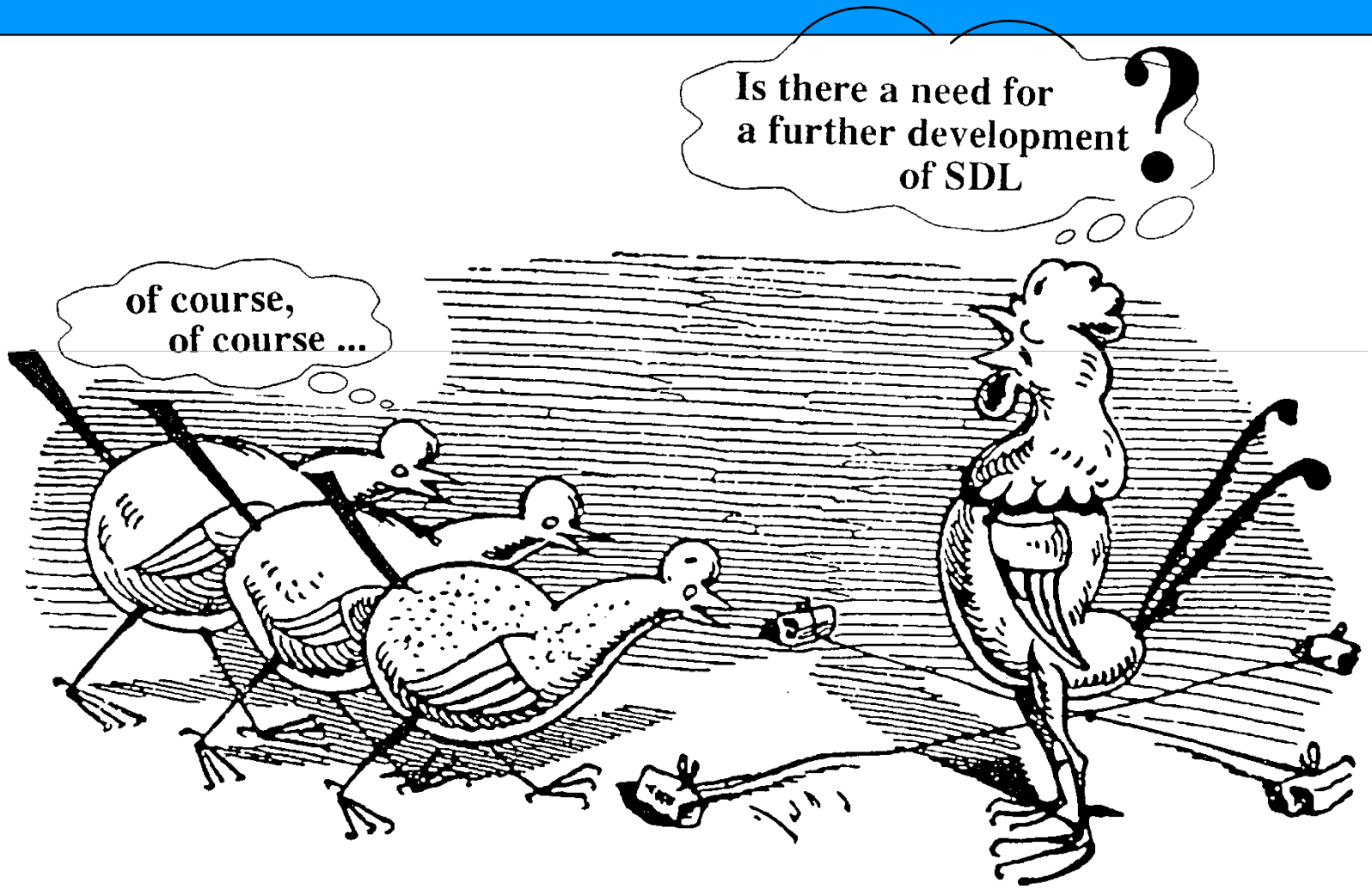
SDL the following ... Part 2

TMSP

Stéphane Maag

TELECOM SudParis

Ready ... ?



Stephane.Maag@telecom-sudparis.eu

Objectives

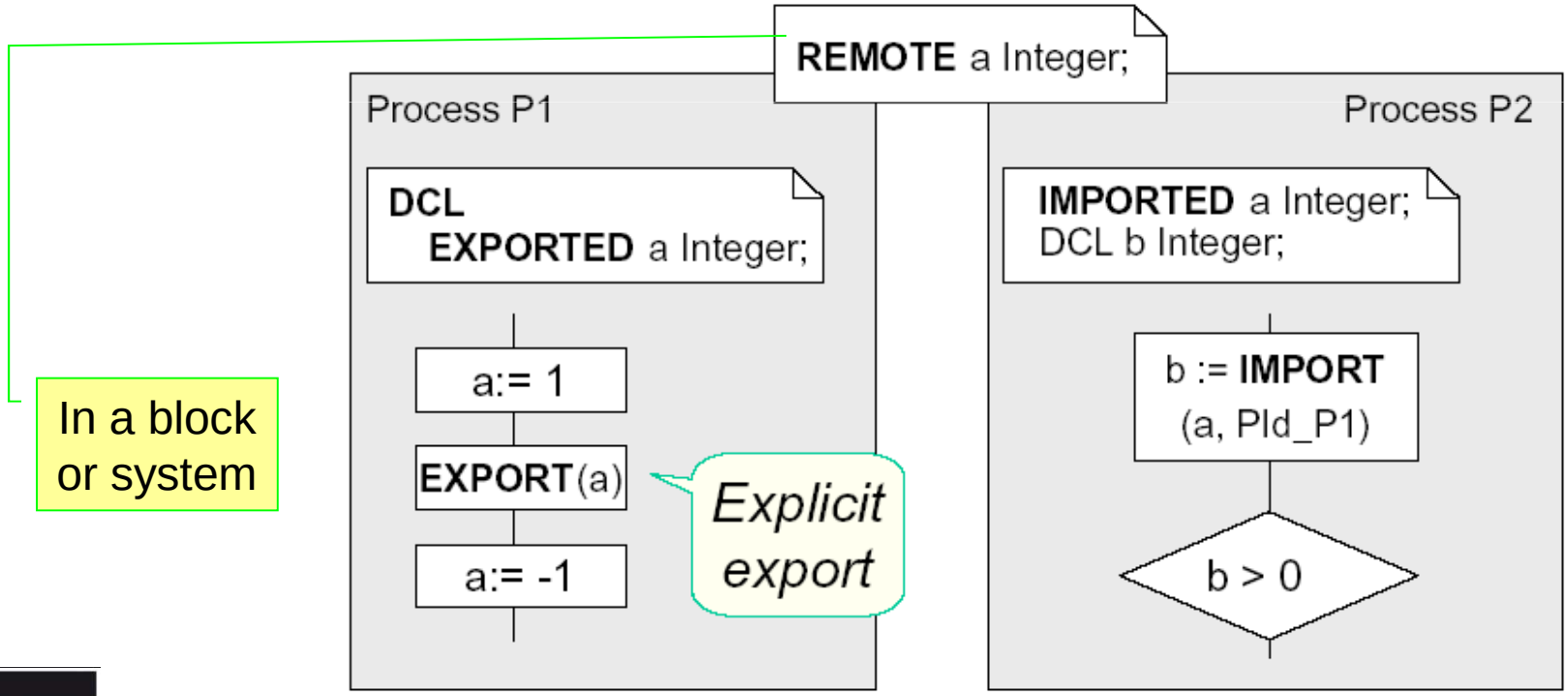
This course intends to make the participants discover:

- Structures
- Structural types
- Packages
- PID
- Procedures
- MacroDefinitions
- ASN.1 - Z.105

Stephane.Maag@telecom-sudparis.eu

Remote Variables

- **EXPORT-IMPORT**: to get the value of a variable of another process (implicit signals)



Definition of Structure

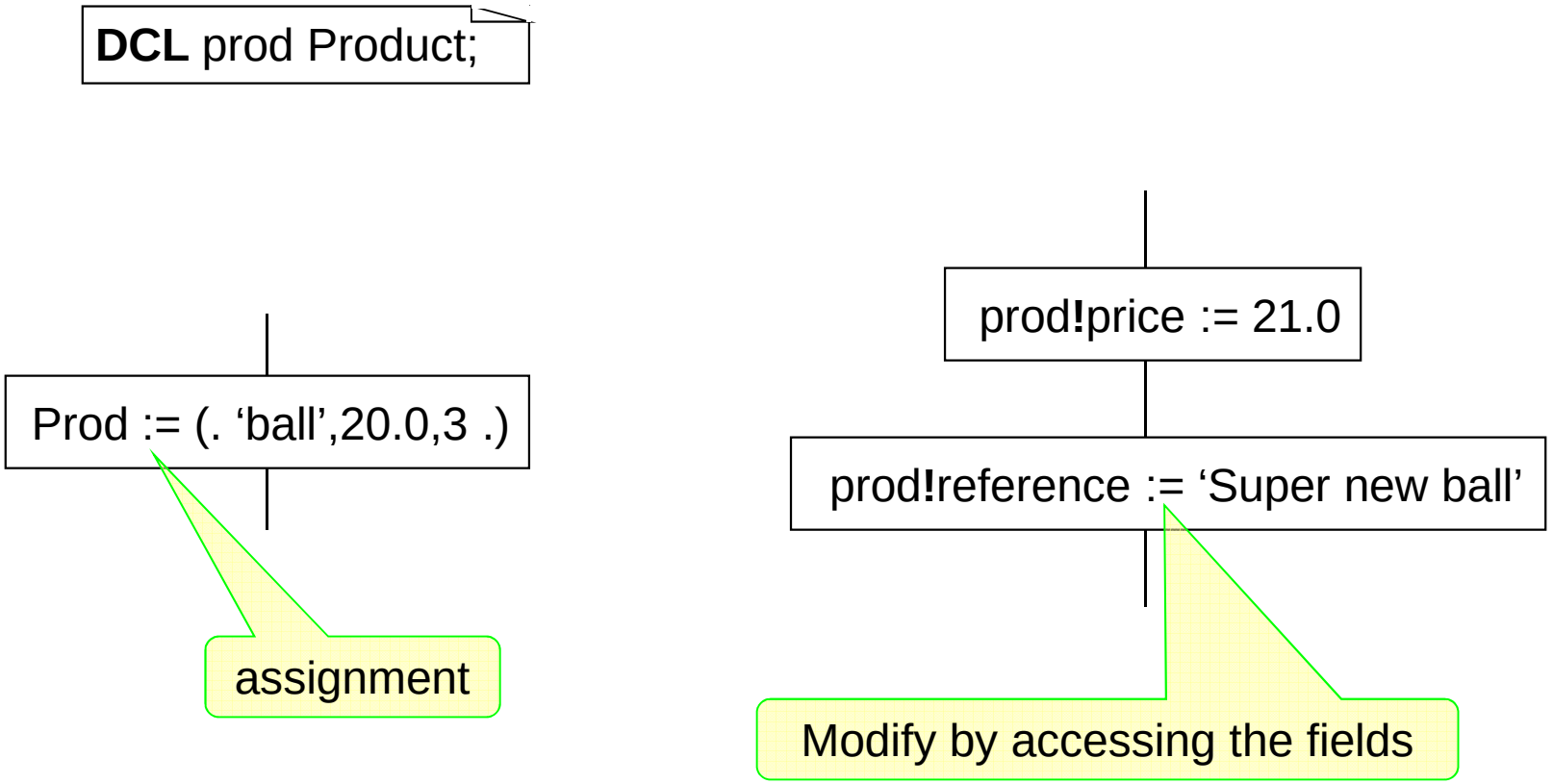
Structure with Fields

```
NEWTYPE Product
STRUCT
        reference CHARSTRING;
        price REAL;
        quantity INTEGER;
ENDNEWTYPE Product;
```

The field types
may be some structures

Stephane.Maag@telecom-sudparis.eu

Use of Structure



Array Type

Type of
the index (**integer)**

NEWTYPE Product_T
ARRAY (**Index_T**,Product)**;**
ENDNEWTYPE;

DCL prod_set Product_T;
DCL prod Product;

Initialization

prod_set := (. (. 'ball',20.0,3 .) .)

Modify

prod_set(1) := prod

Indexes are **integer**

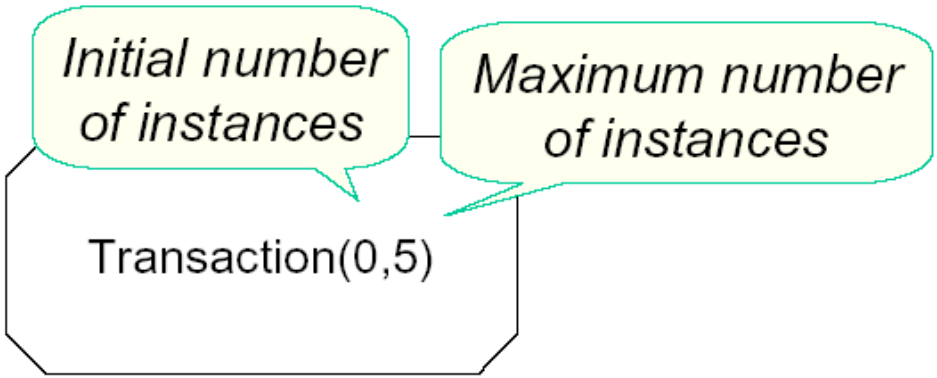
Extract

prod := prod_set(3)

Stephane.Maag@telecom-sudparis.eu

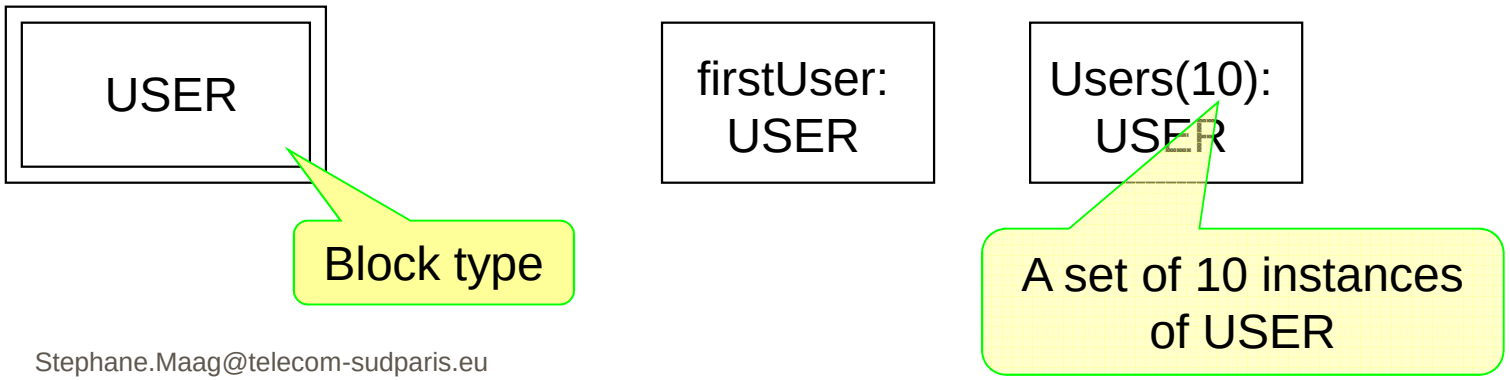
Process: Active Class

- SDL allows to generate process instances:
  - They are active objects
  - Perform their own actions
  - Manage their own data
- Several possible instances may run in parallel.
- To represent them:

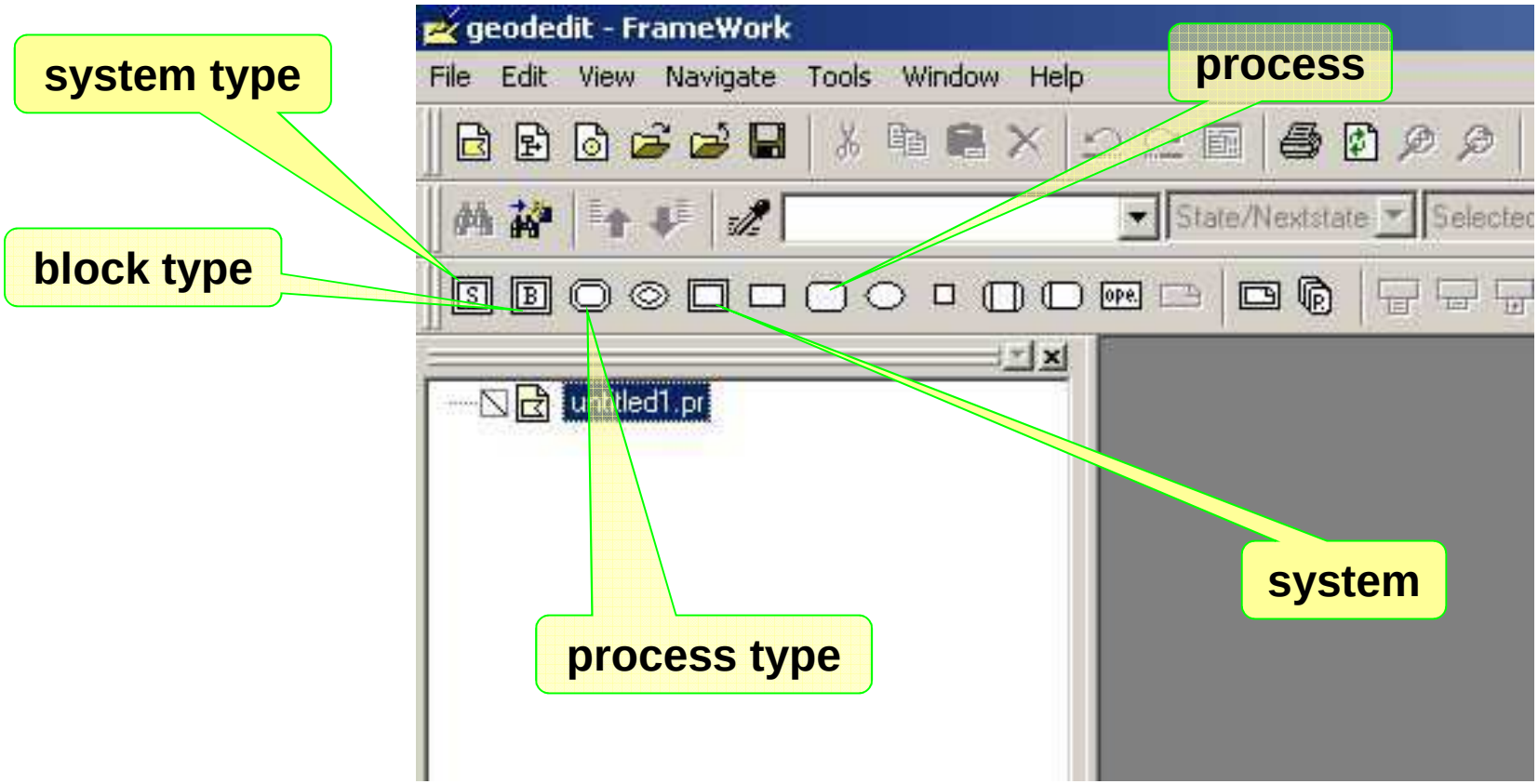


Structural Types and Instances

- **Block type** or **Process type**
  - General description to reuse later
  - Allows to generate many block or process instances
  - Define the content of all instances
- We may define one instance or set of instances



Structural Types in ObjectGEODE



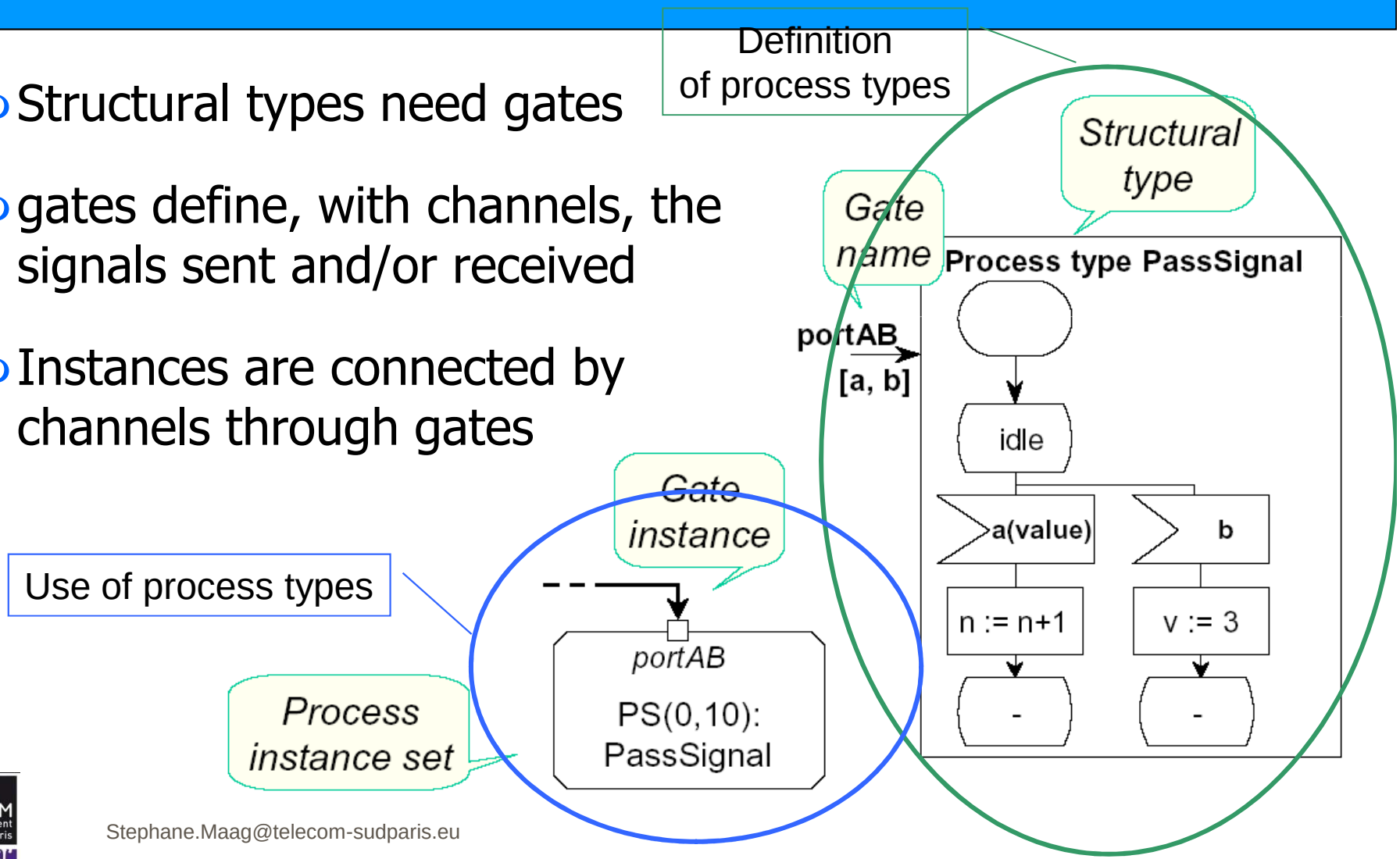
Why Block and Process type ?

- They are defined once and used many times
  - a defined block may then be used in different systems
  - to define only once the content of several processes that run in different blocks
  - structural types and instances available for systems, blocks and processes

Stephane.Maag@telecom-sudparis.eu

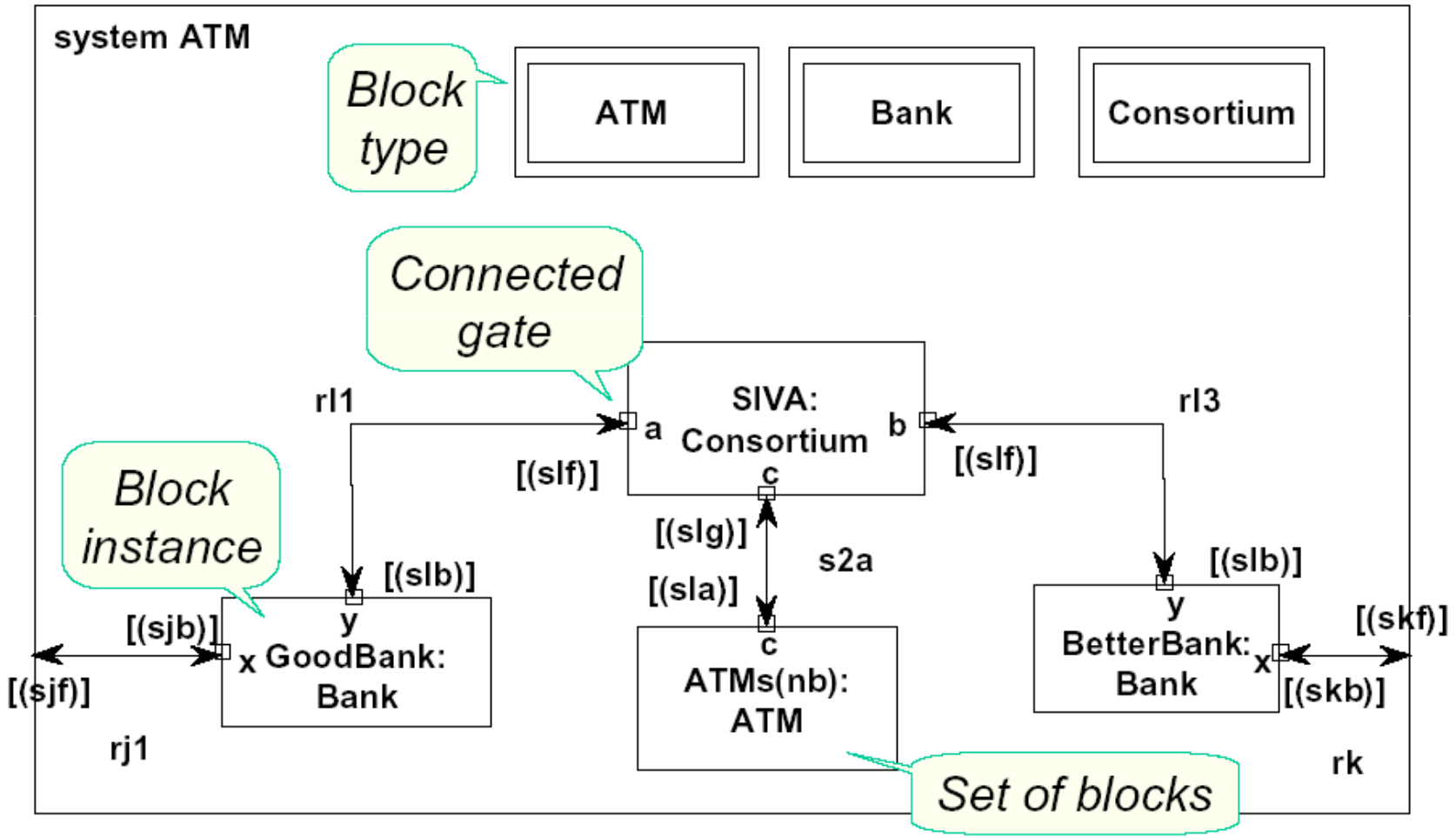
Gates for structural types

- Structural types need gates
- gates define, with channels, the signals sent and/or received
- Instances are connected by channels through gates



Stephane.Maag@telecom-sudparis.eu

Examples of instantiations



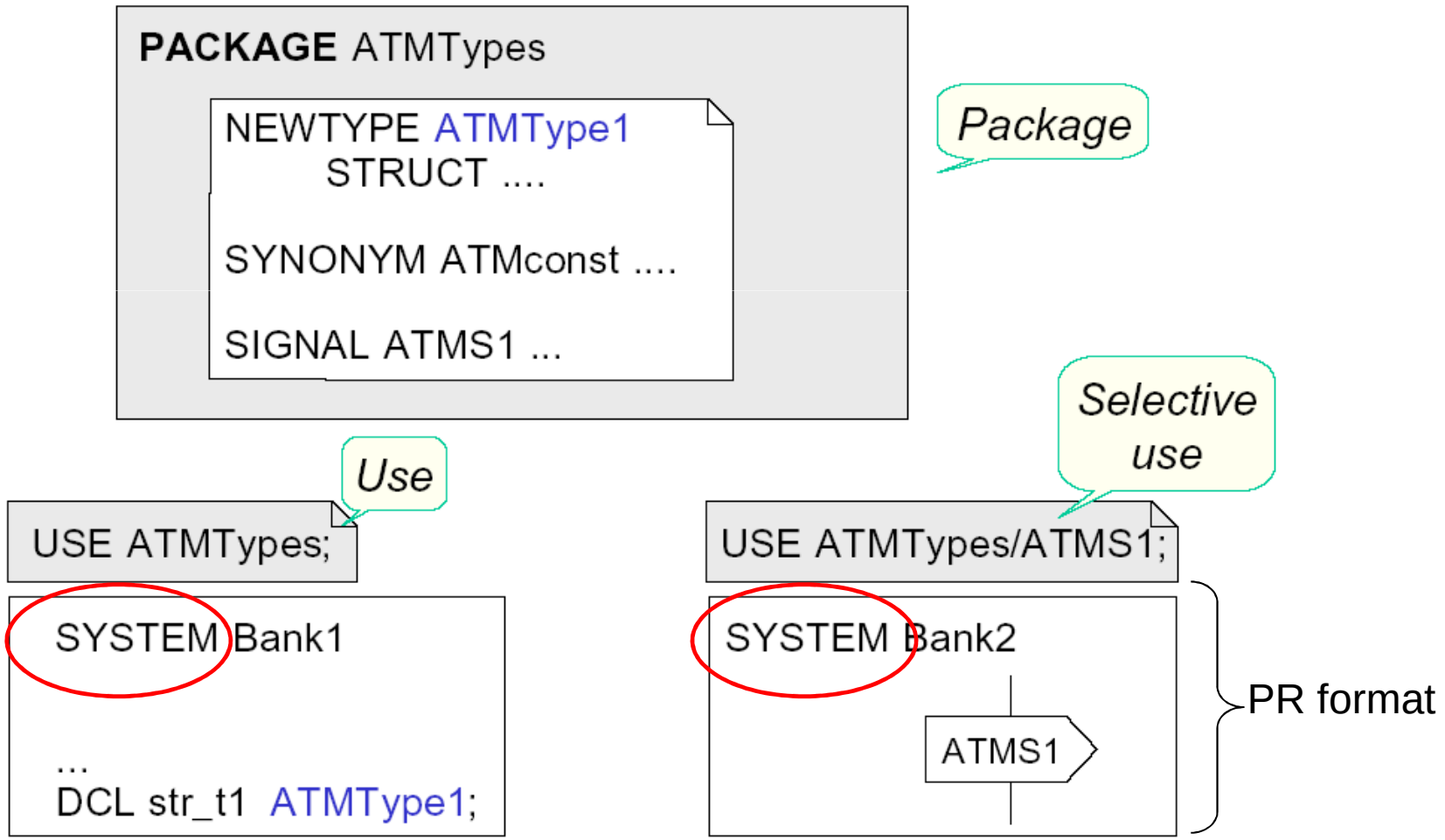
Where to use structural types ?

- In package: a set of types
  - Structural types (block types, process types,…)
  - Signals and lists
  - Constants
  - Data types

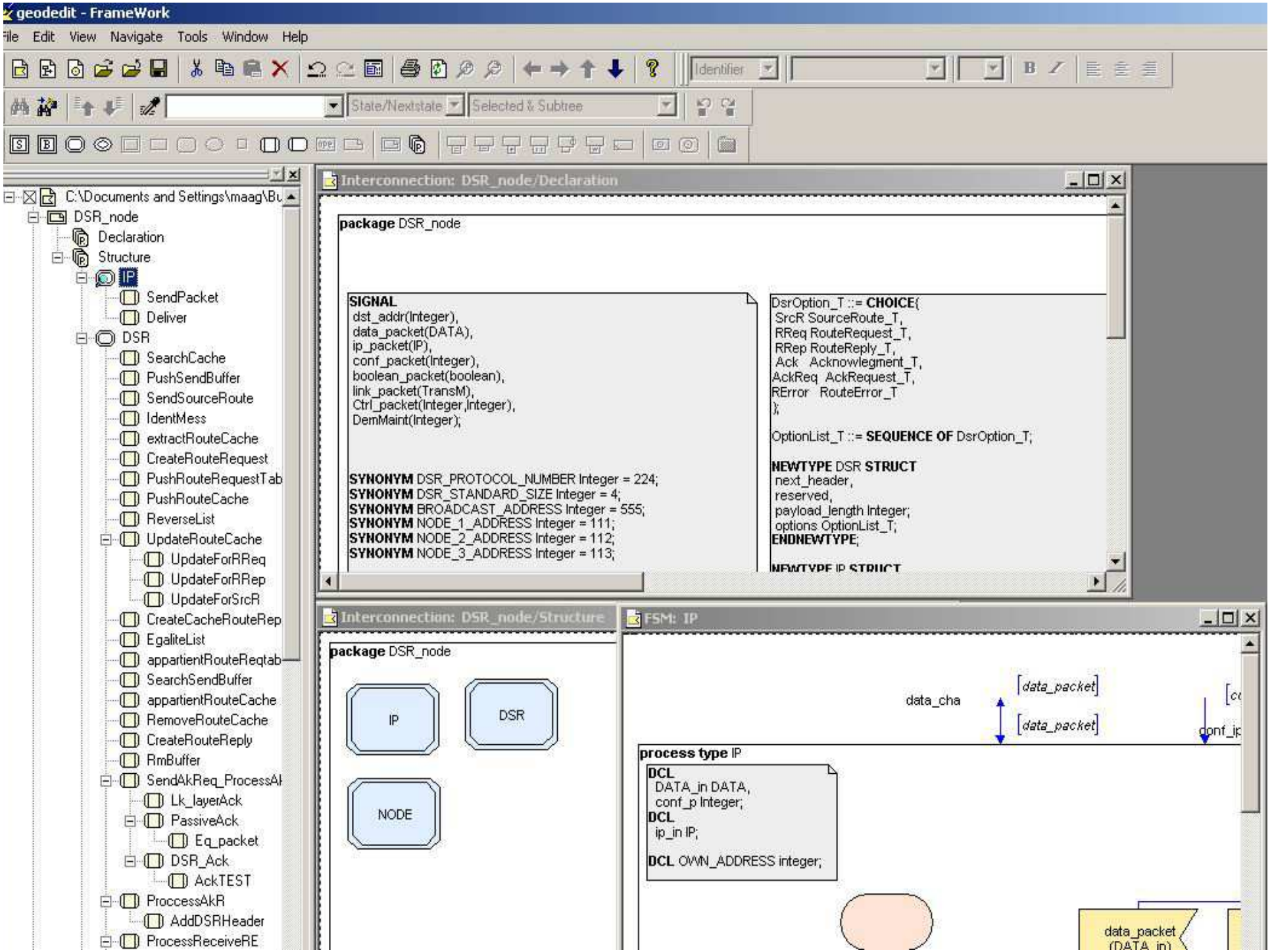
- A package allows to reuse types in several models.

Stephane.Maag@telecom-sudparis.eu

Package example

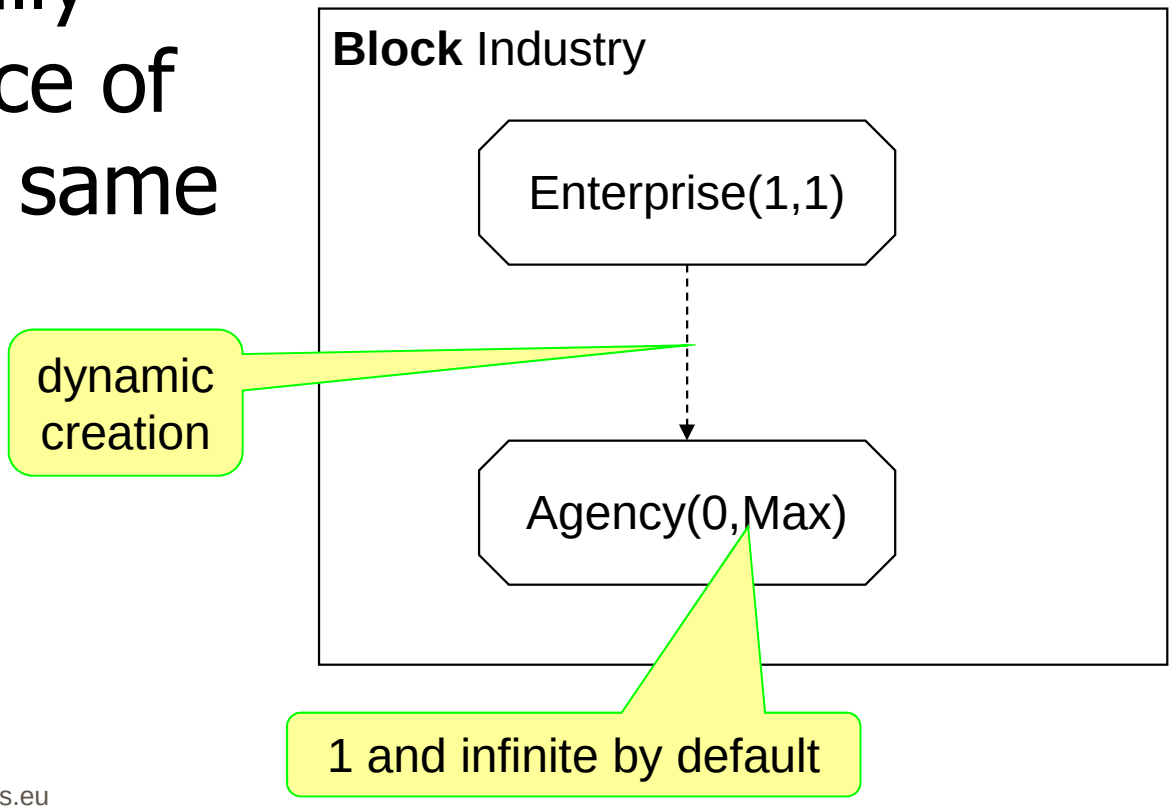


Stephane.Maag@telecom-sudparis.eu


geodedit - FrameWork
File Edit View Navigate Tools Window Help
Identifier
State/Nextstate
Selected & Subtree
C:\Documents and Settings\maag\Bu
DSR_node
Declaration
Structure
IP
SendPacket
Deliver
DSR
SearchCache
PushSendBuffer
SendSourceRoute
IdentMess
extractRouteCache
CreateRouteRequest
PushRouteRequestTab
PushRouteCache
ReverseList
UpdateRouteCache
UpdateForRReq
UpdateForRRep
UpdateForSrcR
CreateCacheRouteRep
EgaliteList
appartientRouteReqtab
SearchSendBuffer
appartientRouteCache
RemoveRouteCache
CreateRouteReply
RmBuffer
SendAkReq_ProcessAk
Lk_layerAck
PassiveAck
Eq_packet
DSR_Ack
AckTEST
ProccessAkR
AddDSRHeader
ProcessReceiveRE
Interconnection: DSR_node/Declaration
package DSR_node
SIGNAL
dst_addr(Integer),
data_packet(DATA),
ip_packet(IP),
conf_packet(Integer),
boolean_packet(boolean),
link_packet(TransM),
Ctrl_packet(Integer,Integer),
DemMaint(Integer);
SYNONYM DSR_PROTOCOL_NUMBER Integer = 224;
SYNONYM DSR_STANDARD_SIZE Integer = 4;
SYNONYM BROADCAST_ADDRESS Integer = 555;
SYNONYM NODE_1_ADDRESS Integer = 111;
SYNONYM NODE_2_ADDRESS Integer = 112;
SYNONYM NODE_3_ADDRESS Integer = 113;
DsrOption_T ::= CHOICE{
SrcR SourceRoute_T,
RReq RouteRequest_T,
RRep RouteReply_T,
Ack Acknowlegment_T,
AckReq AckRequest_T,
RError RouteError_T
};
OptionList_T ::= SEQUENCE OF DsrOption_T;
NEWTYPE DSR STRUCT
next_header,
reserved,
payload_length Integer;
options OptionList_T;
ENDNEWTYPE;
Interconnection: DSR_node/Structure
package DSR_node
IP
DSR
NODE
FSM: IP
data_cha
[data_packet]
[data_packet]
process type IP
DCL
DATA_in DATA,
conf_p Integer;
DCL
ip_in IP;
DCL OWN_ADDRESS integer;
data_packet
(DATA_in)

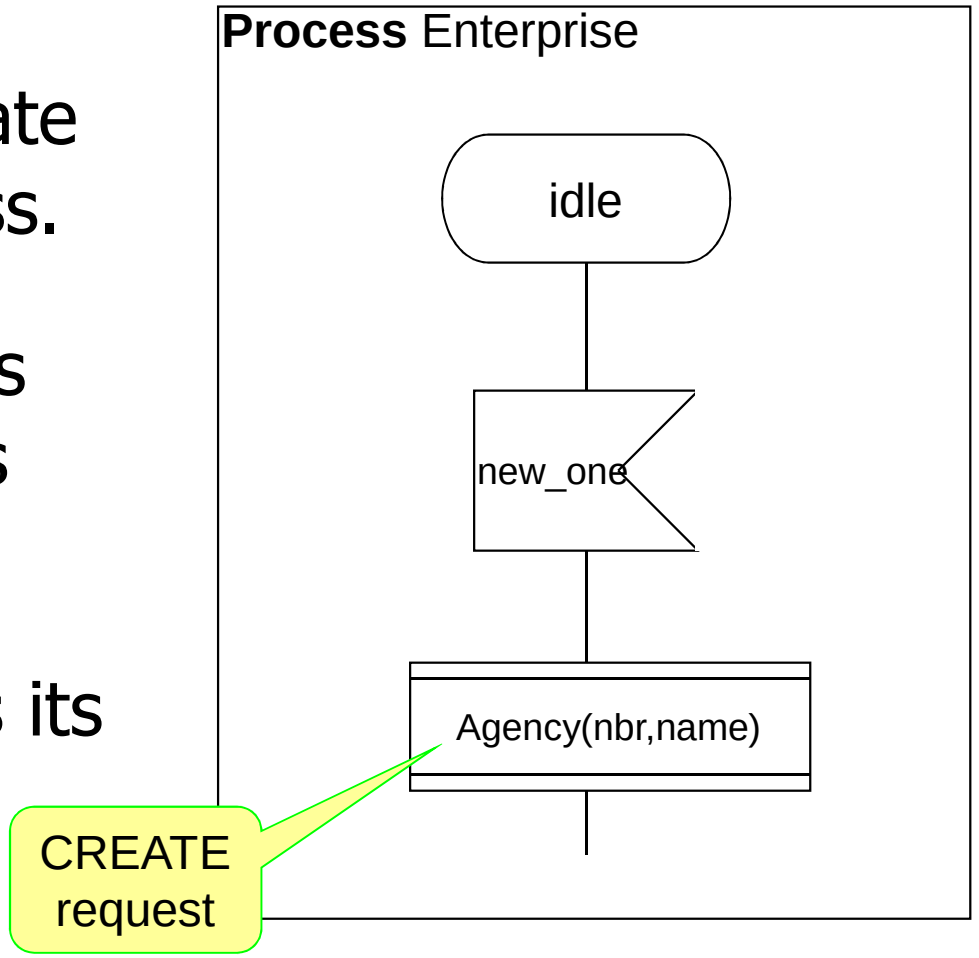
Dynamic Creation of Processes

- Process instance may dynamically creates instance of process in the same block.



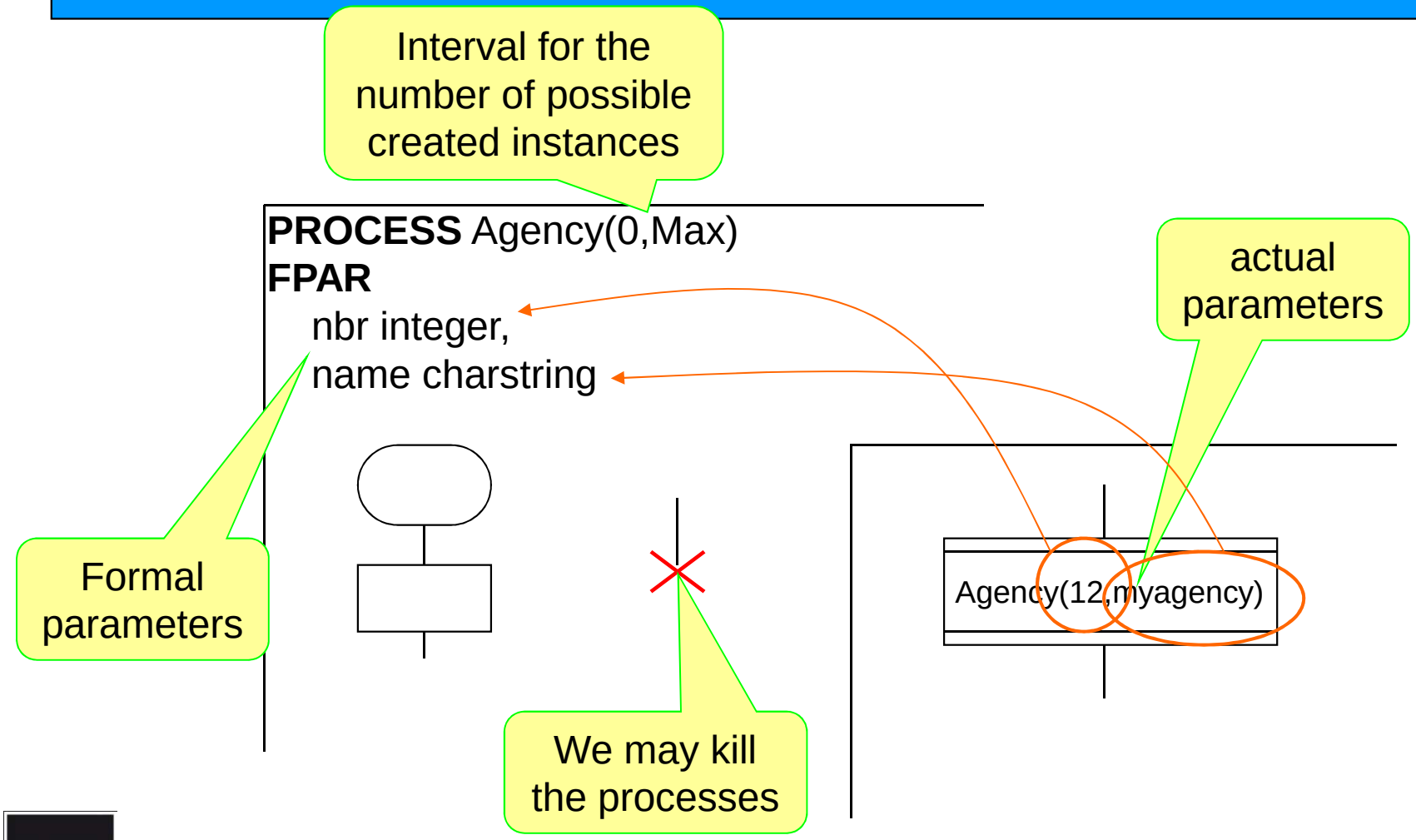
How to dynamically create processes

- The CREATE request provokes the immediate creation of the process.
- The created processes may carry parameters given by the creator.
- The new instance has its own new PID



Stephane.Maag@telecom-sudparis.eu

Process parameters



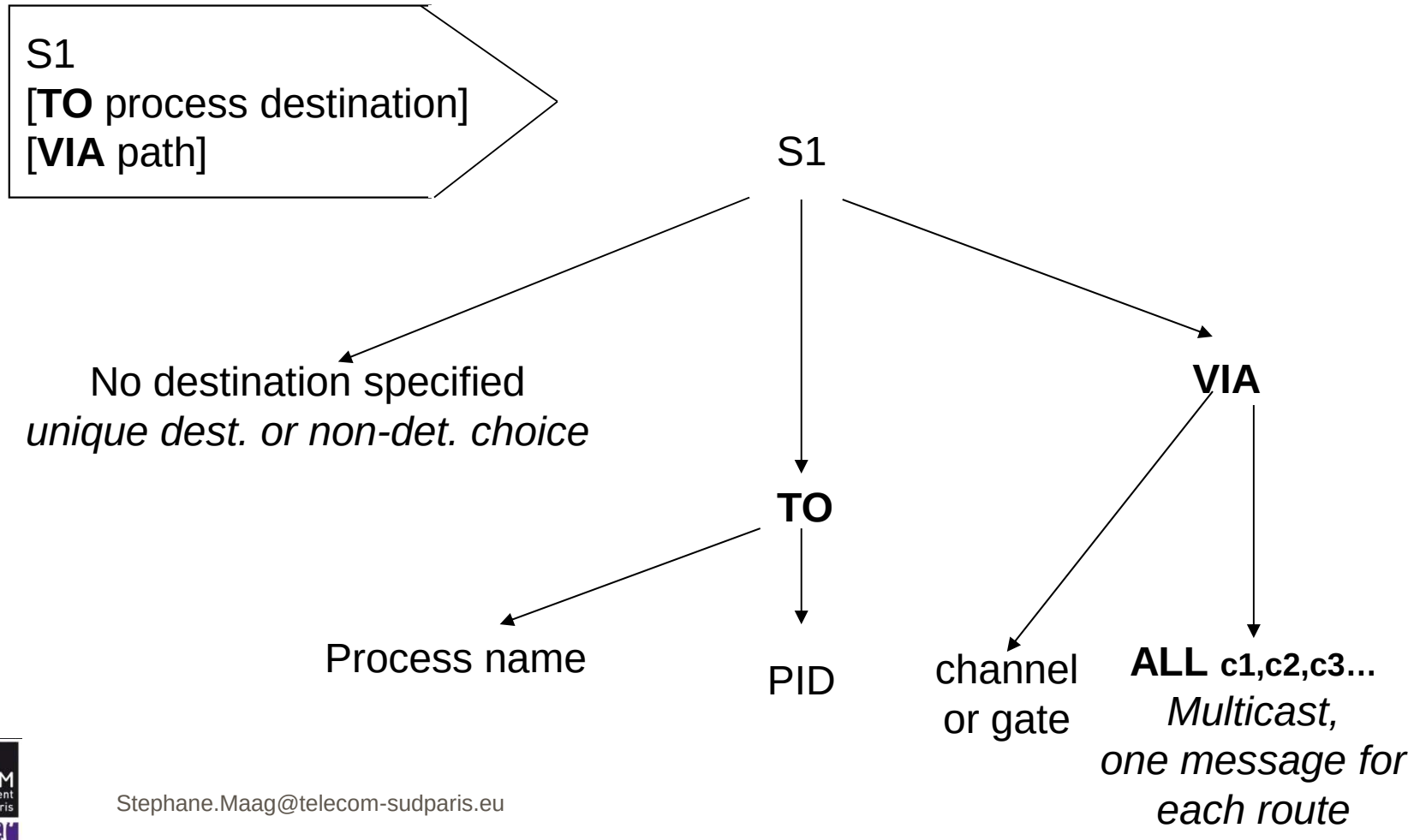
Process IDentification

- The PID is the unique identifier of each instance of process.
  - Remember … PID is a predefined type !
  - The PID cannot be modified
  - The PID type has one predefined constant: NULL
- PIDs are used for communication in case of many possible receivers.
  - Client/server, mobile topologies (broadcast),…
  - signals that are both sent and received, …
  - …

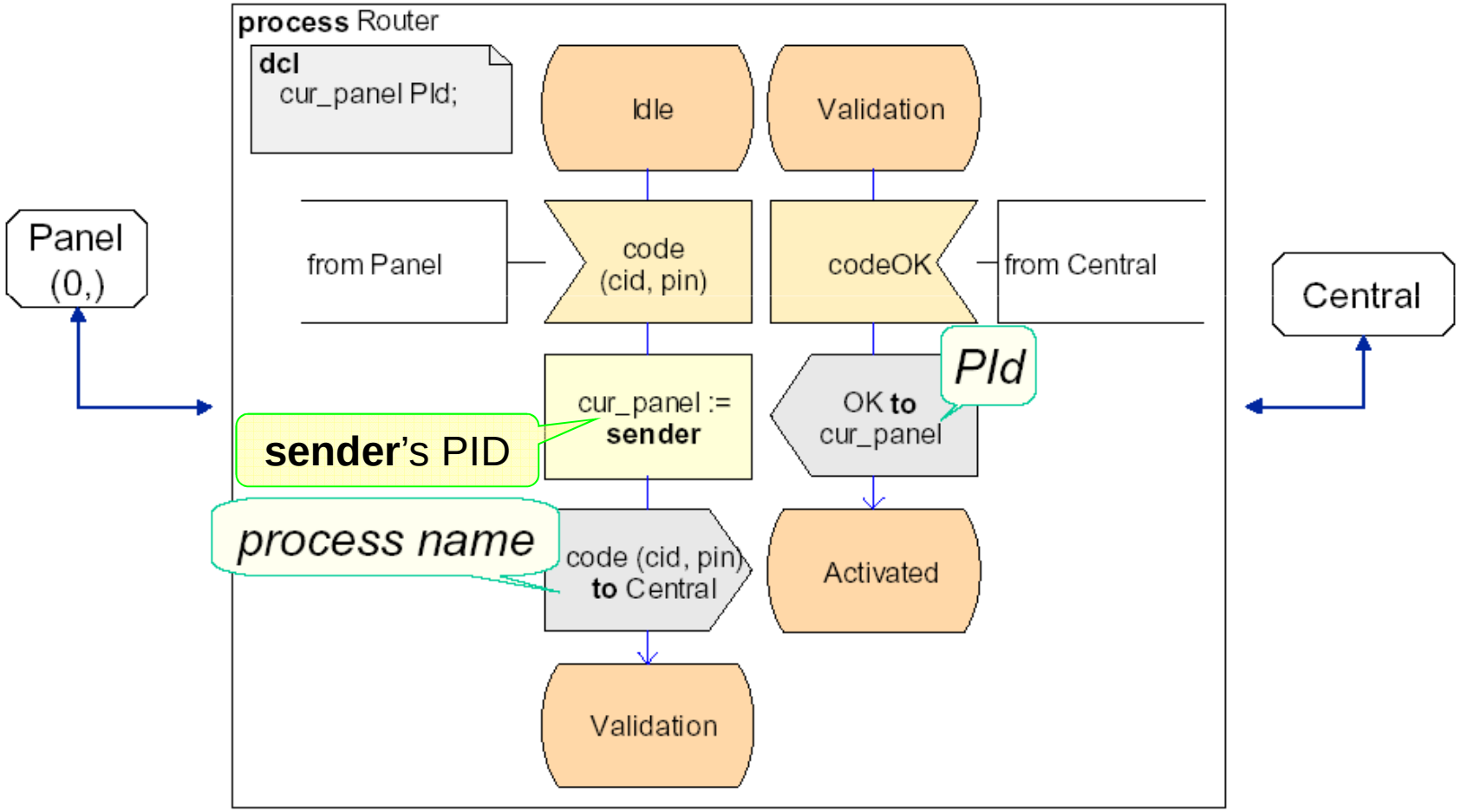
Predefined PID expressions

- SELF: PID of the process itself
- OFFSPRING: last process instance created by itself. If none was created then OFFSPRING is NULL
- PARENT: PID of parent process. If SELF was not dynamically created, then PARENT is NULL
- SENDER: PID of the process that has sent the last consumed signal by SELF. If no consumed signal the SENDER is NULL

Process destination for Output



Example of PID use



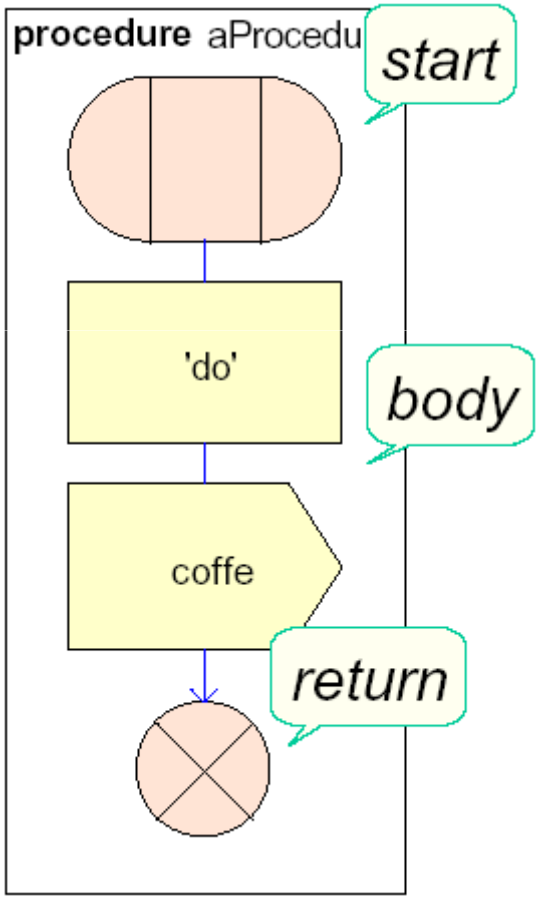
Procedures

- Use to factorize and parameterize actions
- Encapsulation, abstraction
- Allow to reduce the EFSM size
- **Executed in their owning process**

Stephane.Maag@telecom-sudparis.eu

Procedure definition

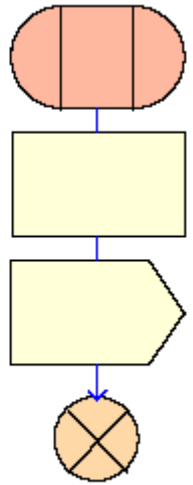
- It is described like process, using EFSM
- It is executed using the queue of its own process
- It does not have any PID (even the one of the process)
- Signals are sent to process only
- It may have **local variables**
- It can be **defined** at process, block or system level
- No Stop in a procedure



Procedure parameters

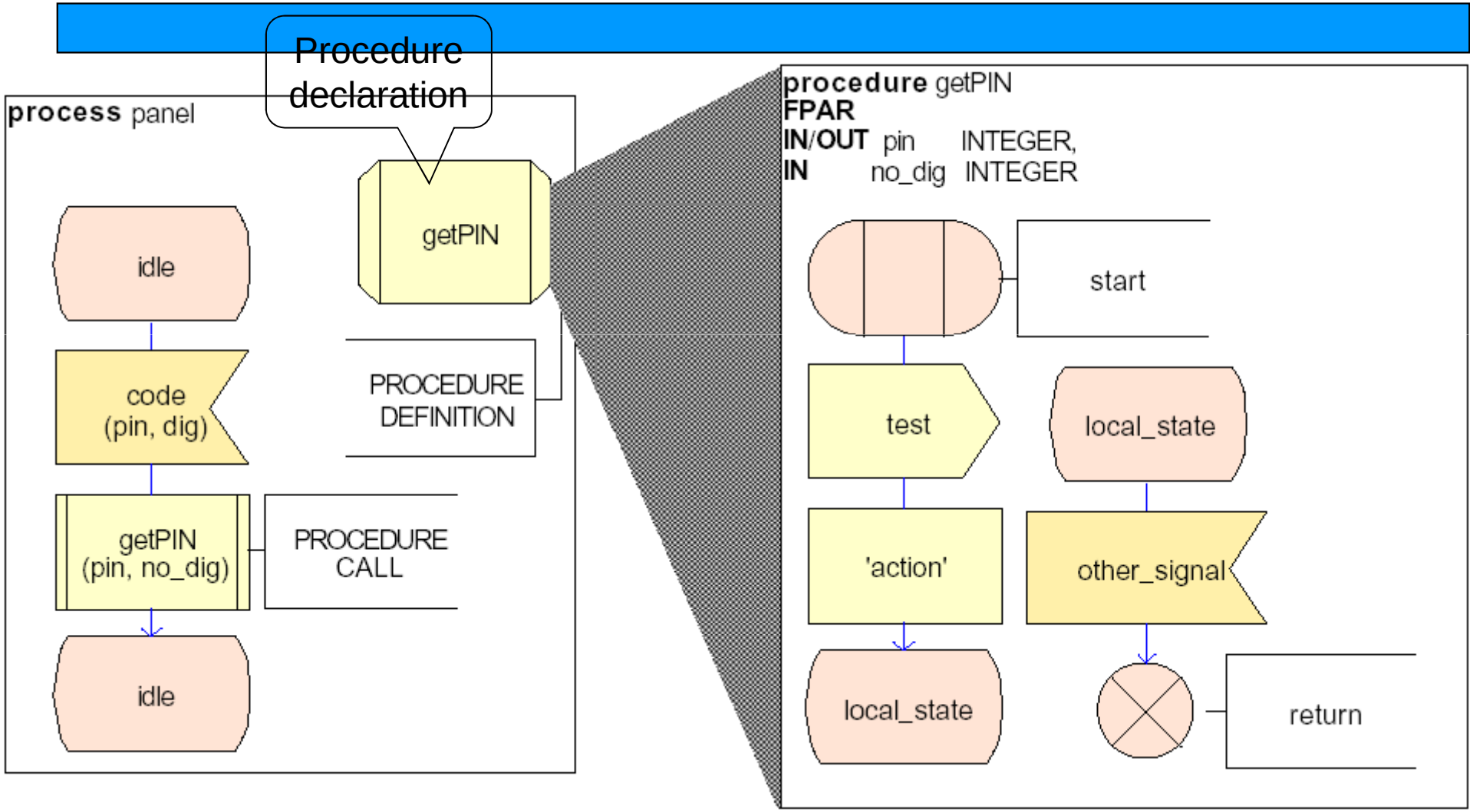
- Introduced by **FPAR** and **IN** or **IN/OUT**:
  - **FPAR**: parameters of the procedure
  - **IN/OUT**:
    - by reference
    - it means that the parameter may be modified
  - **IN**:
    - by value
    - it means that the caller may not see the changes
  - by default the parameter are **IN**.

```
PROCEDURE GetPIN
FPAR
    IN/OUT pin integer,
    IN no_dig integer
```



Stephane.Maag@telecom-sudparis.eu

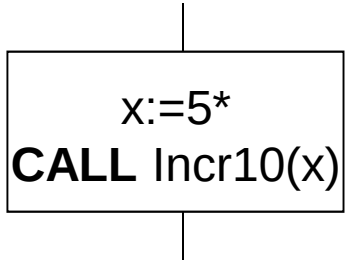
Procedure Example

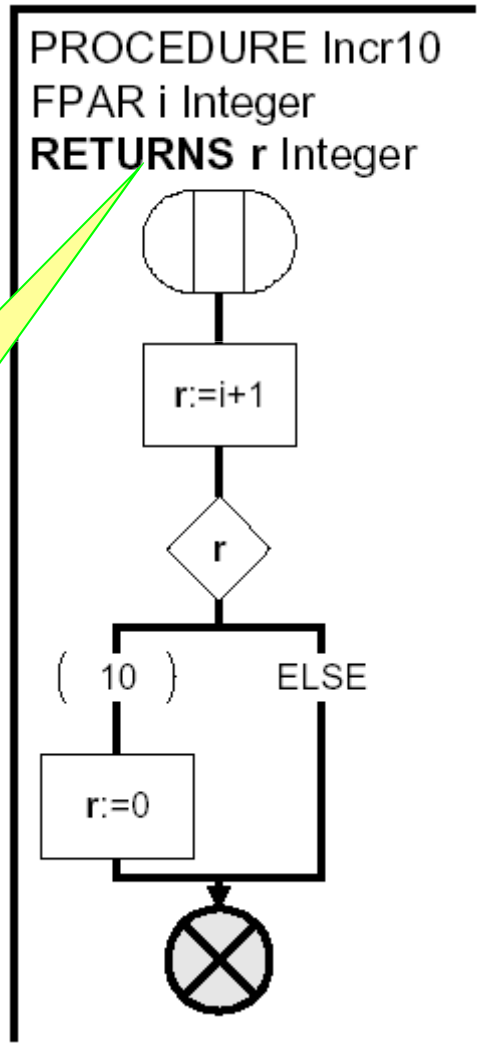


Stephane.Maag@telecom-sudparis.eu

Procedures as classic functions

- They may be called as classic functions in expressions: allows to return explicit values.





a value return

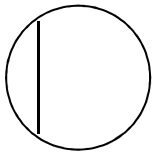
Stephane.Maag@telecom-sudparis.eu

MacroDefinition

- The Macros allow to treat the repetition of code, a description, a behavior that is often repeated,
- Used only within processes or procedures,
- May have formal parameter, it is necessary to transmit them.

Call of a macro

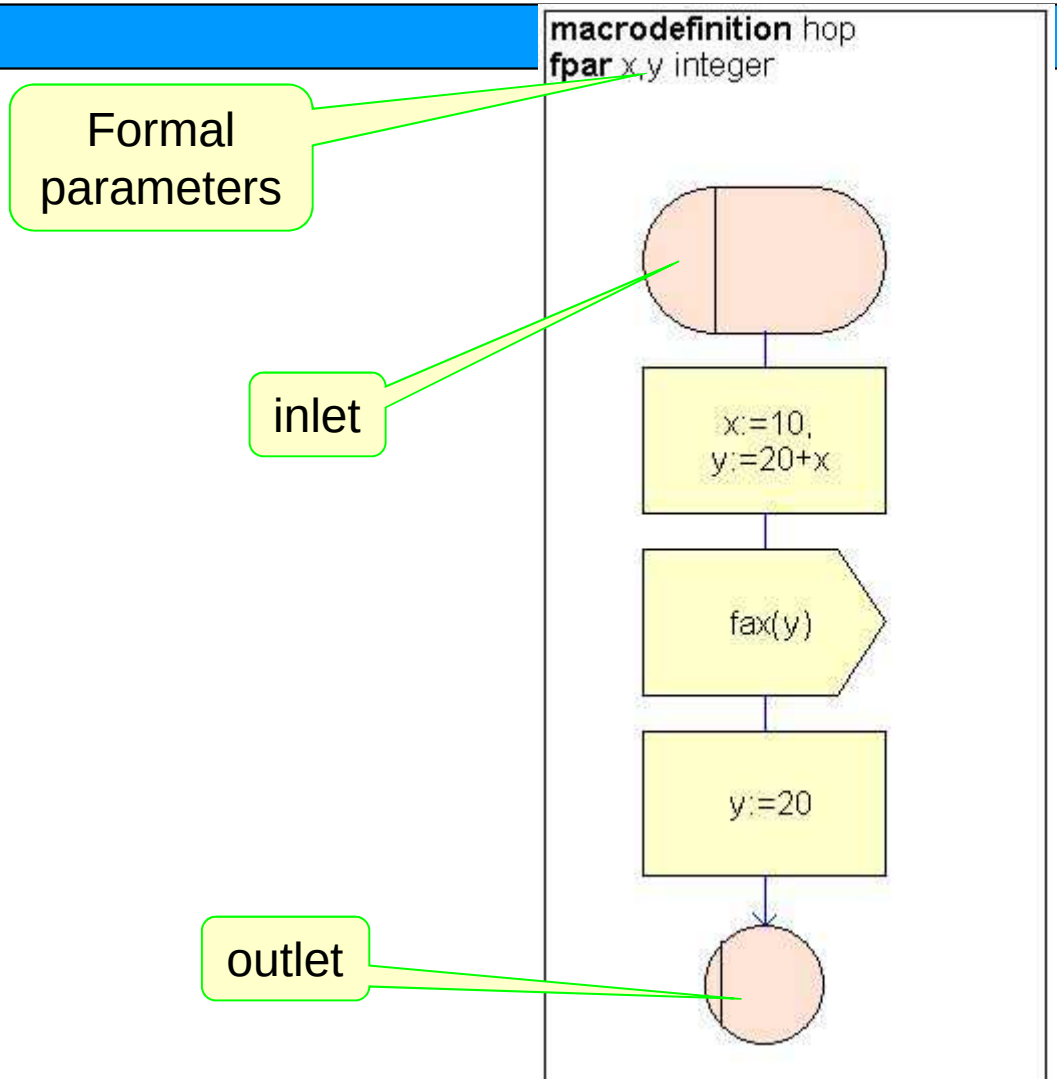
Macro inlet symbol



Macro outlet symbol

Stephane.Maag@telecom-sudparis.eu

Macrodefinition Example



Stephane.Maag@telecom-sudparis.eu

ASN.1 and SDL

- Z.105: Inclusion of ASN.1 in SDL
- Standard: ASN.1 is widely used in standards and can be part of the requirements
- Technically, ASN.1 allows to focus only on data: values, set of values, …

Use of ASN.1 in Z.105

- ASN.1 declarations located in same box than SDL ones,
- ASN.1 and SDL declaration may be mixed,
- Z.105 is not case-sensitive,
- hyphens (“-”) cannot be used.

Stephane.Maag@telecom-sudparis.eu

ASN.1 predefined types

- INTEGER == INTEGER
- BOOLEAN == BOOLEAN
- REAL == REAL
- CHARSTRING == IA5String

SDL

ASN.1

NEWTYPE colors **LITERALS**
red, blue,black, yellow, white**;**
ENDNEWTYPE colors**;**

colors ::= **ENUMERATED**
{red, blue,black, yellow, white};

SYNONYM clearcolor colors
= white;

clearcolor colors ::= white;

Stephane.Maag@telecom-sudparis.eu

Composite types in ASN.1: Sequence types (Structure in SDL)

```
NEWTYPE T_Seq
STRUCT
        a BOOLEAN;
        b CHARSTRING OPTIONAL;
        c INTEGER DEFAULT 5;
ENDNEWTYPE T_Seq;
```

```
T_Seq ::= SEQUENCE {
        a BOOLEAN,
        b IA5String OPTIONAL,
        c INTEGER DEFAULT 5};
```

assignment

```
s1 T_Seq ::= {  a TRUE,
                b 'OK',
                c 12};
s2 T_Seq ::= { a FALSE,
                c 23};
s3 T_Seq ::= {a TRUE};
```

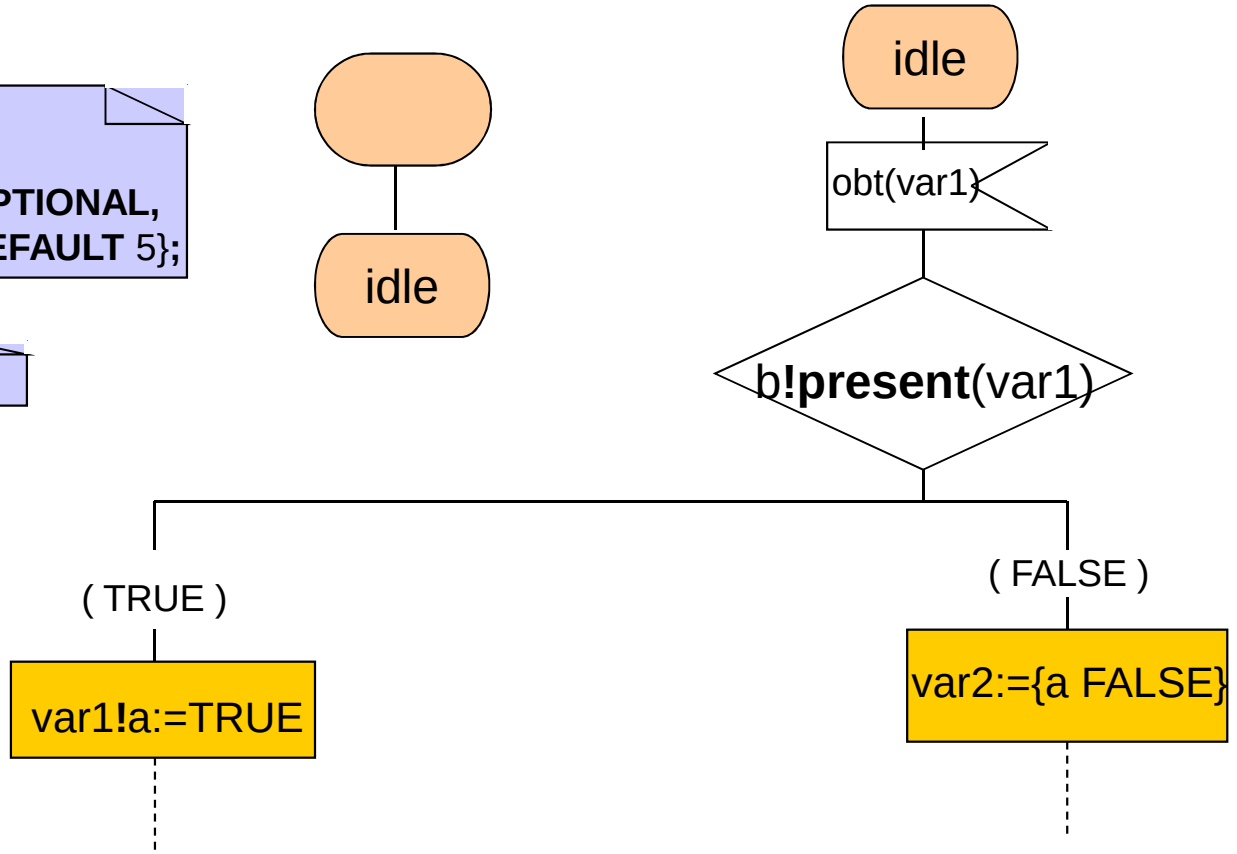
Stephane.Maag@telecom-sudparis.eu

Sequence Example

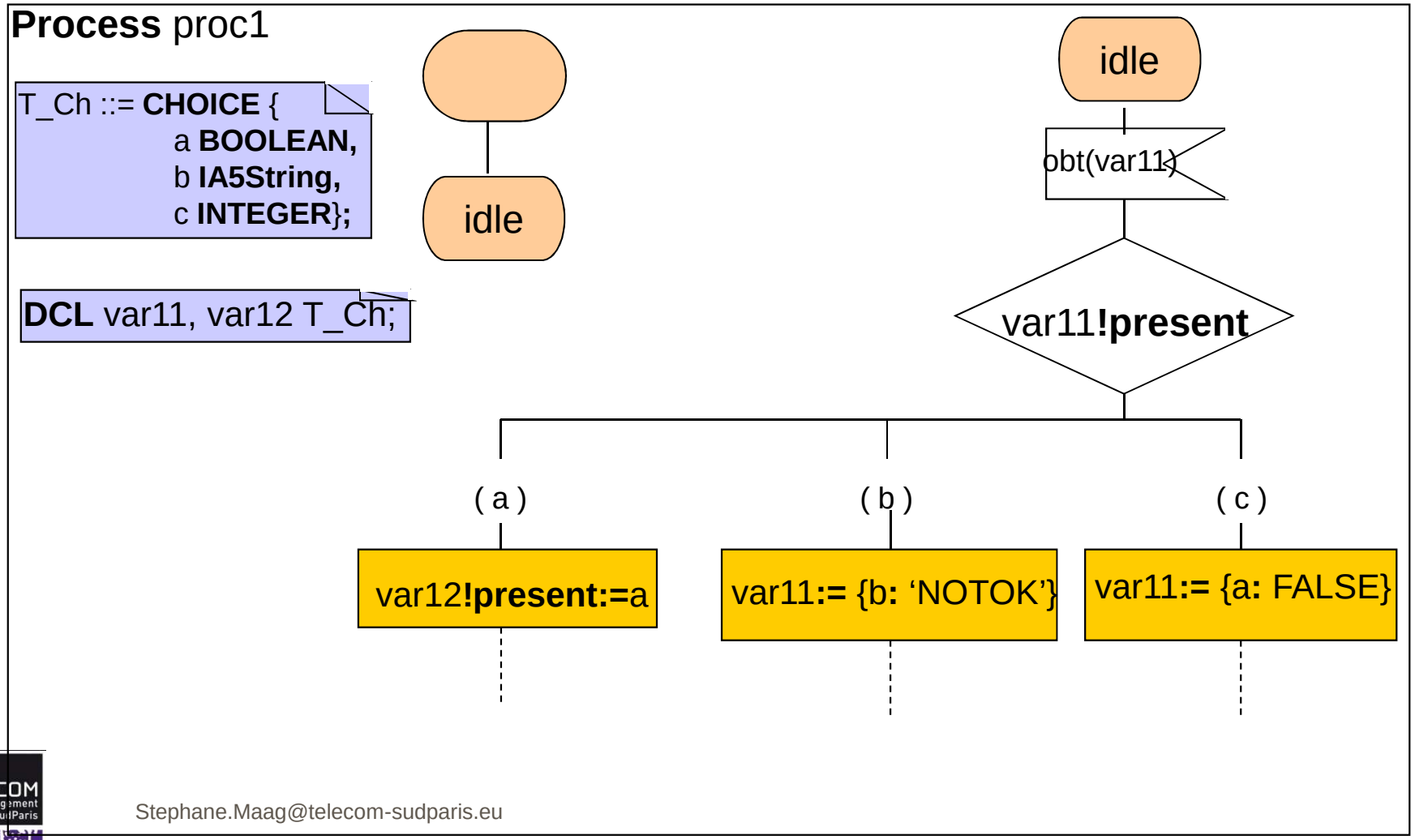
Process proc1

```
T_Seq ::= SEQUENCE {
            a BOOLEAN,
            b IA5String OPTIONAL,
            c INTEGER DEFAULT 5};

DCL var1, var2 T_Seq;
```



Composite types in ASN.1: CHOICE types



CONCLUSION

- Think to use structural types for reusability
- Process ID
- Readability with procedure and macrodefinitions
- CHOICE in ASN.1

Stephane.Maag@telecom-sudparis.eu

Exercises

Specify a process that receives a message *ATM_req* containing a data *atm_req* as a structure "*(quantity, ticket)"* where *quantity* is an integer and *ticket* is an optional character.

This process sends the output *OK* if *ticket* is received or *NOK* if not.

Stephane.Maag@telecom-sudparis.eu