PRAGMADEV
modeling and testing tools

Automatic Verification of BPMN Models

Mihal Brumbulli - PragmaDev

Emmanuel Gaudin - PragmaDev

Ciprian Teodorov - ENSTA Bretagne

Introduction

- BPMN describes processes and interactions between different participants in a complex organization
  - Air traffic control
  - Satellite constellations
  - Army forces coordination
- BPMN is part of NAF, which is used by the French Army to describe interactions between actors in a mission
- VeriMoB is a research project financed by the DGA (Direction Générale de l'Armement) in collaboration with Eurocontrol and Airbus DS who provided some real use cases

Motivation

- Make sure BPMN models are correct
  - Syntax
  - Semantic
  - Logic
- Automatically verify a given scenario
  - Record scenarios
  - Replay scenarios
- Automatically verify possible scenarios
  - Model complexity
  - Property verification

Outcome

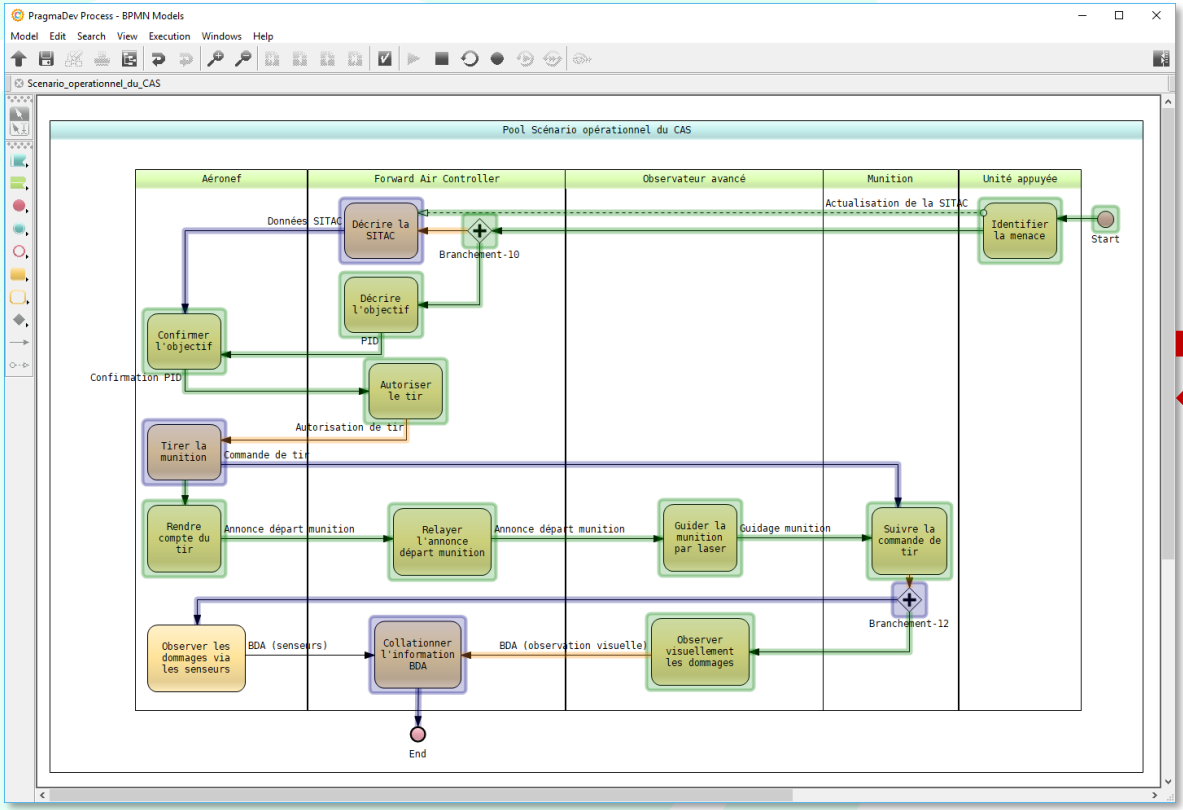
- BPMN
  - Viewer
  - Executor
  - Explorer
- Use cases
  - DGA
  - Eurocontrol
  - Airbus DS

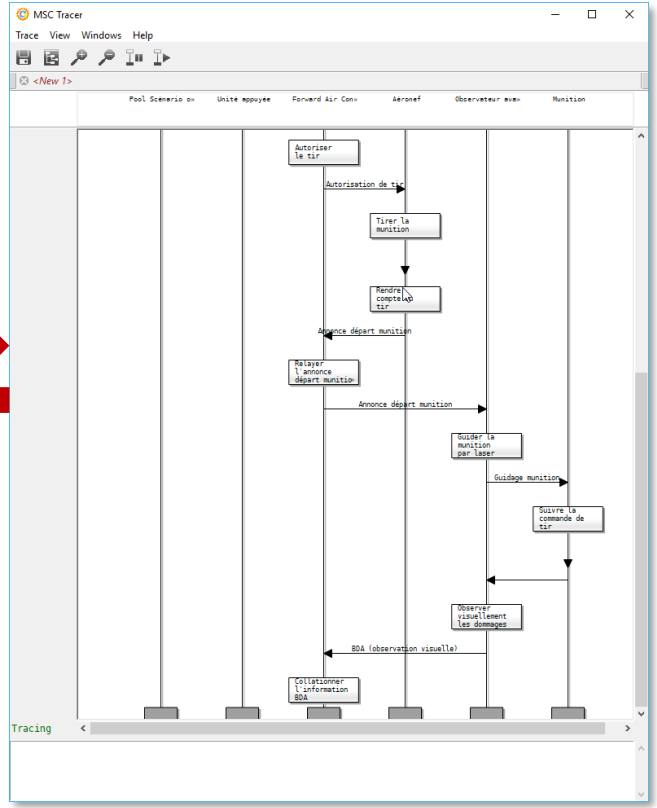
BPMN model executor

- A BPMN model element may accept actions
  - **Enable**
  - **Disable**
- Actions can be issued on sequence flows, message flows, and process call-activities
- A BPMN model element has a list of states (most recent state is shown in color)
  - **None**
  - **Active**
  - **Ready**
  - **Enabled**
  - **Disabled**

PRAGMADEV
modeling and testing tools

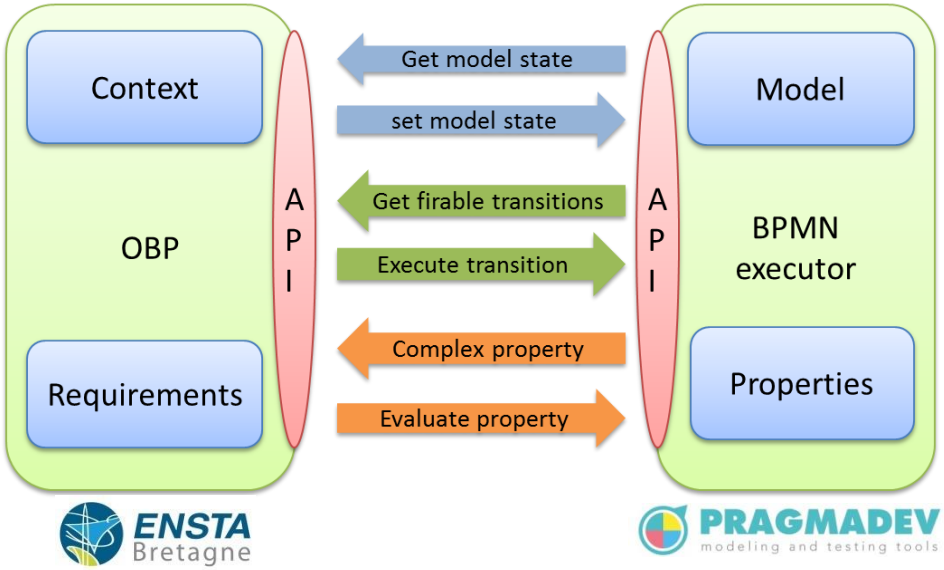
Interactive execution & replay





Verification tool architecture

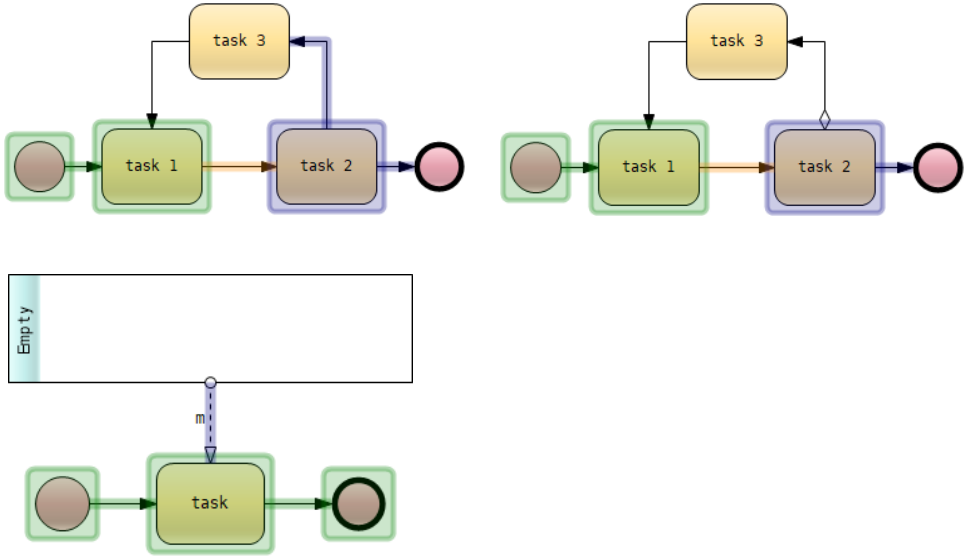
- Model Executor
  - Loads and executes the model
  - Evaluates properties
- Observer Based Prover
  - Language agnostic
    - Relies on external execution engine
    - Knows nothing about the model
  - Temporal logic verification
  - Integration of debugging & model-checking



Model exploration issues

The current execution state of a BPMN model is composed of the execution states of all elements in the model

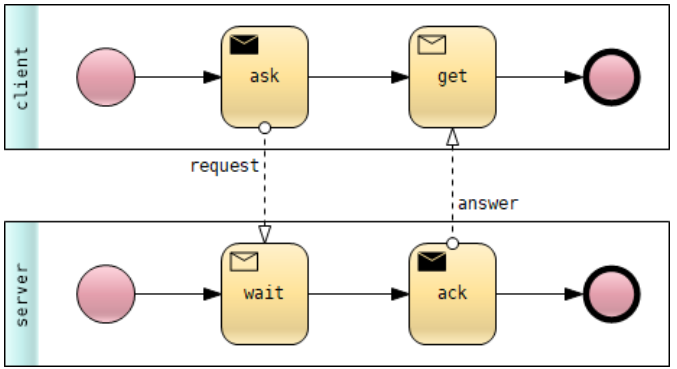
- Issue 1: Loops
  - Looping indefinitely is possible
  - Limit to one iteration
- Issue 2: Black-box pools
  - Infinite sends
  - Limit sends with no receives

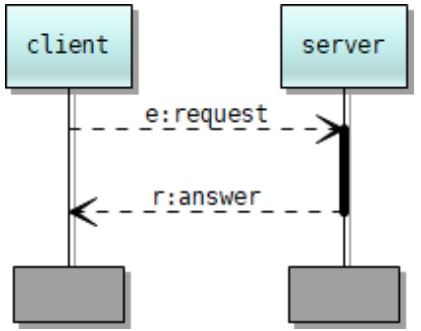


Property Sequence Chart

A property is a relation on exchanged messages with or without constraints

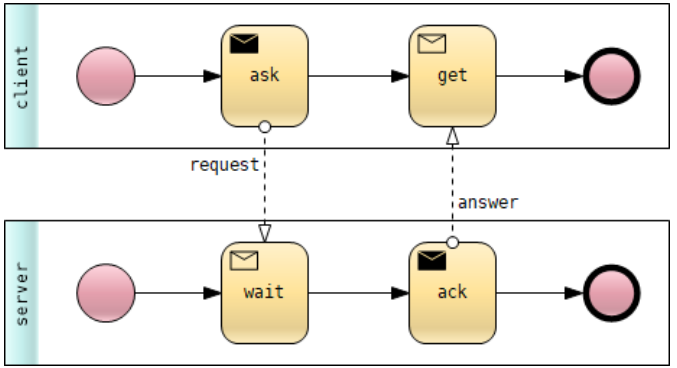
- Simple and expressive formalism similar to MSC
- Can describe both positive and negative scenarios
- Three kinds of messages:
  - Regular
  - Required
  - Fail



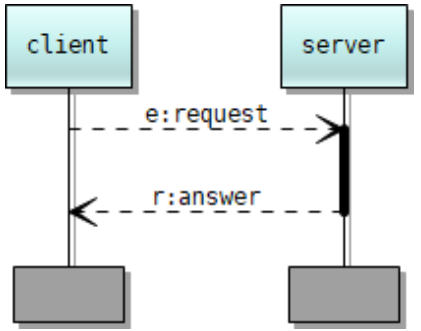


Generic Property Specification Language

- Properties can be defined also in GPSL
- GPSL is the language used by OBP
  - Structural layer
  - Atomic Propositions
  - Propositional Layer
  - Büchi Automaton Layer
  - LTL Layer

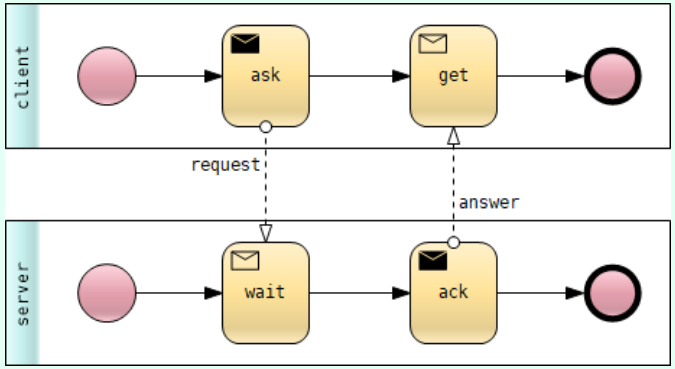


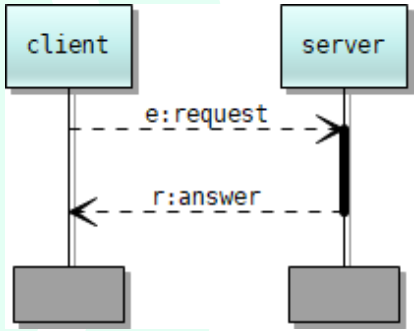

```
A = |+:E:/0:SEM_SYMB_21|
R = |+:E:/0:SEM_SYMB_20|

main_property = [] (R -> <> A)
```



Translating the properties

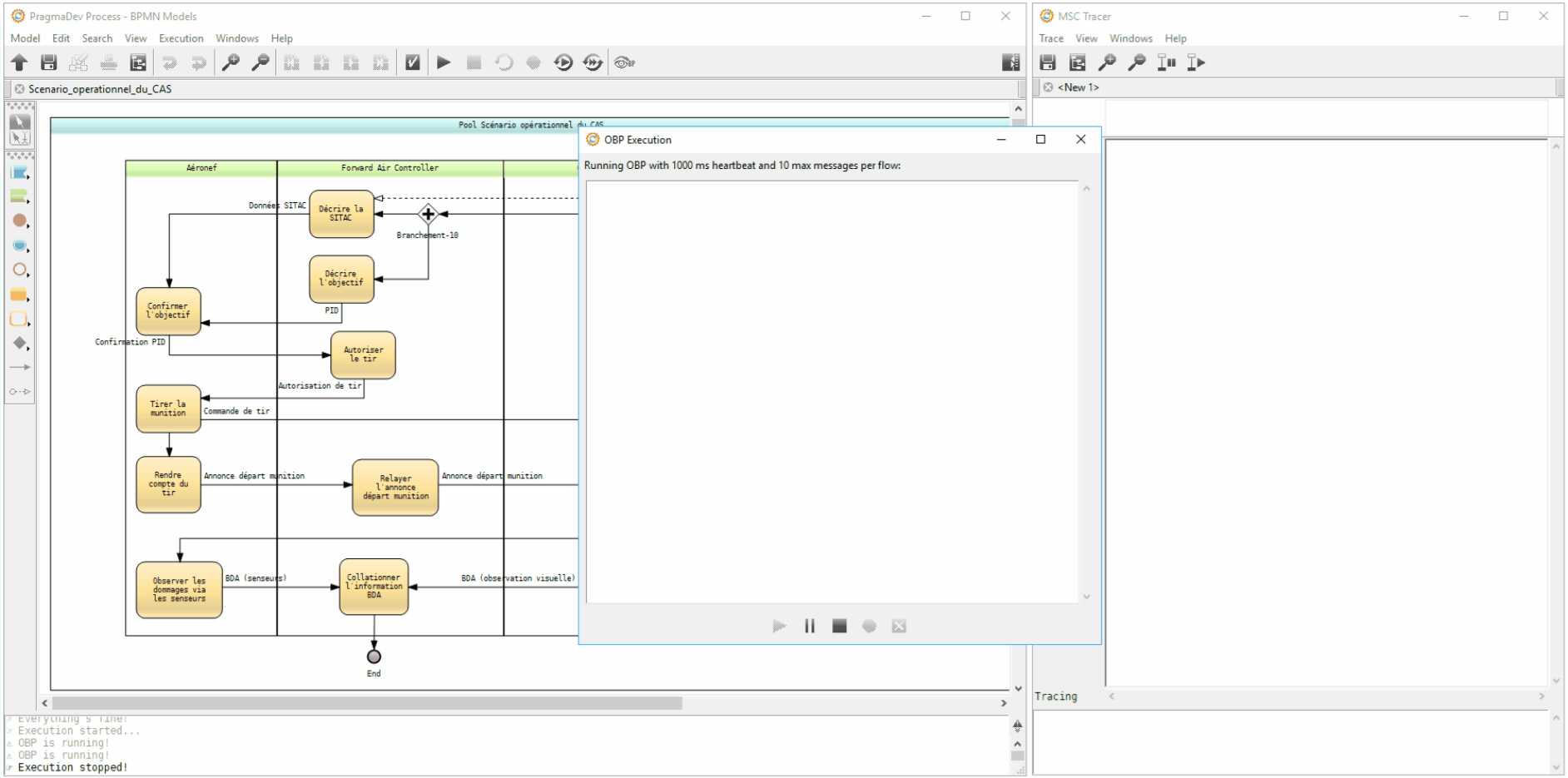
- PSC to Büchi automata (University of l'Aquila)
- Adapted the transformation to generate Büchi automata conforming with the GPSL Büchi automaton layer





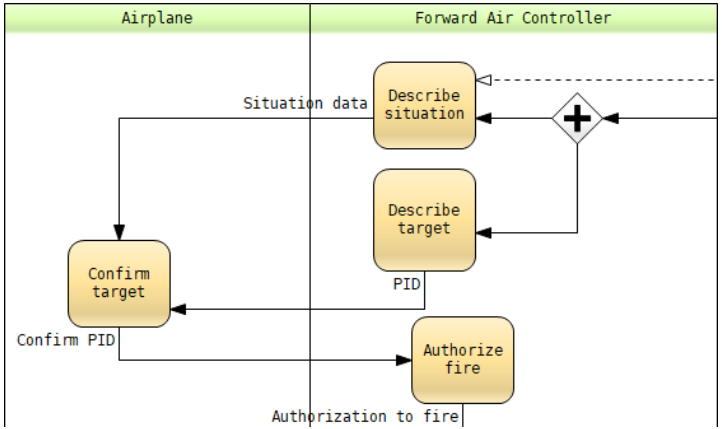

```
main_property =
  let
    server_answer_client = |+:E:/0:SEM_SYMB_21|,
    client_request_server = |+:E:/0:SEM_SYMB_20|
  in
    states S3, S1, S4;
    initial S1;
    accept S3;
    S1 [not client_request_server] S1;
    S1 [client_request_server] S3;
    S3 [not server_answer_client] S3;
    S3 [server_answer_client] S4;
    S4 [not client_request_server] S1;
    S4 [client_request_server] S3
```

PRAGMADEV
modeling and testing tools



Example

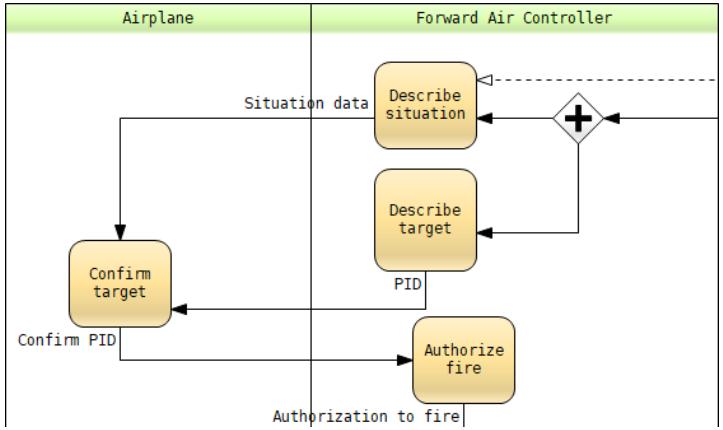
- CAS - Close Air Support model
- State-Space exploration with OBP provides information on the complexity of the model

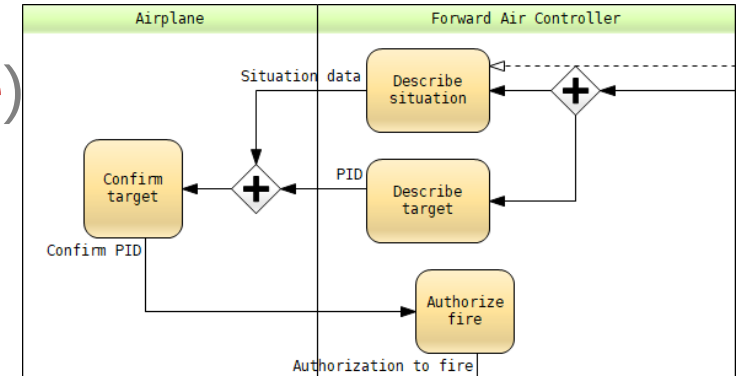


If the number of possible execution paths is very large compared to the model complexity, it probably means that there is a mis-construct in the model

Example

- Mixing of different gateways for fork and merge
- Multiple unintended flows create complexity
- 38 configurations after fixing the merging gateways (9000 otherwise)

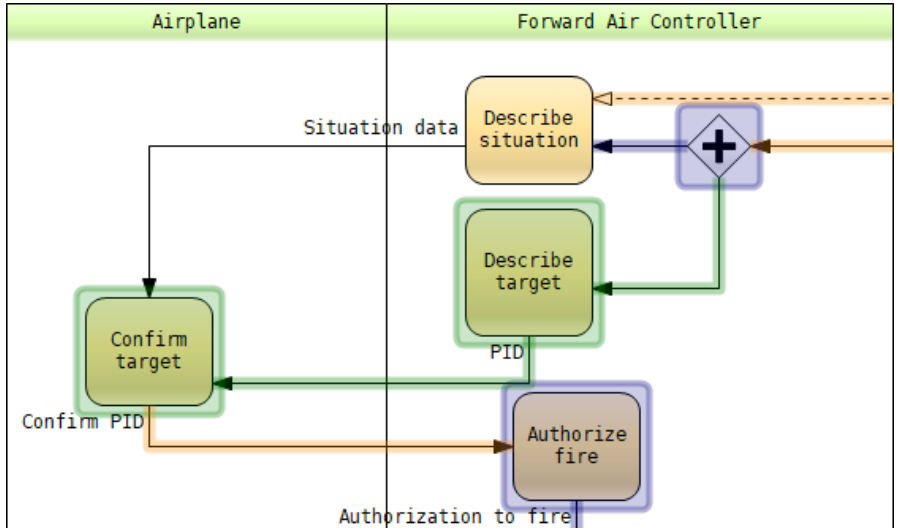




Example

- Mis-construct better identified with a property

```
// S -> "Describe situation" was enabled
S = |=:E:CAS:MegaId-331F39A752DD5FDC|

// T -> "Describe target" was enabled
T = |=:E:CAS:MegaId-331F39BE52DD603B|

// P -> "Authorize fire" becomes active
P = |+:A:CAS:MegaId-331F3B2A52DD67E0|

// Property: P if both S and T
main_property = <>P -> (!P U S) && (!P U T)
```



Conclusion

- VeriMoB project was aiming at verifying BPMN models
- A set of tools have been developed including a viewer, an executor, and an explorer
- BPMN models can be verified statically and dynamically, and properties can be checked while exploring the model
- The tools made possible identifying problems in all use cases involved in the project