Automatic Interleaving for Testing Distributed Systems

Mihal Brumbulli and Emmanuel Gaudin
PragmaDev

ERTS2
January 2016, Toulouse, France

Introduction

- Constant ever-growing interest for large-scale distributed systems
  - The Internet of Things interconnects billions of smart objects

- Complex applications due to heterogeneity and distribution scale
  - Testing is not a trivial task

Motivation

- Operation of nodes is not isolated
  - Test cases must account for the distribution and interaction between nodes

- Existing test cases have to be adapted to consider distribution
  - Introduce concurrency handling into test cases (need to modify existing test cases)
  - Controlled concurrent execution that deals with all relevant interleavings (need to control execution, e.g., scheduler)

Specification and Description Language (ITU-T)

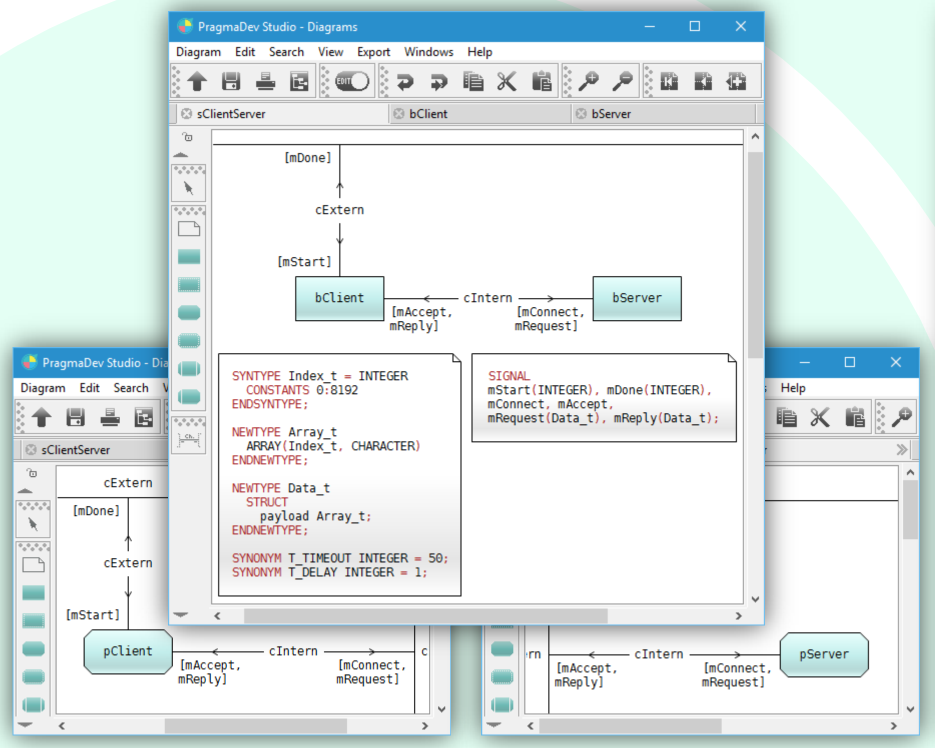
Unified Modeling Language (OMG)

Testing and Test Control Notation Version 3 (ETSI)

TECHNOLOGY

PRAGMADEV
modeling and testing tools

Structure & Behavior



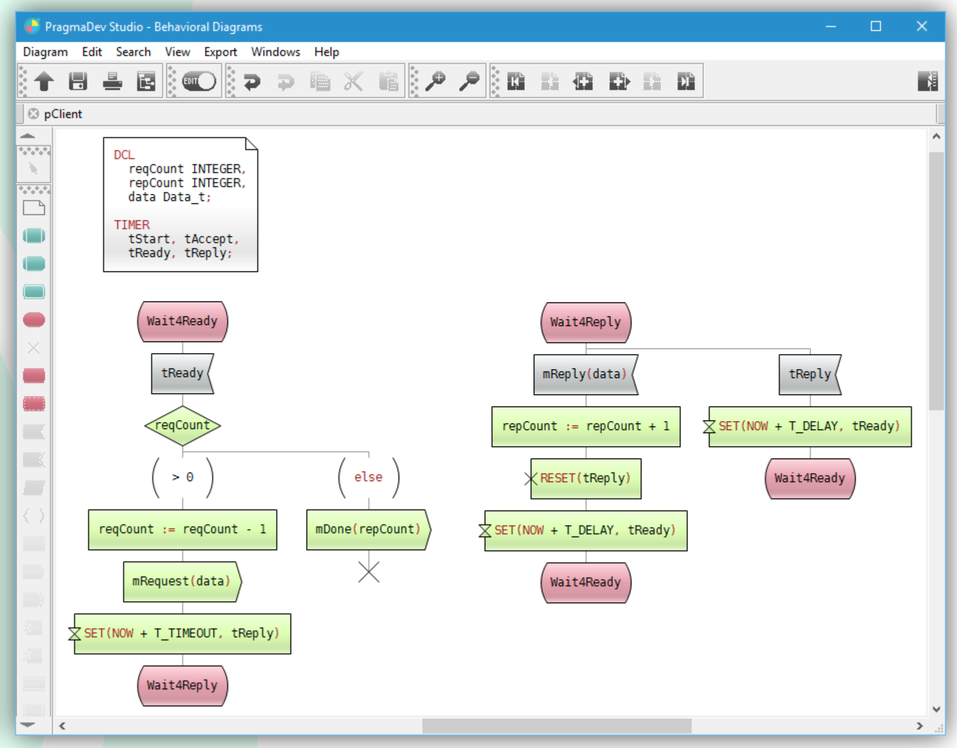
```
SYNTYPE Index_t = INTEGER
  CONSTANTS 0:8192
ENDSYNTYPE;

NEWTYPE Array_t
  ARRAY(Index_t, CHARACTER)
ENDNEWTYPE;

NEWTYPE Data_t
  STRUCT
    payload Array_t;
ENDNEWTYPE;

SYNONYM T_TIMEOUT INTEGER = 50;
SYNONYM T_DELAY INTEGER = 1;
```

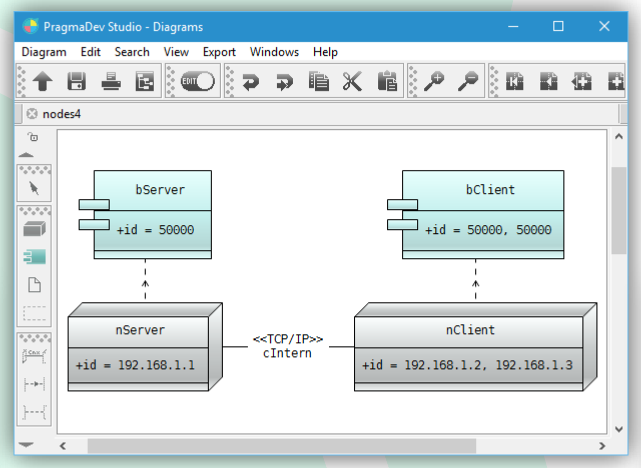
```
SIGNAL
mStart(INTEGER), mDone(INTEGER),
mConnect, mAccept,
mRequest(Data_t), mReply(Data_t);
```

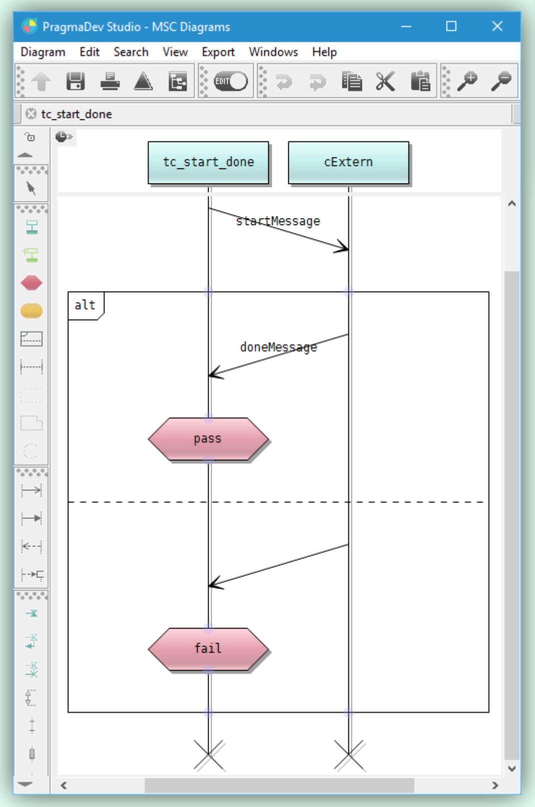


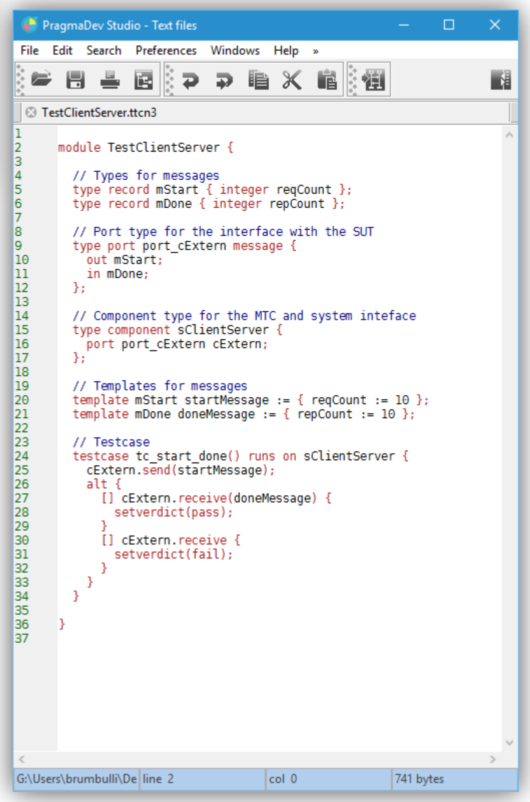
```
DCL
  reqCount INTEGER,
  repCount INTEGER,
  data Data_t;

TIMER
  tStart, tAccept,
  tReady, tReply;
```

Deployment & Test


PragmaDev Studio - Diagrams
Diagram Edit Search View Export Windows Help
nodes4
bServer
+id = 50000
bClient
+id = 50000, 50000
nServer
+id = 192.168.1.1
<<TCP/IP>>
cIntern
nClient
+id = 192.168.1.2, 192.168.1.3


PragmaDev Studio - MSC Diagrams
Diagram Edit Search View Export Windows Help
tc_start_done
tc_start_done
cExtern
startMessage
alt
doneMessage
pass
fail


PragmaDev Studio - Text files
File Edit Search Preferences Windows Help
TestClientServer.ttcn3

```

module TestClientServer {

  // Types for messages
  type record mStart { integer reqCount };
  type record mDone { integer repCount };

  // Port type for the interface with the SUT
  type port port_cExtern message {
    out mStart;
    in mDone;
  };

  // Component type for the MTC and system inteface
  type component sClientServer {
    port port_cExtern cExtern;
  };

  // Templates for messages
  template mStart startMessage := { reqCount := 10 };
  template mDone doneMessage := { repCount := 10 };

  // Testcase
  testcase tc_start_done() runs on sClientServer {
    cExtern.send(startMessage);
    alt {
      [] cExtern.receive(doneMessage) {
        setverdict(pass);
      }
      [] cExtern.receive {
        setverdict(fail);
      }
    }
  }

}

```

What are the effects of distributed execution of test cases?

Rewrite the test cases or execute them in parallel?

Can we simulate parallelism efficiently?

INTERLEAVING

Problem

- Concurrent execution of $\boldsymbol{K}$ test cases
  - with $\boldsymbol{n_i}$ instructions for $\boldsymbol{i = 1, 2, \dots K}$
  - the number of all interleavings is

$$\boldsymbol{I} = \frac{\left(\sum_{i=1}^{K} \boldsymbol{n_i}\right)!}{\prod_{i=1}^{K} (\boldsymbol{n_i}!)}$$

- Concurrent execution of $\boldsymbol{K}$ instances of the same test case
  - with $\boldsymbol{n_i = N} \ \forall \boldsymbol{i}$ instructions
  - the number of all interleavings is

$$\boldsymbol{I} = \frac{(\boldsymbol{KN})!}{(\boldsymbol{N}!)^{\boldsymbol{K}}}$$

- Typical case of the <u>state-explosion problem</u> which makes execution of all interleavings unpractical. However, …

Solution

- Not all interleavings are <u>relevant</u>
  - Distribution may affect behavior <u>only</u> if there is an interaction between nodes
  - If the execution of a test case does not involve any interaction, then distribution will not have any impact

- Interleave execution at <u>critical points</u>
  - instructions that trigger interaction between nodes

Algorithm

- Group the instructions and then interleave execution of the groups
- Each group must include at most one instruction which triggers interaction

$$\underbrace{m_1^0, m_2^1, m_3^0}_{g_1}, \underbrace{m_4^1}_{g_2}, \underbrace{m_5^1, m_6^0}_{g_3}, \underbrace{m_7^1, m_8^0, m_9^0}_{g_4}, \underbrace{m_{10}^1}_{g_5}$$

- m_i^j is an instruction in the test case
  - $i = 1, 2, \dots N$ is the index (relative order) of the instruction,
  - $j = 0, 1$ if the given instruction triggers (or not) any interaction

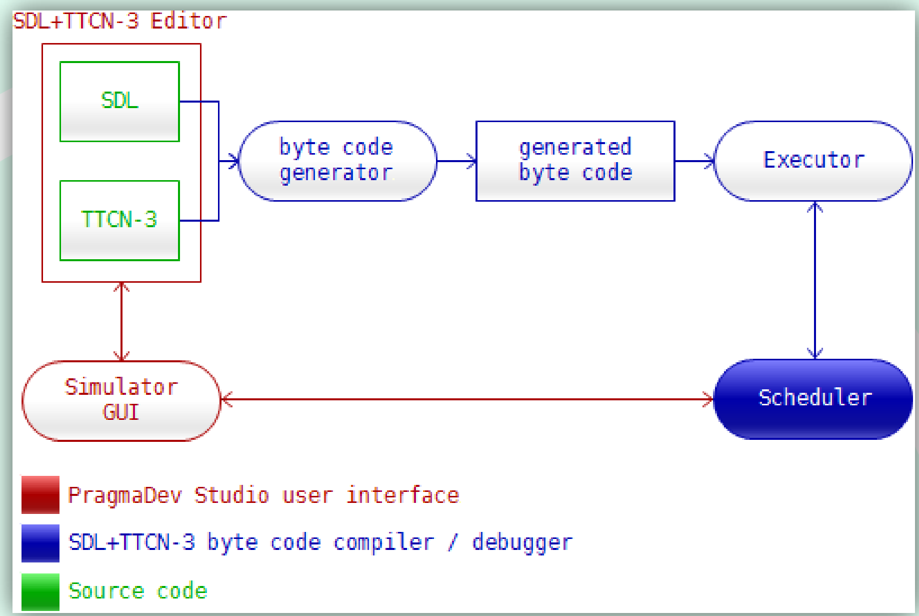
- A group consists of all subsequent m_i^j for which $\sum j \leq 1$

Normal mode: execute test case and mark instructions that trigger interaction based on the deployment diagram

Interleaving mode: automatically generate and execute all interleavings

SIMULATION

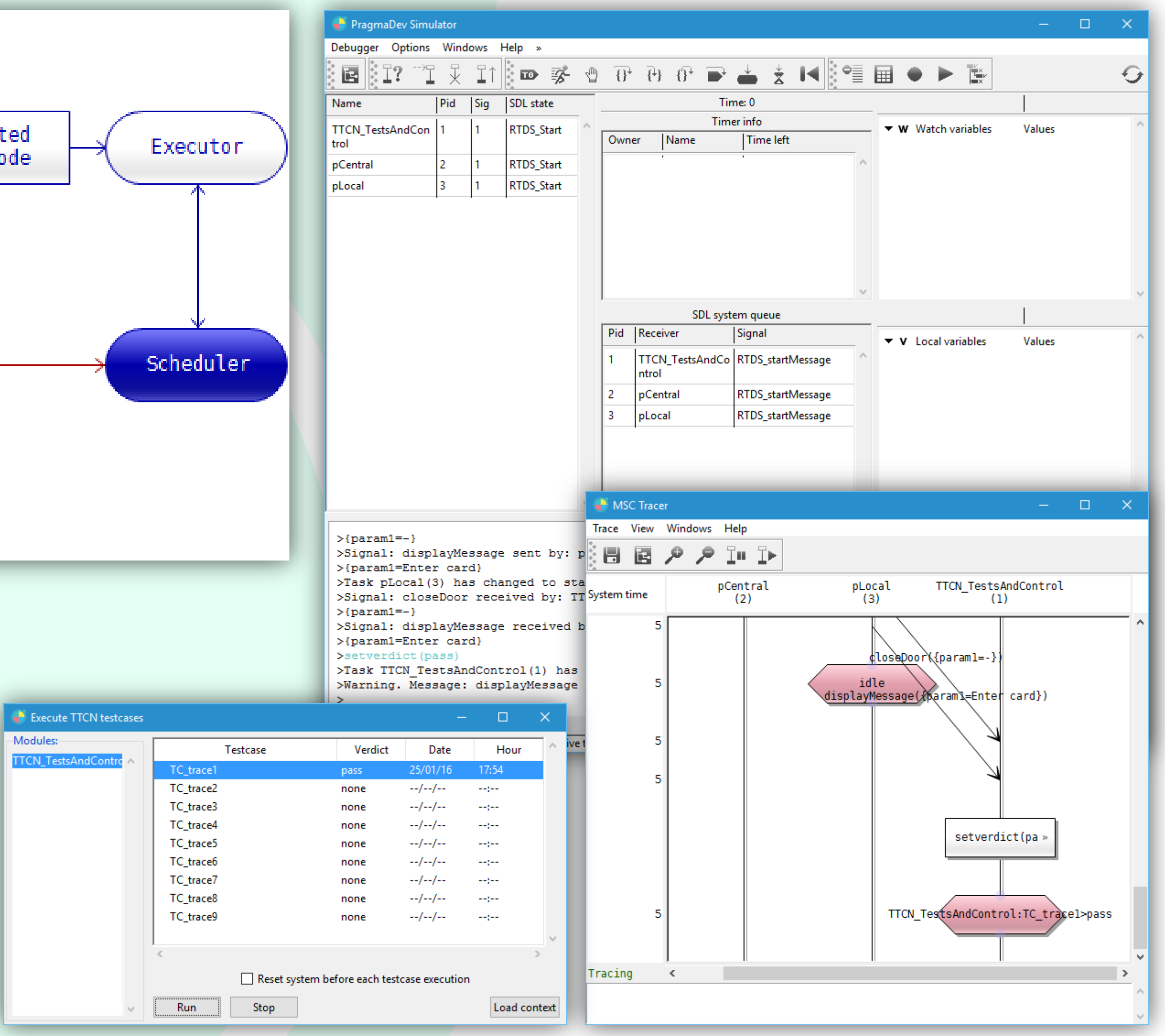
PragmaDev Co-Simulator



🟥 PragmaDev Studio user interface

🟦 SDL+TTCN-3 byte code compiler / debugger

🟩 Source code



Name	Pid	Sig	SDL state
TTCN_TestsAndControl	1	1	RTDS_Start
pCentral	2	1	RTDS_Start
pLocal	3	1	RTDS_Start

Owner	Name	Time left

Pid	Receiver	Signal
1	TTCN_TestsAndControl	RTDS_startMessage
2	pCentral	RTDS_startMessage
3	pLocal	RTDS_startMessage

```
>{param1=-}
>Signal: displayMessage sent by: p
>{param1=Enter card}
>Task pLocal(3) has changed to sta
>Signal: closeDoor received by: TT
>{param1=-}
>Signal: displayMessage received b
>{param1=Enter card}
>setverdict(pass)
>Task TTCN_TestsAndControl(1) has
>Warning. Message: displayMessage
>
```

Testcase	Verdict	Date	Hour
TC_trace1	pass	25/01/16	17:54
TC_trace2	none	--/--/--	--:--
TC_trace3	none	--/--/--	--:--
TC_trace4	none	--/--/--	--:--
TC_trace5	none	--/--/--	--:--
TC_trace6	none	--/--/--	--:--
TC_trace7	none	--/--/--	--:--
TC_trace8	none	--/--/--	--:--
TC_trace9	none	--/--/--	--:--

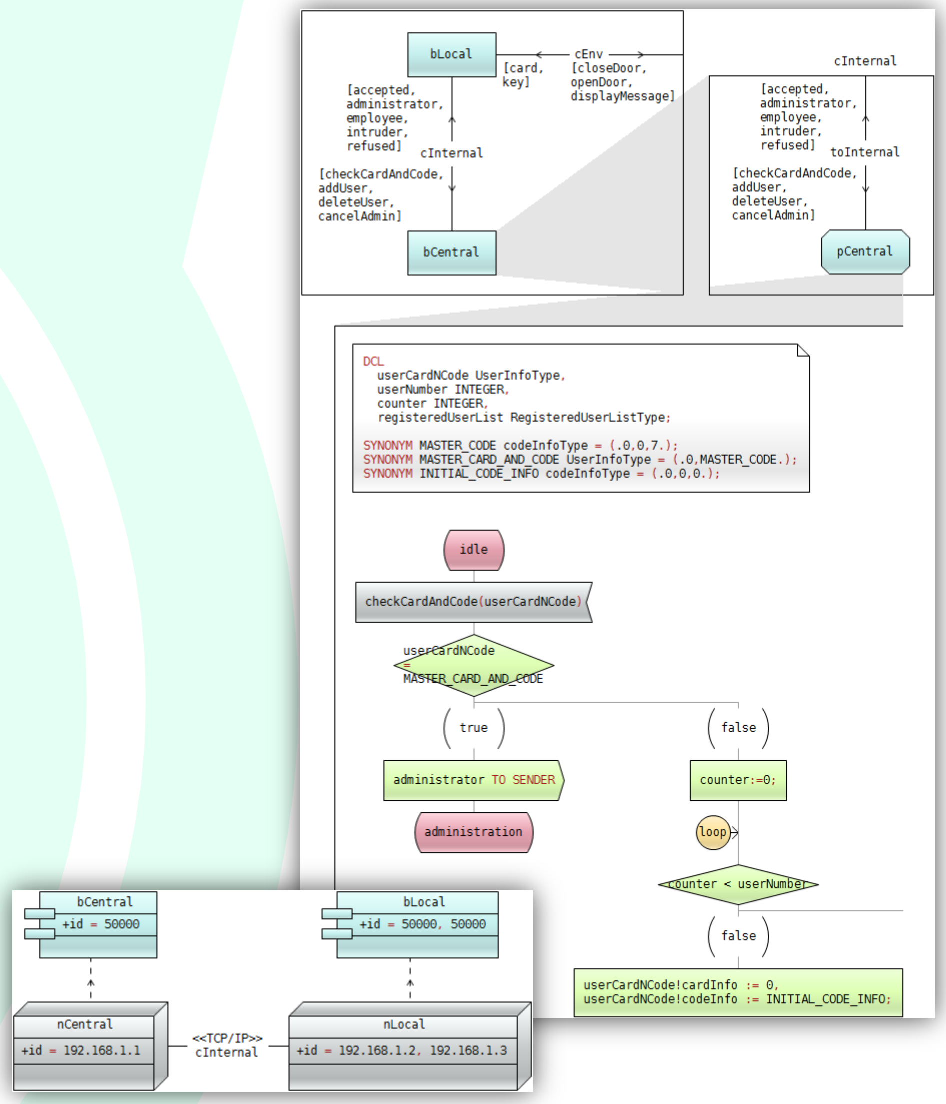
```
System time   pCentral (2)   pLocal (3)   TTCN_TestsAndControl (1)
5             closeDoor({param1=-})
5             idle   displayMessage({param1=Enter card})
5
5
              setverdict(pa »
5             TTCN_TestsAndControl:TC_trace1>pass
```

PRAGMADEV
modeling and testing tools

Example

- Access system has <u>terminals</u> and a <u>central unit</u>
  - Terminal has a slot for the <u>card</u> and a keypad for the <u>key</u>
  - Central unit checks whether access should be granted to a user

- A user can be either <u>administrator</u> or <u>normal</u>


bLocal
cEnv
[card, key]
[closeDoor, openDoor, displayMessage]
[accepted, administrator, employee, intruder, refused]
cInternal
[checkCardAndCode, addUser, deleteUser, cancelAdmin]
bCentral
cInternal
[accepted, administrator, employee, intruder, refused]
toInternal
[checkCardAndCode, addUser, deleteUser, cancelAdmin]
pCentral
idle
checkCardAndCode(userCardNCode)
userCardNCode = MASTER_CARD_AND_CODE
true
false
administrator TO SENDER
counter:=0;
administration
loop
counter < userNumber
false
bCentral
+id = 50000
bLocal
+id = 50000, 50000
nCentral
+id = 192.168.1.1
<<TCP/IP>>
cInternal
nLocal
+id = 192.168.1.2, 192.168.1.3

```
DCL
  userCardNCode UserInfoType,
  userNumber INTEGER,
  counter INTEGER,
  registeredUserList RegisteredUserListType;

SYNONYM MASTER_CODE codeInfoType = (.0,0,7.);
SYNONYM MASTER_CARD_AND_CODE UserInfoType = (.0,MASTER_CODE.);
SYNONYM INITIAL_CODE_INFO codeInfoType = (.0,0,0.);
```

```
userCardNCode!cardInfo := 0,
userCardNCode!codeInfo := INITIAL_CODE_INFO;
```

Example

- Test case: try to get in and out of administrator mode
  - 1 interleaving point; 2 groups
  - 2 terminals; 6 interleavings to execute
  - not much to expect, however…
  - one terminal blocked indefinitely waiting for a reply from the central unit!
- Other 4 problems with the system were identified in the same way

Conclusions

- The algorithm may not always produce significantly less interleavings
  - Degree of interaction between nodes
  - High degree is more an exception than the rule
- Successful application of the approach with a simple example
  - Working on more complex systems
- The approach is based on simulation
  - Cannot be applied (at present) for test cases on real target

PRAGMADEV
modeling and testing tools

Questions?

THANK YOU!