6th UCAAT User Conference on Advanced Automated Testing

ETSI

Ease testing of clock based systems

Emmanuel Gaudin & Mihal Brumbulli

Clock based system principle

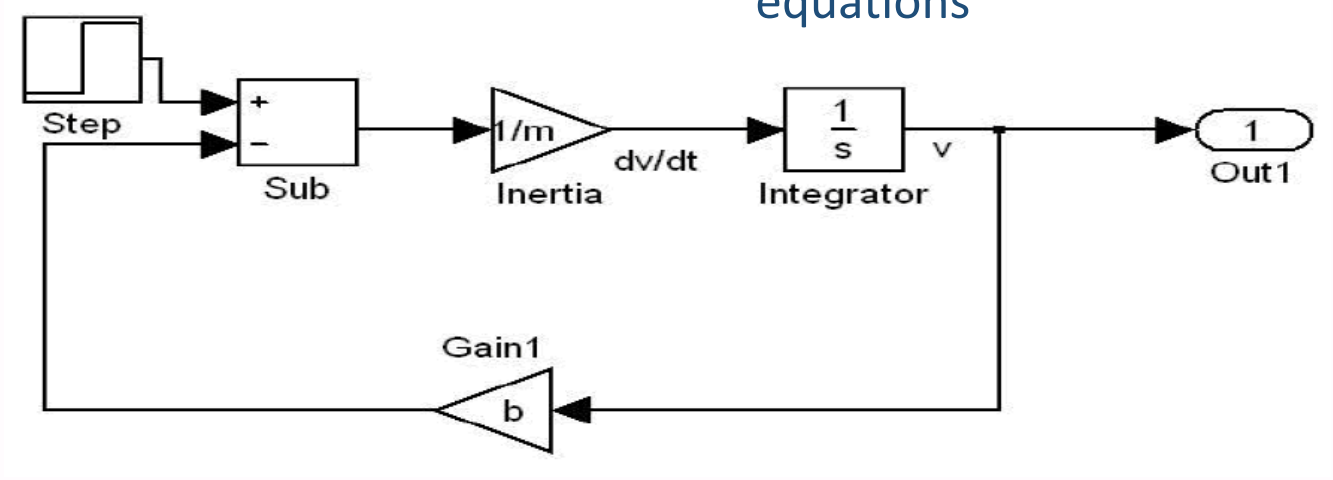
- An input comes from reading a sensor every tick.
  Usually from a few times to a few hundred times per second.
- The information is computed.
- An output is generated.
- Close to the implementation

Clock based models

- Logic system
  - Binary sensor (on/off)
  - Easier to verify

Clock based models

- Continuous system
  - Complex sensor (temperature, gyroscope…)
  - Control law with differential equations



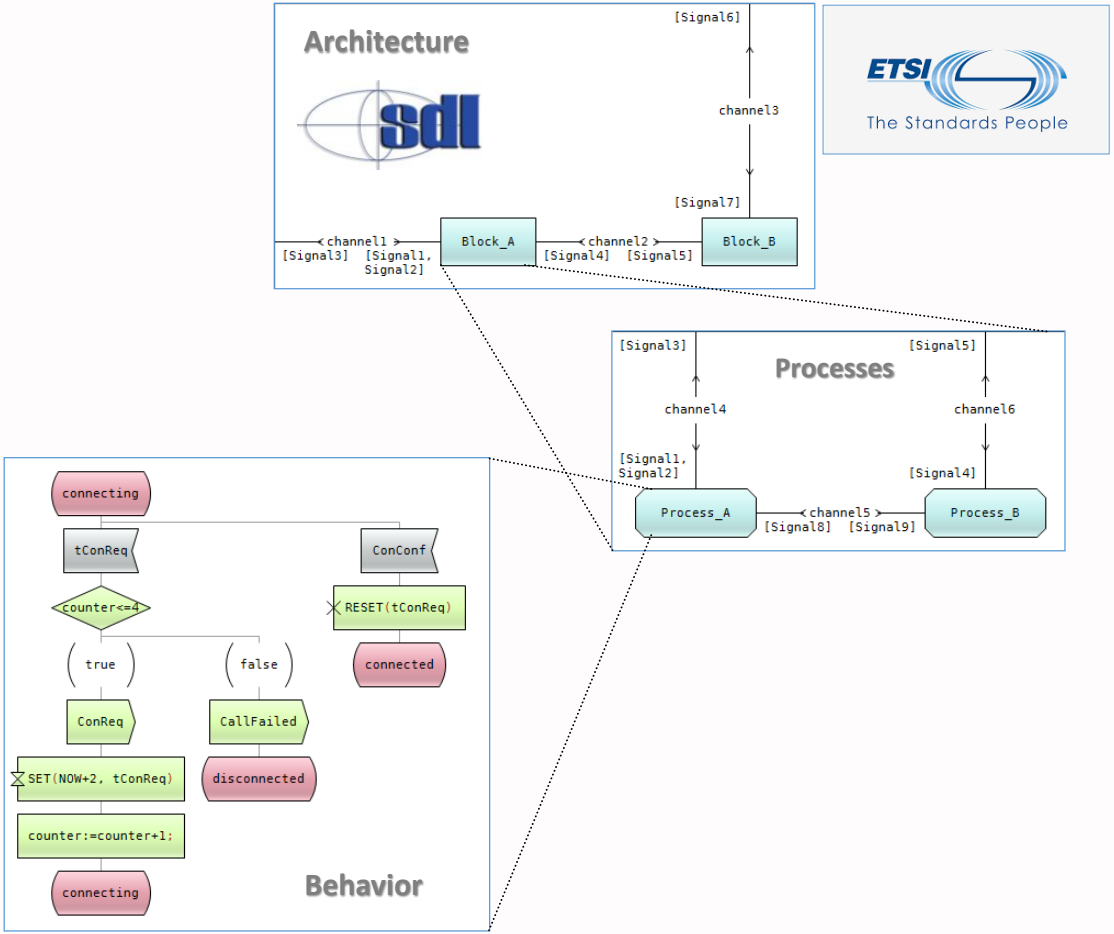
Testing clock based models

- Generate a value for each tick
  - Represents a massive amount of information
  - The underlying logic is not easily readable
- Example of rail signaling system
  - The door closes after 10 seconds
  - If a sample 10 times per second this will generate 100 times the same information
  - The same for the expected output to be verified

Tick	Input 1	Input 2	Output
284	1	0	0
285	1	0	0
286	1	0	0
287	1	0	0
288	1	1	1
289	1	1	1
290	1	1	1
291	1	1	1
292	1	1	1
…			

Event driven systems

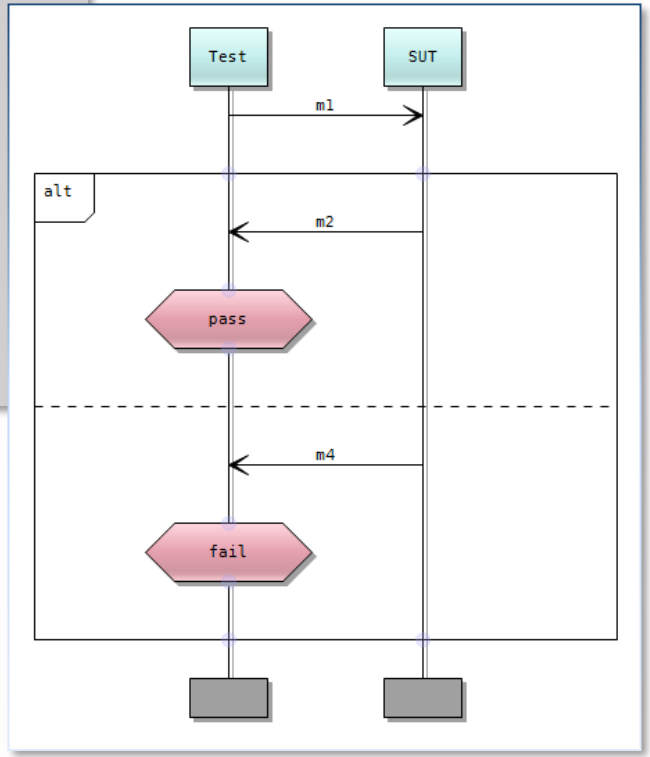
- Based on messages
  - Telecommunication systems
  - High abstraction level
    Globally Asynchronous
    Locally Synchronous
  - Close to the high level requirements
- Parallel processing
  - Generates complexity
  - Very hard to verify



Testing of event driven systems

- Based on messages
- Stays readable
- Sequence of events more important than absolute timing
- Interleaving of messages and various parameter values create a large number of possible scenarios
- Impossible to test all possible scenarios

```
testcase simple()
  {
  port.send(m1)
  alt
    {
    []port.receive(m2)
      {
      setverdict(pass);
      }
    []port.receive(m4)
      {
      setverdict(fail);
      }
    }
  }
```



TTCN-3

Event based testing applied to clock based system

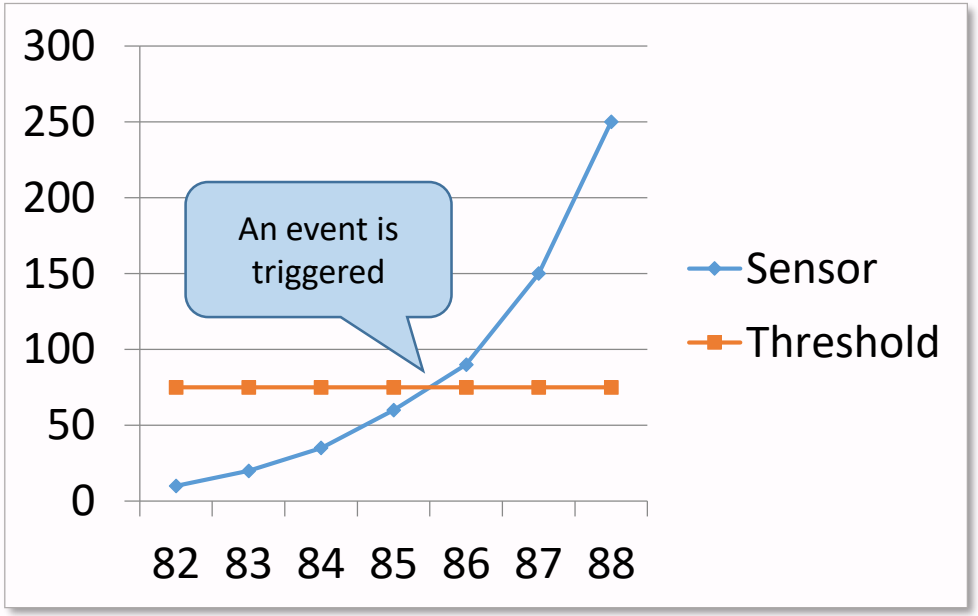
- The logical value triggers an event
  - Example

```
Sync  0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1
Async - - - - - - - - - - - - - - On- - - - - - - - - -
```

  - It is only a matter of indicating when the event occurs
  - With event driven approach only the sequence of events matter

Event based testing applied to clock based system

- The continuous value eventually triggers an event
  - An event is generated when a threshold is reached.
  - The threshold might have a margin for triggering the event.

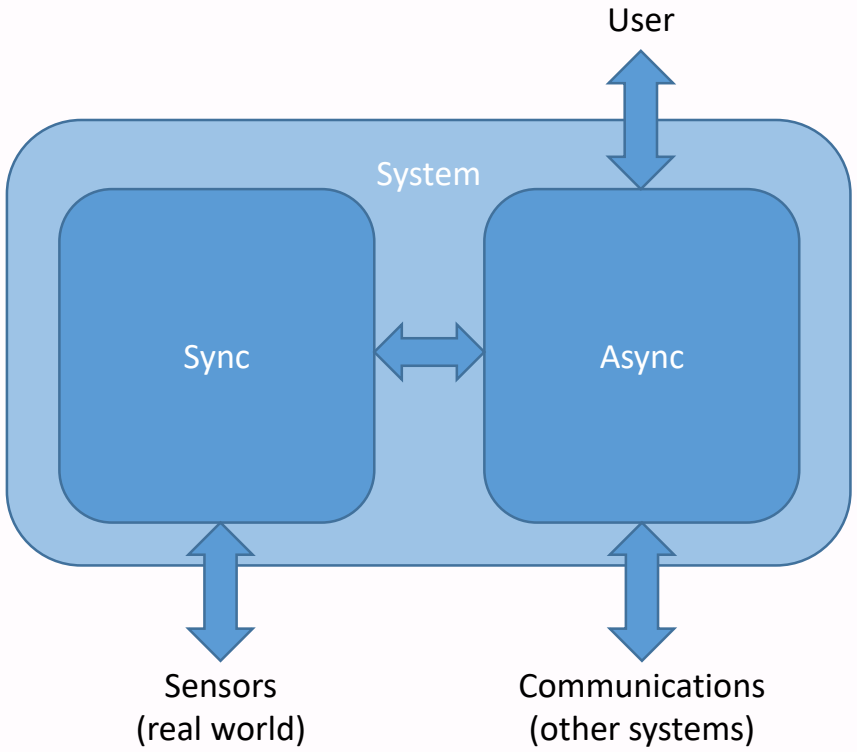


High level requirements

- Is often event based
- Model based
- Conformance testing is close to the requirements
- Conformance testing is most suited with an event based approach
- Links to the clock based implementation

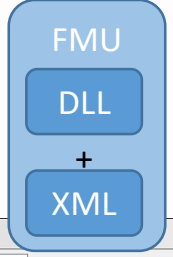
Cyber physical systems

- Combines event driven and clock based approach
- Even driven testing can be used on both parts
- Event based requirements
- Event based testing is most suited



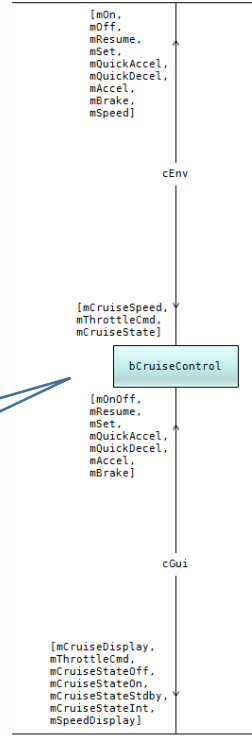
FMI

- Functional Mock-up Interface (FMI)
- Connects Functional Mock-up Units (FMU)
- Initially made to connect clock based models
- The clock is part of the interface
- With the same testing principles, it can be a link between event driven and clock based approach



Example

- Cruise control example
- Speed and throttle are physical measures
- Start or stop are logical events
- Increase or decrease speed are events



Example

- Requirements covered by test cases

PragmaDev Studio - Requirements table "MyReq.csv"

Table Windows Help

ReqId	ReqDescription	Cover	Test
CC_HLR_CCB_01	When the driver starts the car, the CC shall be off. The output CruiseState should be set to OFF		TTCN_TestsAndControl.TC_HLR_CCB_01
CC_HLR_CCB_02	The CC shall be set on when the driver push the on button.		TTCN_TestsAndControl.TC_HLR_CCB_02
CC_HLR_CCB_03	The CC shall automatically go off when the off button is pressed.		TTCN_TestsAndControl.TC_HLR_CCB_03
CC_HLR_CCB_04	If the car speed is in the speed limit and the accelerator pedal is not pressed, the CC shall be on and regulate the car speed. The output CruiseState shall be set to ON.		TTCN_TestsAndControl.TC_HLR_CCB_04
CC_HLR_CCB_05	The CC system shall be automatically disabled when the accelerator pedal is pressed, or the car speed is outside the speed limit. The output CruiseState shall be set to STDBY.		TTCN_TestsAndControl.TC_HLR_CCB_05
CC_HLR_CCB_06	The system shall return to the ON state when both the accelerator pedal is not pressed, and the car speed is inside the speed limit. The last set cruise speed shall be re-used.		TTCN_TestsAndControl.TC_HLR_CCB_06
CC_HLR_CCB_07	The CC shall be immediately interrupted when the brake is pressed. The output CruiseState shall be set to INT.		TTCN_TestsAndControl.TC_HLR_CCB_07
CC_HLR_CCB_08	The system shall resume either to the ON or STDBY states, depending on the Accelerator pedal and the Speed value when the resume button is pressed. The last set cruise speed shall be re-used.		TTCN_TestsAndControl.TC_HLR_CCB_07
CC_HLR_CDC_01	When the CC is off, the car speed shall be driven using the accelerator pedal		TTCN_TestsAndControl.TC_HLR_CCB_08
CC_HLR_CDC_02	When the CC is on, the car speed shall be automatically regulated.		
CC_HLR_CDC_03	The regulation shall be done using a proportional and integral algorithm, with Kp and Ki factors.		
CC_HLR_CDC_04	The regulation algorithm shall be protected against the overshoot of its integral part: the integral action shall be reset when the CC is going on, and frozen when the throttle output is saturated.		
CC_HLR_CDC_05	The throttle command shall be saturated at ThrottleSatMax when automatically regulating, in order to limit the car acceleration for the		

PragmaDev Studio - Text files

File Edit Search Preferences Windows Help

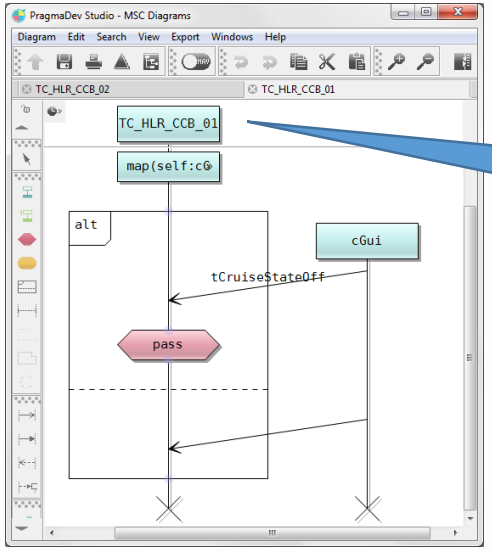
TTCN_TestsAndControl.ttcn3

```
testcase TC_HLR_CCB_01() runs on runsOn_CruiseControl system CruiseControl
 {
  activate(RTDS_fail());
  map(self:cGui, system:cGui);
  alt
    {
    []cGui.receive(tCruiseStateOff)
      {
      setverdict(pass);
      };
    []cGui.receive
      {
      repeat;
      };
    }
  }

testcase TC_HLR_CCB_02() runs on runsOn_CruiseControl system CruiseControl
 {
  activate(RTDS_fail());
  map(self:cGui, system:cGui);
  timer tSetTest;
  var boolean testFinished := false;

  tSetTest.start(1000);
  alt
    {
    []cGui.receive(tCruiseStateOff)
      {
      // Get to a speed above 30 to start the CC
      cGui.send(tAccel90)
      alt
        {
        []cGui.receive(tSpeedAbove30)
          {
          // Activate the CC
          cGui.send(tOnOff)
          // The CC will start when the accelerator is set to 0
          cGui.send(tAccel0)
          alt
            {
            []cGui.receive(tCruiseStateOn)
              {
              setverdict(pass);
              testFinished := true;
              };
            []cGui.receive
              {
              repeat;
              };
            };
          };
        []cGui.receive
          {
          repeat;
          };
        };
      };
    []cGui.receive
      {
      if ( not testFinished )
        {
        repeat;
        }
      };
    }
  }
```

G:\Users\gaudin\dev\Test\FmuTesting\TTCN_TestsAn | line 93 | col 7 | 3016 bytes

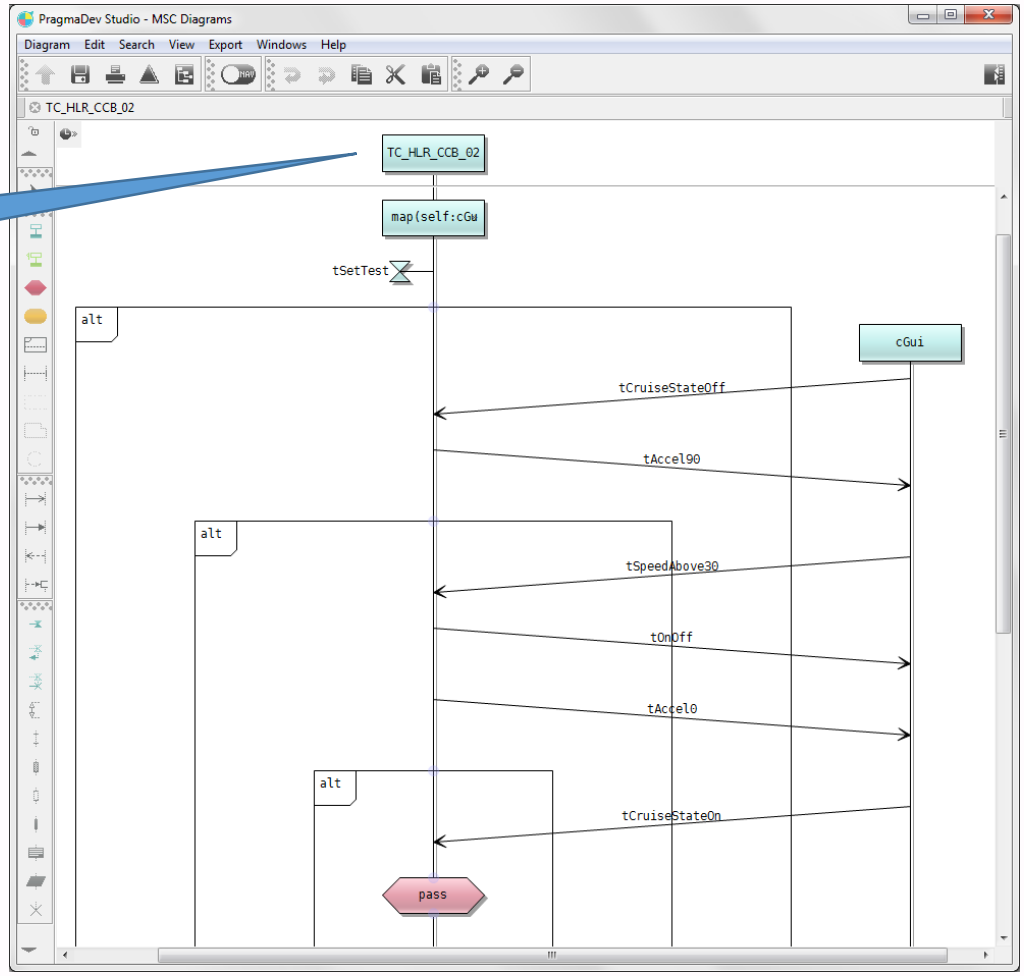
Example

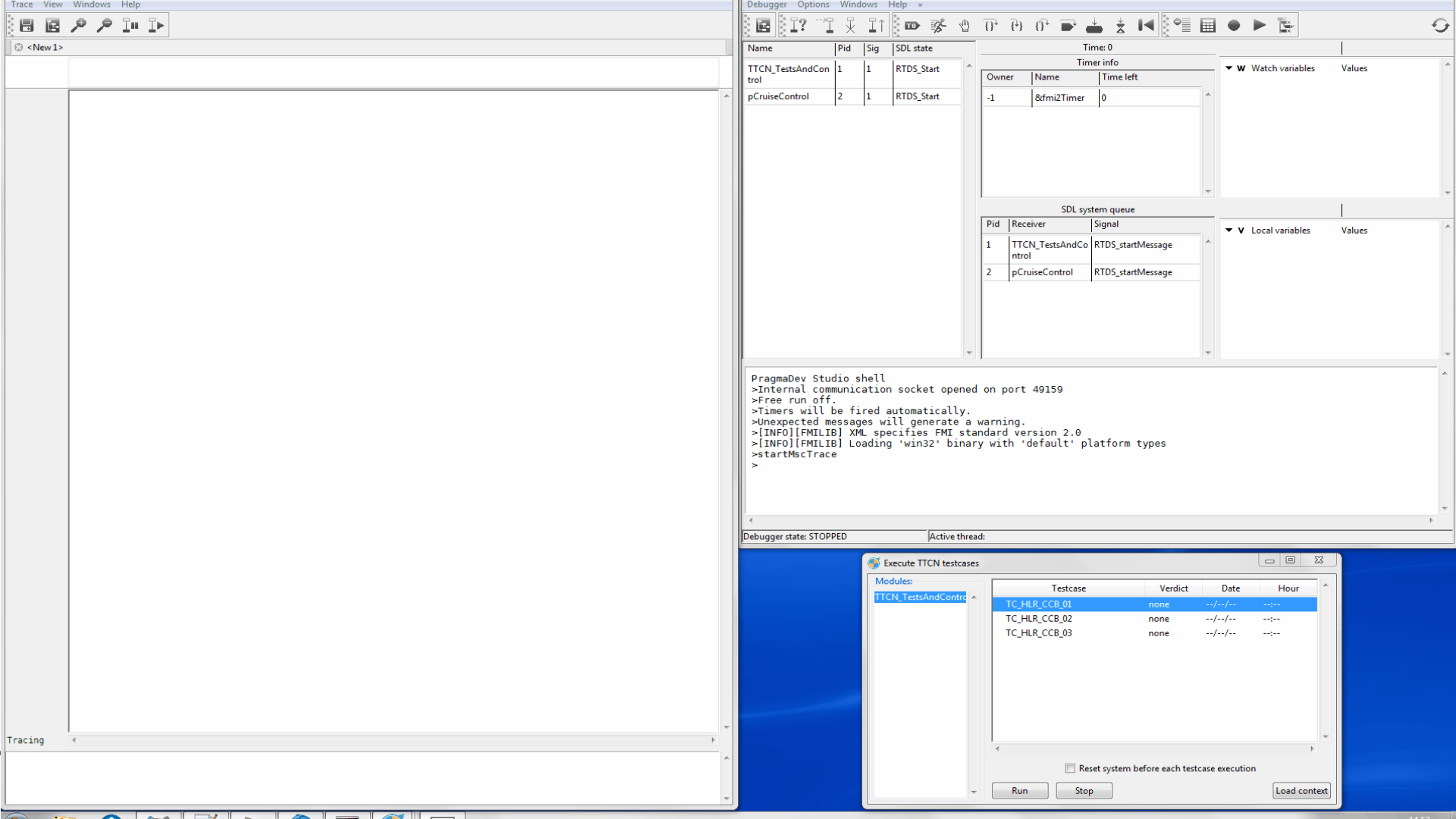
- Graphical view of the test cases



TC_HLR_CCB_01

TC_HLR_CCB_02




Trace View Windows Help
<New 1>
Tracing
Debugger Options Windows Help »
Name Pid Sig SDL state
TTCN_TestsAndControl 1 1 RTDS_Start
pCruiseControl 2 1 RTDS_Start
Time: 0
Timer info
Owner Name Time left
-1 &fmi2Timer 0
SDL system queue
Pid Receiver Signal
1 TTCN_TestsAndControl RTDS_startMessage
2 pCruiseControl RTDS_startMessage
Watch variables Values
Local variables Values
PragmaDev Studio shell
>Internal communication socket opened on port 49159
>Free run off.
>Timers will be fired automatically.
>Unexpected messages will generate a warning.
>[INFO][FMILIB] XML specifies FMI standard version 2.0
>[INFO][FMILIB] Loading 'win32' binary with 'default' platform types
>startMscTrace
>
Debugger state: STOPPED
Active thread:
Execute TTCN testcases
Modules:
TTCN_TestsAndContro
Testcase Verdict Date Hour
TC_HLR_CCB_01 none --/--/-- --:--
TC_HLR_CCB_02 none --/--/-- --:--
TC_HLR_CCB_03 none --/--/-- --:--
Reset system before each testcase execution
Run
Stop
Load context

Conclusion

- Even driven testing is best suited
  - Readable
  - Closer to the requirements
  - Can test any type of system (sync, async, and CPS)