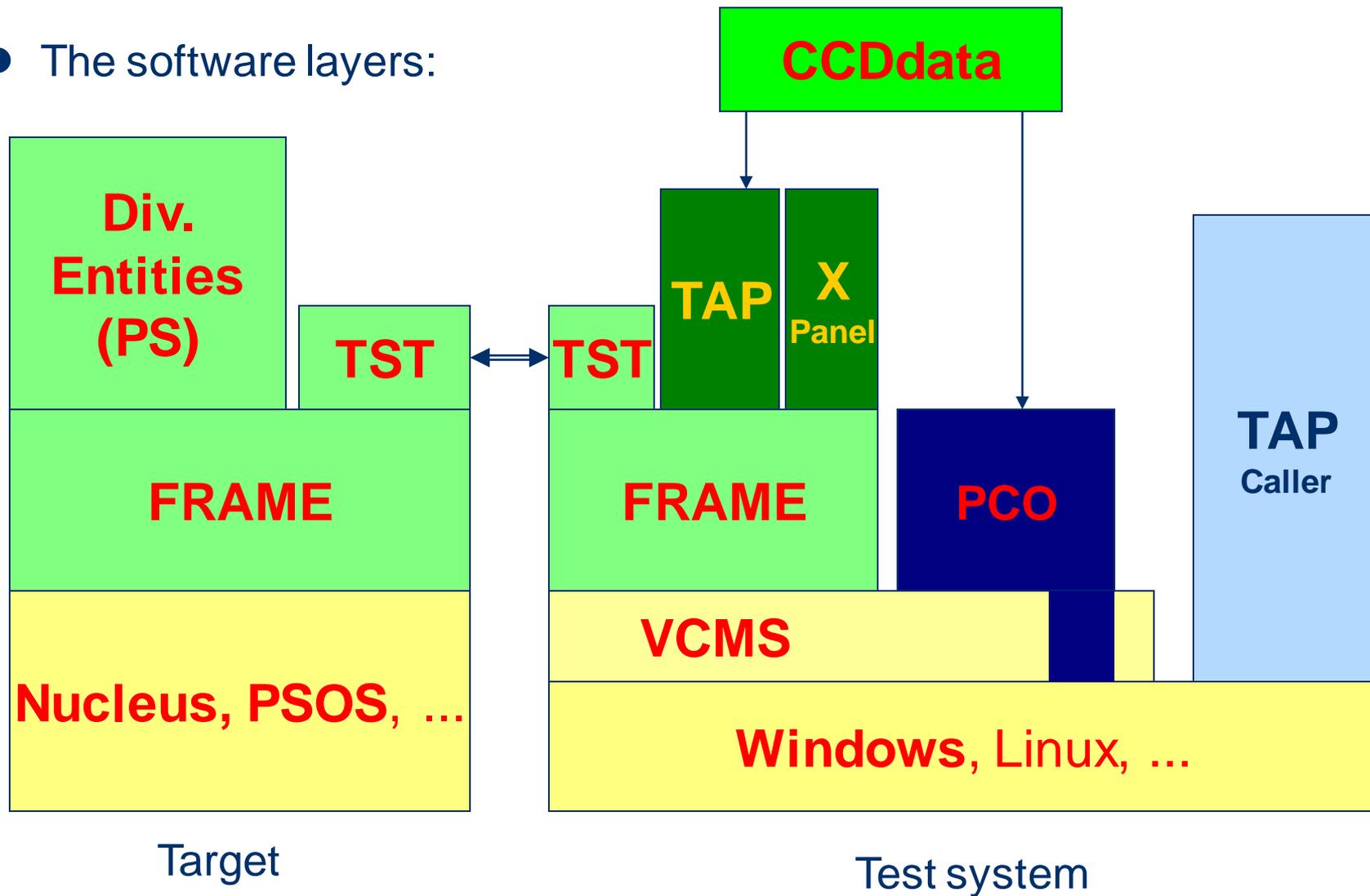
# TC-Gen ... *testing concept overview*

- The software layers:



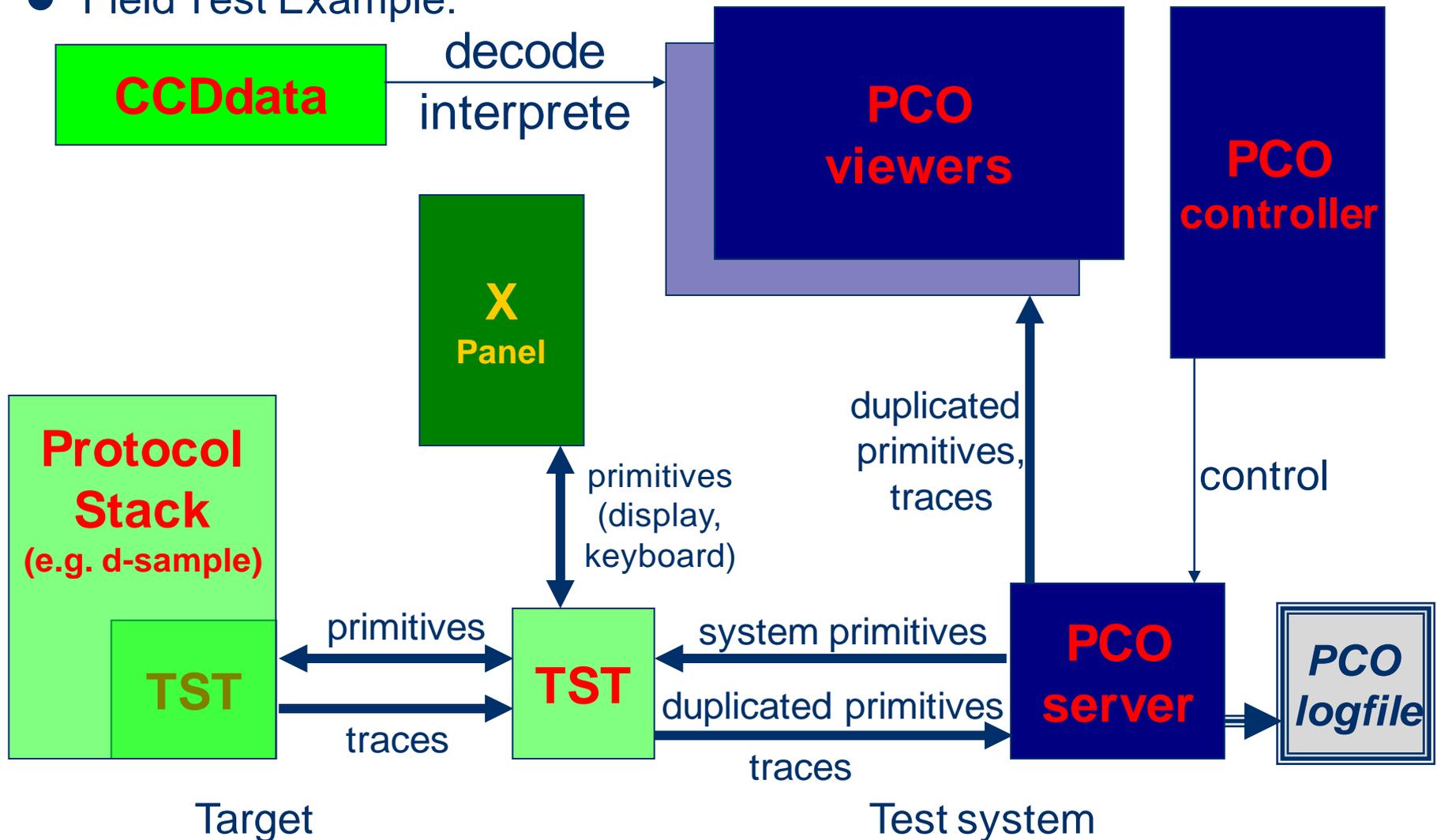
# TC-Gen ... *testing concept overview*

- The software layers:



# TC-Gen ... *testing concept overview*

- Field Test Example:



# TC-Gen ... *testing concept overview*

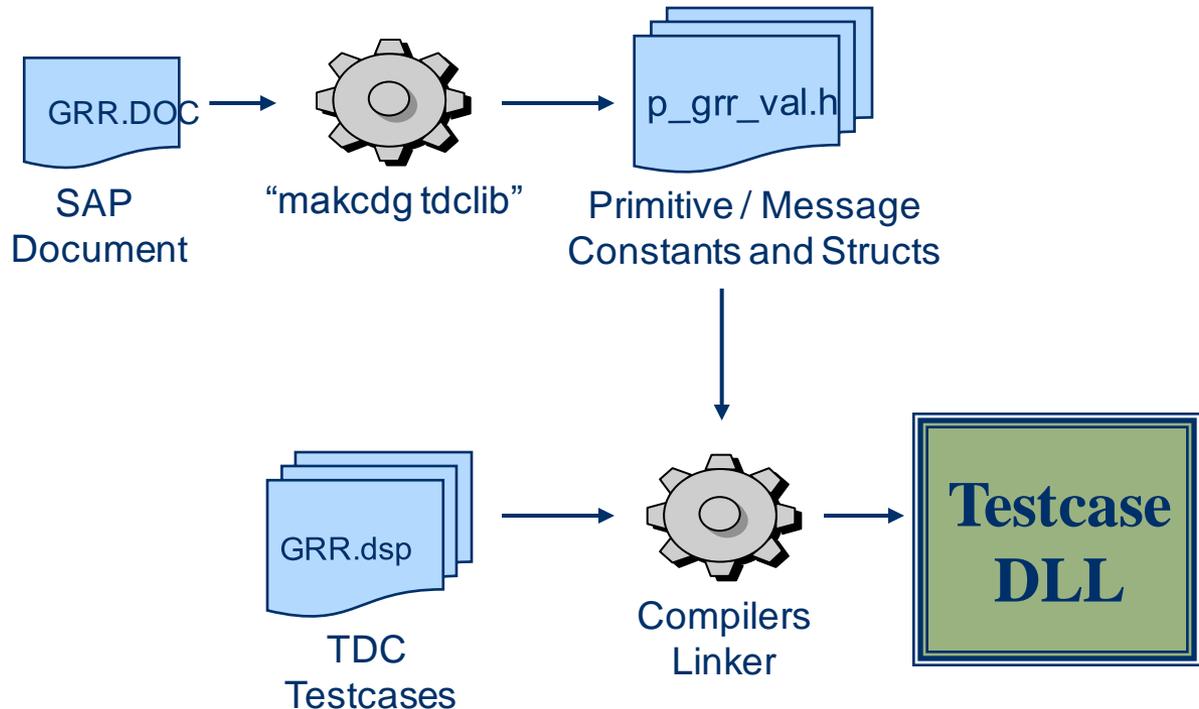
- TAP-Test Example (test case build):

⇒ Generators

- ◆ doc2txt
- ◆ xgen
- ◆ ccdgen

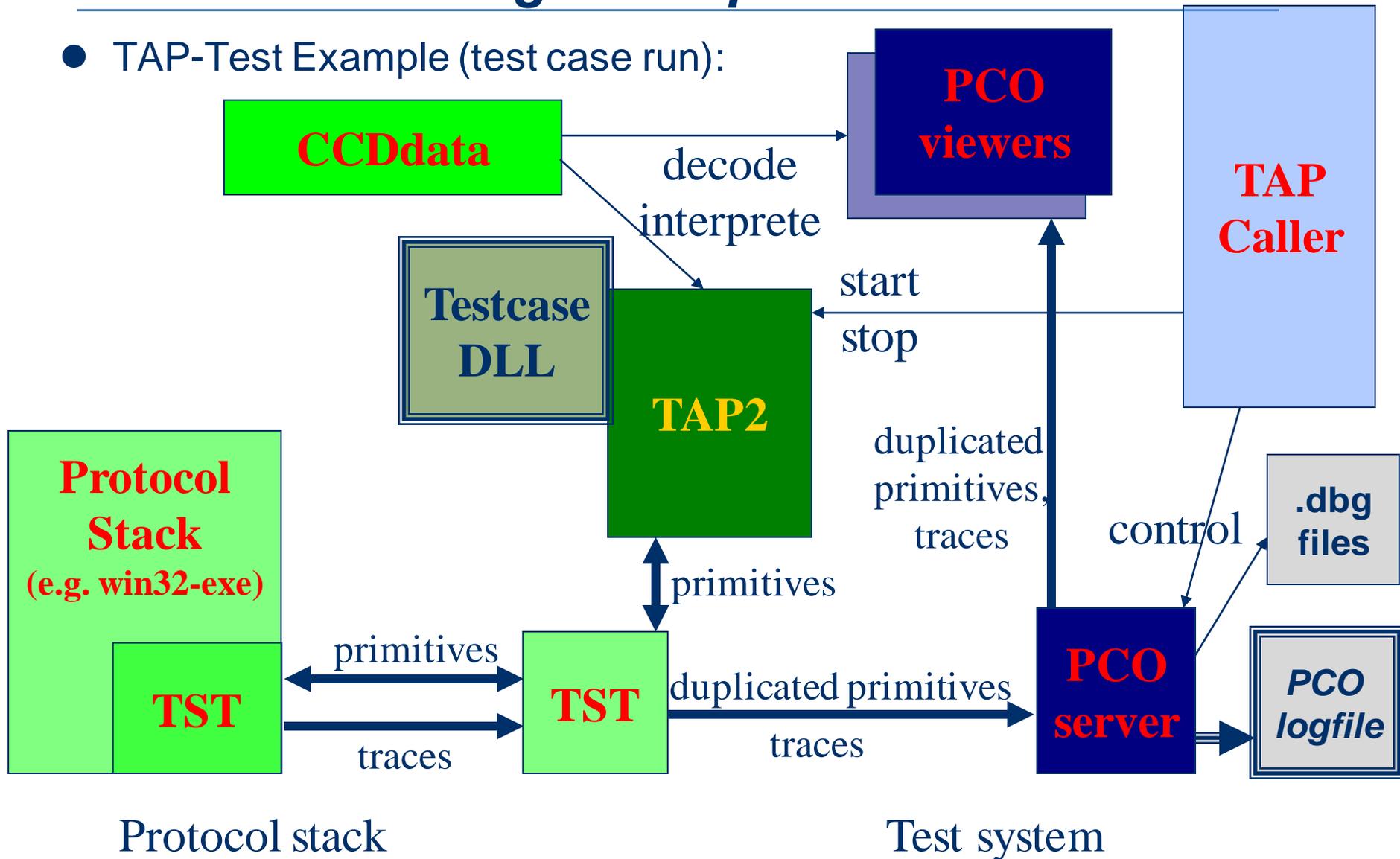
⇒ Compilers

- ◆ msdev / cl
- ◆ (mktc)



# TC-Gen ... *testing concept overview*

- TAP-Test Example (test case run):



# TC-Gen ... *testing concept overview*

- What is a PCO-logfile anyway ?
  - ⇒ contains all traces and primitives received during a time period of a testsession (e.g. a field test)
    - ◆ ... defined by starting/stopping logging
  - ⇒ the format is binary but with linebreaks after each primitive
    - ◆ ... and can be interpreted by PCO-server only
  - ⇒ any kind of PCO-viewer may request (portions of) it
    - ◆ ... for replaying or other tasks

## TC-Gen ... *testing concept overview*

- For more information see:

⇒ \gpf\DOC\testplatform\_userguide.doc

⇒ \gpf\DOC\tdc\tdc\_intro.ppt

⇒ \gpf\DOC\pco\pco\_intro.ppt

⇒ \gpf\DOC\tapcaller\tapcaller\_intro.ppt

⇒ ... and many other documents in \gpf\DOC !

# TC-Gen ...

---

## Testing Concept Overview

### **Motivation**

### Usage

### How it works

## Current Limitations & Outlook

# TC-Gen ... *motivation*

---

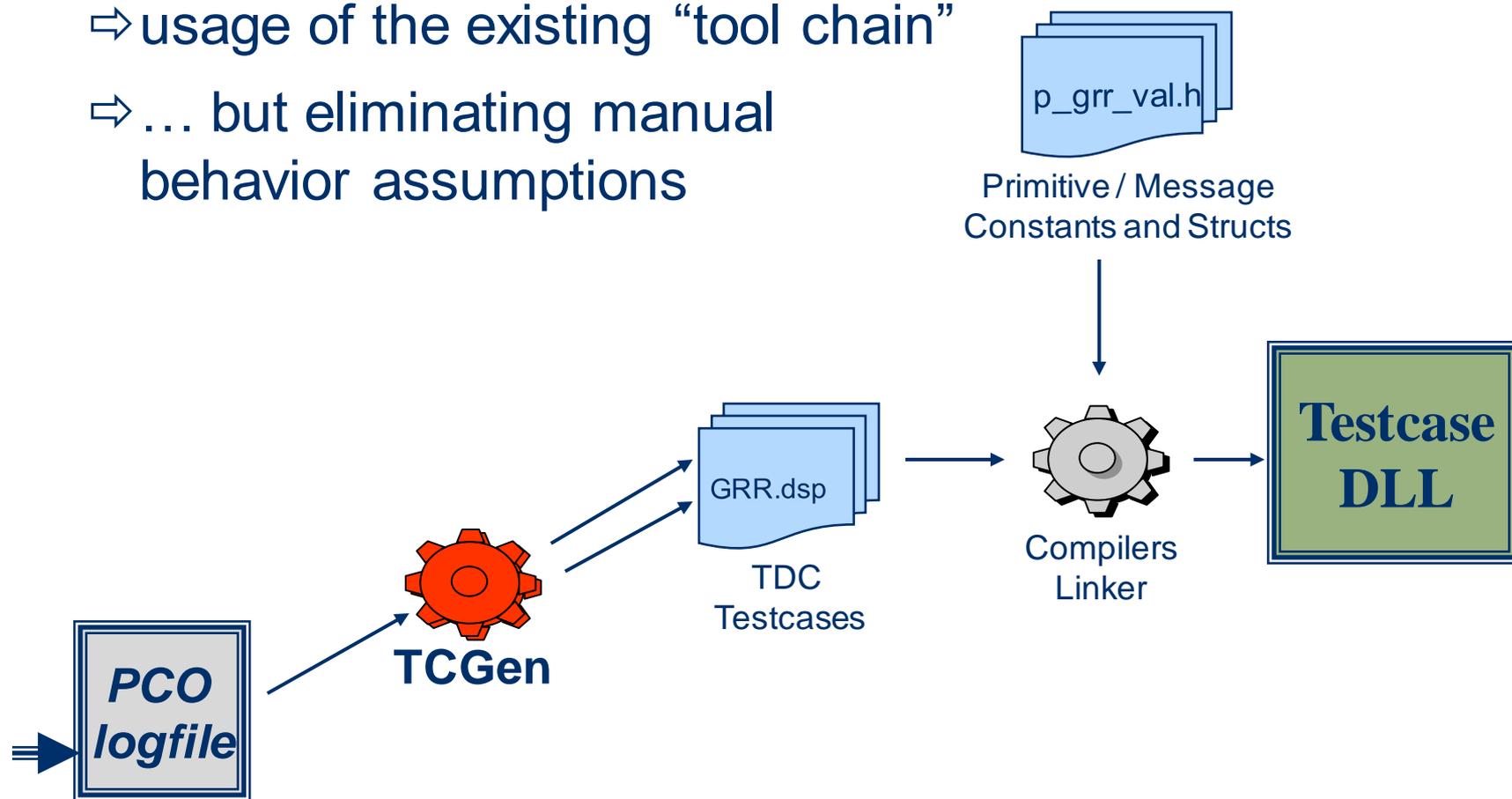
- Why do we need another generator ?
  - ⇒ current generators (like tdsgen) don't actually generate
    - ◆ ... but translate e.g. Word documents into TDS-files
    - ◆ (and anyway TDS will not be used anymore)
  - ⇒ the actual test case has to be written manually
    - ◆ ... requiring detailed knowledge of the expected behavior
  - ⇒ with an automated generator taking the real and up-to-date primitive flow into account ...
    - ◆ new test cases could be easily created
    - ◆ buggy behavior could be easily reproduced

# TC-Gen ... *motivation*

- The Idea:

- ⇒ usage of the existing “tool chain”

- ⇒ ... but eliminating manual behavior assumptions



# TC-Gen ...

---

Testing Concept Overview

Motivation

**Usage**

How it works

Current Limitations & Outlook

# TC-Gen ... usage

- At first we need a PCO-logfile:
  - ⇒ request duplication of primitives out of PS (currently manually, upcoming cddata features will enable more comfortable selecting – „Matrix Reloaded“ :-)
  - ⇒ and start logging into a specified .pco-file
  - ⇒ optionally watch arriving primitives

main.svc - P:\gpf\util\tcgen\testing\ans\4\_2523.pco loaded

T	Time	Pr...	Name	P...	Content
63736626	ms	MMI	GMMREG_ATTACH_REQ	??	04 03 01 00 E0 AB 00 00 40 70 31
63736707	ms	GMM	MMGMM_REG_REQ	??	01 02 04 00
63736707	ms	GMM	GMMRR_ENABLE_REQ	??	04 0B 01 00 10 00 00 02
63736847	ms	GRR	GMMRR_CR_IND	??	00 00 00 00
63742705	ms	GRR	GMMRR_CELL_IND	??	01 00 00 01 00 01 0F 33 01 00 05
63743026	ms	MM	MMGMM_ACTIVATE_IND	??	01 00 00 01 00 01 0F 00 01 00 01
63743236	ms	GMM	GMMREG_ATTACH_CNF	??	02 03 00 00 01 00 00 01 00 01 0F
63743436	ms	GMM	GMMRR_SUSPEND_REQ	??	00 01 02 01
63743446	ms	GRR	GMMRR_SUSPEND_CNF	??	06 0B 01 00
63743446	ms	GMM	MMGMM_REG_REQ	??	01 01 04 00
63743466	ms	MM	MMGMM_REG_CNF	??	01 00 00 01 00 01 0F 01 01 00 01
63743496	ms	GMM	GMMREG_ATTACH_CNF	??	02 00 3A 6F 01 00 00 01 00 01 0F
63743506	ms	GMM	GMMRR_RESUME_REQ	??	06 0B 01 00
63743516	ms	GRR	GMMRR_CELL_IND	??	01 00 00 01 00 01 0F 33 02 00 3A 6F 01 00
63743797	ms	GMM	GMMRR_CELL_RES	??	01 00 00 01
63744047	ms	GMM	LLGMM_ASSIGN_REQ	??	FF FF FF FF 44 33 22 91 00 00 00
63744087	ms	GMM	GMMRR_ASSIGN_REQ	??	44 33 22 91 44 33 22 D1 01 01 00
63744217	ms	GMM	GMMRR_ATTACH_START...	??	12 44 33 22
63744237	ms	GMM	LL_UNITDATA_REQ	??	01 00 00 00 44 33 22 91 00 03 00

PCO-Controller

Session

Switch to REPLAY mode

Name: GMM001

Start logging Stop logging

Test environment

Activate Add viewer ...

Restart Configure ...

Stack control

Send stack configuration

Send:

Send:

Send:

Send:

Send:

Traceclasses Primitives

	GMM	MMI	MM	RR	GRR	PL
GMM	<input checked="" type="checkbox"/>	<input type="checkbox"/>				
MMI	<input checked="" type="checkbox"/>	<input type="checkbox"/>				
MM	<input checked="" type="checkbox"/>	<input type="checkbox"/>				
RR	<input checked="" type="checkbox"/>	<input type="checkbox"/>				
GRR	<input checked="" type="checkbox"/>	<input type="checkbox"/>				
PL	<input type="checkbox"/>	<input type="checkbox"/>				

ROUTING

Send system primitive to MMI

Exit Communication... ?

# TC-Gen ... *usage*

- Second we use TCGen with the PCO-logfile:

⇒ start the PCO-server (if not already running)



⇒ call TCGen:

```
usage:
tcgen -h ! -ver ! [-i <ini-file>] [-wait] <pco-input-file> [-analyse ! <output-d
ir> <entities> [-n <testcase-name>] [-p <project>]]
example: tcgen mmgmm001.pco c:/tdc_testing/mmgmm MM,GMM -p mmgmm -n MMGMM023
```

⇒ ok, call it with appropriate parameters ☺

(library paths and more parameters are taken from the tcgen.ini file)

```
[P:\gpf\BIN]tcgen c:\testsessions\GMM_SM001.pco \g23m\condat\ms\test\dsample GMM
,SM -p gmmsm -n GMMSM028
sourcing P:\gpf\BIN\..\cfg\tcgen.ini ...
sourcing P:\gpf\BIN\..\cfg\pco.ini ...
generating testcase ...
... generation succeeded !
-> See "\g23m\condat\ms\test\dsample" for results.
```

- ⇒ evtl. inspect the generated files in the VisualStudio  
(if the project already contained cases the new stuff will be merged at the beginning of all files)
- ⇒ ... or just build the testcase DLL from command line:

```
[P:\gpf\BIN]msdev \g23m\condat\ms\test\dsample\gmmsm_test.dsp /MAKE
Building default target: gmmsm_test - Win32 Debug
```

# TC-Gen ... usage

- A look at the results:

tap\_prims.svc - c:\testsessions\GMM\_SM001.pco loaded

File Edit View Server Target Tools Help

T	Snd	Name	Rcv	Content
MM	MMGMM_TMSI_IND	GMM	C2 63 D2 05	
GMM	GMMREG_ATTACH_CNF	MMI	02 0B 12 04	
MMI	GMMREG_PLMN_MODE_REQ	GMM	00 03 00 00	

Element	Value
GMMREG_ATTACH_CNF	OPC: 0x7300
attach_type (Attach type)	02
plmn (PLMN identification)	01 02 06 02 00 01 0F 03
v_plmn (valid flag)	01
mcc (mobile country code)	02 06 02
mcc [0]	02
mcc [1]	06
mcc [2]	02
mnc (mobile network code)	00 01 0F
lac (location area code)	06 30
rac (routing area code)	FF
cid (cell id)	16 8F
gprs_indicator (GPRS indicator)	00
search_running (Search is still running)	00

Ready

```
// Test Cases: gmm_sm_cases.cpp
```

```
T_CASE GMMSM028()
{
    BEGIN_CASE ("replay_from_logfile")
    {
        setup_routings_for_GMMSM028();
        run_logged_stuff_from_GMMSM028();
    }
}
```

```
// Test Step Excerpt: gmm_sm_steps.cpp
```

```
port2GMM.SEND (mmgmm_tmsi_ind_GMMSM028_8());
AWAIT (gmmreg_attach_cnf_GMMSM028_9());

TIMEOUT(300);

port2GMM.SEND (gmmreg_plmn_mode_req_GMMSM028_10());
```

```
// Constraints Excerpt: gmm_sm_constraints.cpp
```

```
T_PRIMITIVE_UNION gmmreg_attach_cnf_GMMSM028_9()
{
    T_GMMREG_ATTACH_CNF prim;
    prim->attach_type= 0x02;
    prim->plmn= plmn_GMMSM028_6();
    // ...
}

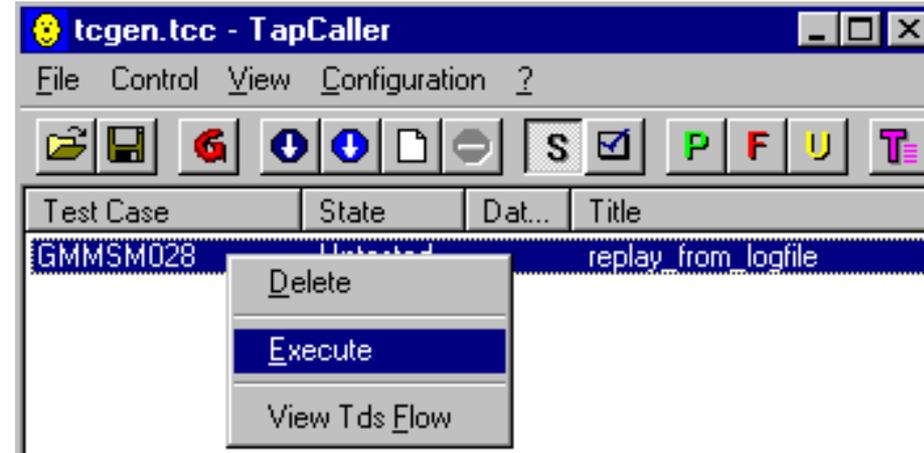
T_plmn plmn_GMMSM028_6()
{
    T_plmn pstruct;
    pstruct->v_plmn= 0x01;
    pstruct->mcc= mcc_array_GMMSM028_7();
    // ...
}

T_ARRAY<U8> mcc_array_GMMSM028_7()
{
    const U8 array_elements[3]=
    {
        0x02,
        0x06,
        0x02
    };
    return T_ARRAY<U8>(array_elements);
} // mcc_array_GMMSM028_7;
```

# TC-Gen ... *usage*

- Now what to do with it ?

⇒ you might use TAP and run the fresh test case with a Windows-PS



⇒ you might edit the TDC files and adapt them to your special needs

- ◆ e.g. change „online parameters“ like IMSI
- ◆ unfortunately manually by using the VisualStudio
- ◆ ... due to the lack of a dedicated „test case editor“
- ◆ of course the beloved perl scripts can do nice work, too 😊

⇒ ... then rebuild the test case and use TAP again

# TC-Gen ...

---

Testing Concept Overview

Motivation

Usage

**How it works**

Current Limitations & Outlook

## TC-Gen ... *how it works*

---

- TCGen is implemented as a PCO-Viewer ...
  - ⇒ by internally deriving from viewer core C++ classes
  - ⇒ which enables it to easily request all primitives from a specified PCO-logfile
  - ⇒ ... as often as it wants to
- In a first pass ...
  - ⇒ all necessary routings are generated in TDC
    - ◆ using information about the primitive sender and receiver
    - ◆ ... and the SAP name, currently taken from primitive name
  - ⇒ all declarations of constants, structs a.s.o.
    - ◆ are written to the TDC header files

# TC-Gen ... *how it works*

---

- Many further passes ...
  - ⇒ look inside the primitive hierarchy level by level
  - ⇒ generate:
    - ◆ SEND() / AWAIT() commands for primitives
    - ◆ parameter values and sub-structures
    - ◆ TIMEOUT() commands for delays
  - ⇒ ... and finally the whole test case
  
- More implementational details ...
  - ⇒ will be documented in  
  \gpf\DOC\tcgen\tcgen\_description.doc
  - ⇒ can be found out from the sources in  
  \gpf\util\tcgen\... 😊

# TC-Gen ...

---

Testing Concept Overview

Motivation

Usage

How it works

**Current Limitations & Outlook**

## TC-Gen ... *current limitations & outlook*

---

- Unfortunately in many cases the pure generated test case will fail due to:
  - ⇒ a lot of parameter values coming from network
    - ◆ ... and with Windows-PS we don't have that
  - ⇒ sometimes missing initial primitives
    - ◆ e.g. with BMI-image the first SIM\_GMM\_INSERT\_IND gets lost
  - ⇒ several parameters which could be ignored
    - ◆ ... but which are expected with exactly the recorded value
- *A quite funny side effect:*
  - ⇒ *if the PCO-logfile comes from a TAP run which failed*
  - ⇒ *... the new generated test case will pass 😊*

# TC-Gen ... *current limitations & outlook*

---

- Further problems are ...

- ⇒ quit long time differences between primitives

- ◆ ... which are not necessary with the Windows-PS
- ◆ but a maximal timeout value can be defined

- ⇒ primitive duplication does not work with all entities by now

- ◆ because of changed timing
- ◆ e.g. for LLC the PS will crash

- ⇒ currently no support of duplication of pointers by the FRAME

- ◆ e.g. for descriptor lists

- And finally ...

- ⇒ air-messages are currently handled as binary blocks

- ⇒ no support of other formats then TDC

- ◆ but TTCN-3 should be no problem

- ⇒ no complete GUI exists

- ◆ allowing execution of all processes from one place

## TC-Gen ... *current limitations & outlook*

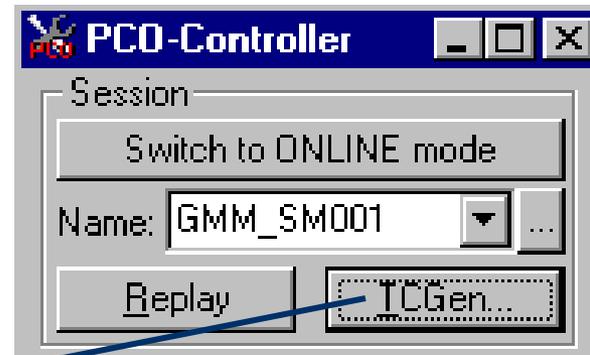
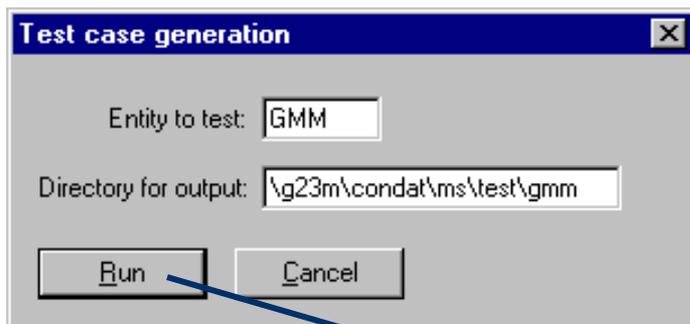
- Some solutions seem obvious:

- ⇒ the generated TDC files could be post processed to ...
  - ◆ set parameters to „not used“ etc.
- ⇒ the TST queue on target could be temporarily enlarged
  - ◆ to reduce timing influences
- ⇒ the support of pointer duplication could be implemented
  - ◆ e.g. similarly as done with PCON on UMTS stacks

## TC-Gen ... *current limitations & outlook*

### ● Concerning the GUI support ...

⇒ a first step is already done:  
TCGen can be started right  
from the PCO-controller



```

MS-DOS P:\GPF\BIN\tcgen.exe
sourcing P:\gpf\BIN\..\cfg\tcgen.ini ...
sourcing P:\gpf\BIN\..\cfg\pco.ini ...
generating testcase ...
... generation succeeded !
-> See "\\g23m\condat\ms\test\gmm" for results.
-> For compiling the test-dll call e.g.:
    msdev \\g23m\condat\ms\test\gmm\GMM_test.dsp /MAKE
press any key to continue ...
  
```

## TC-Gen ... *current limitations & outlook*

- TCGen could/will be enhanced to support:
  - ⇒ interpretation of air messages using ccddata-DLL
  - ⇒ upcoming ccddata features to find out redirections independent from „random“ primitives in logfile
  - ⇒ the option to generate primitives with direct parameter passing
  - ⇒ configuring the level of structure expanding
  - ⇒ rules specified in an extra file and applied during generation
    - ◆ to e.g. set/ignore special parameter values
  - ⇒ a dedicated GUI for convenient generation
- ... after all TCGen is still in Beta state 😊

---

**Thanx for your patience !**