



# Proof-Based System Engineering with AADL

Fabrice Hesling  
Amélie Schyn  
Romain Sezestre  
**Jean-François Tilman**

Axlog Ingénierie

# Overview

- Part I: Goals and principles
- Part II: Representation of PBSE information with AADL
- Part III: An example – Meta scheduling
  - Informal presentation
  - AADL description
- Conclusion & perspectives

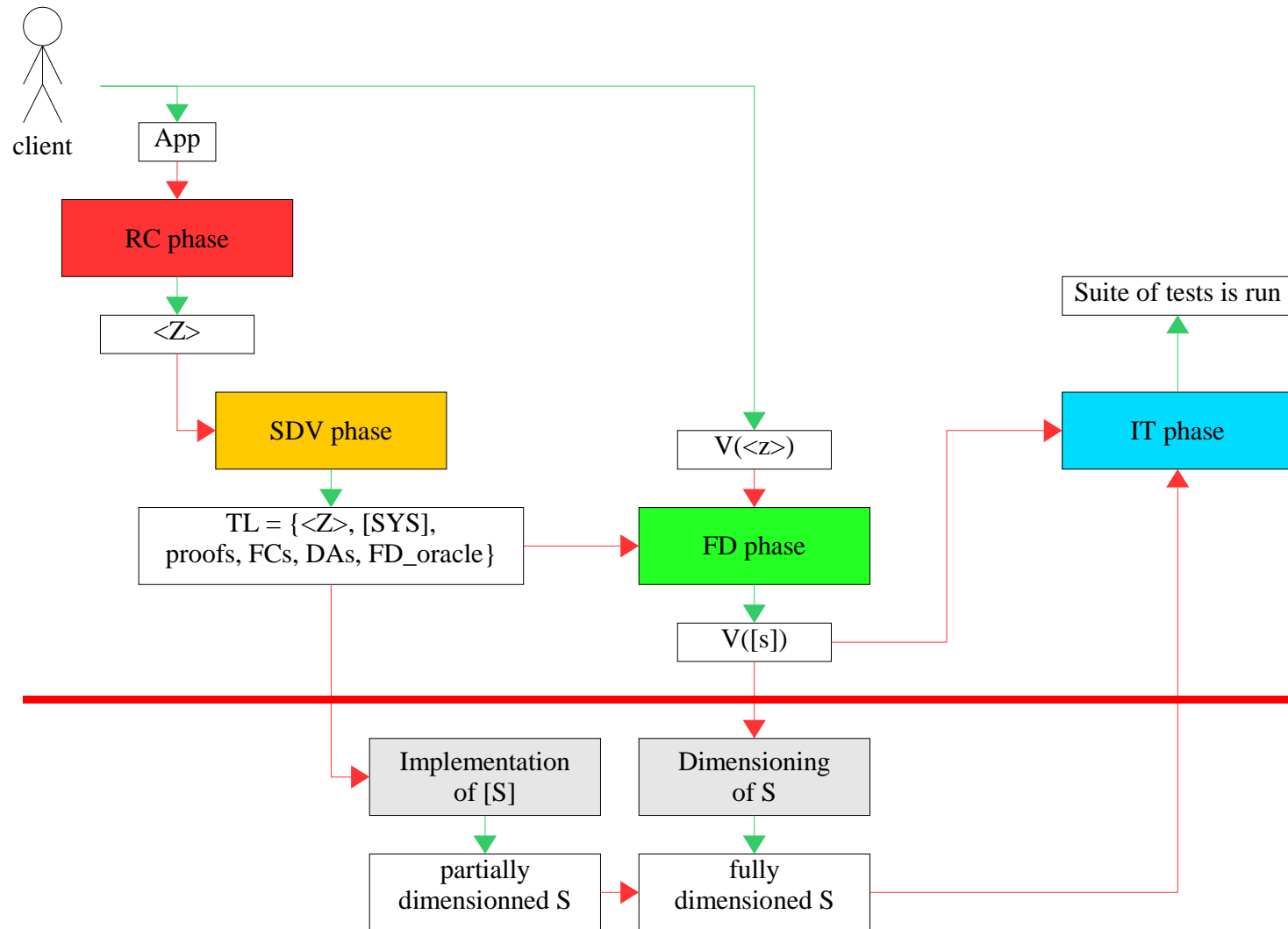
# Part I

## Goals and principles

# PBSE principles

- PBSE = *proof-based system engineering*
- Goal: development of embedded systems, or building blocks, by applying proofs and formal methods
- Several PBSE phases:
  - Requirement capture (RC): capture of the non-functional requirements (noted  $\langle Z \rangle$ )
  - System design & validation (SDV): design systems/BBs meeting the requirements ([S])
  - Feasibility & dimensioning (FD): dimensioning of the system/BBs and ensuring their feasibility

# PBSE process



# Use of AADL for PBSE

- PBSE needs
  - Formal representation of the information handled by the PBSE process (the *technical leaflets*)
  - Use of this representation by dedicated tools
- Solution
  - Use of AADL to represent the system architecture
  - Use of extensions to support PBSE specificities

# Part II

## Representation of the elements of a *technical leaflet*

# Problem <Z>: Models<sub>PBSE</sub>

- Models “adversaries” of the system
- Can be sorted in six categories
  - **Computational:** computational & communication delays
  - **Resources & data:** resource management problems
  - **Process:** structure, execution conditions, behaviour & data modification patterns
  - **Event:** periodic, sporadic or aperiodic...
  - **Failure:** erroneous behaviour of a component
  - **Failure occurrence:** occurrence models associated to failure

# AADL modelling

- Most of the models<sub>PBSE</sub> can be modelled using new AADL properties
- Example:

```
Computation_Model : enumeration ( Pure_Synchronous,  
Fixed_Delay_Pure_Synchronous,  
Partially_Synchronous,  
Eventual_Synchronous,  
Incomplete_Synchronous,  
Eventual_Incomplete_Synchronous,  
Asynchronous,  
Time_Free_Partially_Synchronous,  
Pure_Asynchronous_Augmented_Failure_Detectors,  
Pure_Asynchronous )  
  
applies to (all);
```

- Exception: failure & failure occurrence models are describe using error model annex.

# Problem $\langle Z \rangle$ : properties<sub>PBSE</sub>

- Give information about system behaviour
- Can be sorted in four categories
  - **Safety:** “Nothing bad happens”
  - **Liveness:** “Something good eventually happens”
  - **Timeliness:** “Something good eventually happens in finite, bounded predictable time”
  - **Dependability:** Concerned with system behaviour in the presence of partial failures

# AADL modelling

- 1) Use of AADL string properties

```
PBSE_Project::Timeliness => ( "promptness", "punctuality" );
```

- 2) Use of an annex to give the semantics

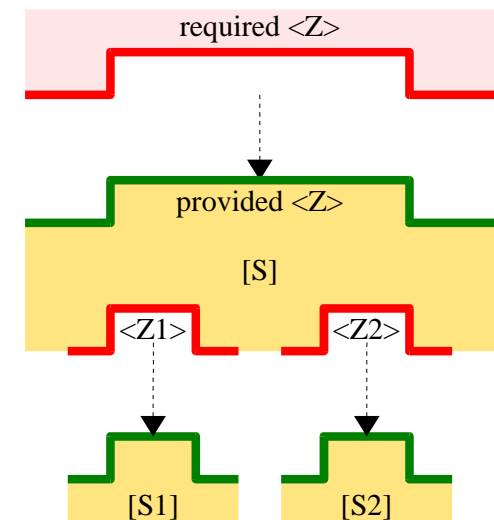
```
Annex Timeliness {**  
    promptness => "there is a maximal delay between the  
                  occurrence of an event and its reaction";  
    punctuality => "there is an exact delay between two events";  
**}
```

- 3) Use of a temporal logic approach for automation  
(e.g., TCTL)

```
Annex Timeliness {**  
    promptness => "AG [send(m) => AF<5 receive(rm)]";  
    punctuality => "EG [send(m) => AF=11 receive(rm)]";  
**}
```

# Solution [S] – composed BB

- PBSE building block  $\equiv$  AADL component
- Specification of a solution [S]  
 $\equiv$  description of a component type and implementation
- Composition of building blocks  
 $\equiv$  subcomponents into a component implementation
- The AADL component description contains the other parts of the associated technical leaflet



# Solution [S] – terminal BB

- A terminal building block can be an algorithm
- => Use of new AADL properties to contain the description of algorithms
- Many possible representations: informal description, pseudo code, programming code, formal language...

```
property set PBSE_Properties is  
  Algorithm : aadlstring applies to ( all ) ;  
  Algorithm_Files : list of aadlstring applies to ( all ) ;  
  Algorithm_Language : PBSE_Project::Supported_Algorithm_Languages  
    applies to ( all ) ;  
end PBSE_Properties ;  
  
property set PBSE_Project is  
  Supported_Algorithm_Languages : type enumeration (text, matlab) ;  
end PBSE_Project ;
```

# Design assumptions

- Design assumption: prerequisite to use a building block
- Description provided by AADL strings to represent constraints on the properties of used BBs

```
property set PBSE_Properties is
  Design_Assumptions : list of aadlstring
    applies to ( all ) ;
end PBSE_Properties ;
```

- Objective: verify design assumptions by tools

```
PBSE_Properties::Design_Assumptions => (
  "PBSE_Project::Computation_Model => Pure_Synchronous"
);
```

# Other TL elements

- **Proofs:** ensure that the solution solves the problem
  - For human reader only
- **Feasibility conditions:** necessary and/or sufficient
  - Formulas which may be handled by tools  
=> possible use of a mathematical language
- **FD\_Oracle:** says whether a given dimensioning of the system is possible
- AADL representations
  - Description provided by AADL strings
  - Possible semantics => automatic operations

# Part III

## An example: meta scheduling

# Meta scheduling

- **Problem**
  - Scheduling processes belonging to different inter-criticality classes (criticality classes A, B and C)
  - Overloads shall be accounted
  - In presence of overload
    - No class A process shall miss its deadline
    - A class B process may miss its deadline if no class C process meet its deadline
- **A proposed solution: EDF+ algorithm**
  - => design of a reusable building block

# Building block EDF+

```
process EDFp

properties
  PBSE_Project::Event_Model_arrival => (Periodic, Sporadic, Aperiodic);
  PBSE_Project::Timeliness => ("Property_A", "Property_B");
  PBSE_Properties::Algorithm_Files => "EDFplus_algorithm.txt" ;
  PBSE_Properties::Design_Assumptions => (
    "PBSE_Project::Computation_Model => Pure_Synchronous
      applies to (PBSE_project::Requires_Thread)",
    "PBSE_Project::Process_Criticality_Class => (A, B, C)
      applies to (PBSE_project::Requires_Thread)" ) ;

annex Timeliness {**
  Property_A => "[class(p,A)] => [AG meet-deadline(p)]
    applies to (PBSE_project::Requires_Thread) ";
  Property_B => "[class(p,B) ^ !overload] => [AG meet-deadline(p)]
    ^ [class(p,B) ^ overload]
    => [EG !meet-deadline(p)]
    U [(class(q,C)) ^ meet-deadline(q)]
    applies to (PBSE_project::Requires_Thread) ";
**}

end EDFp ;
```

# Building block composition

```
system Using_Meta_Scheduler
properties
  -- PBSE models
  PBSE_Project::Process_Model_criticality => B ;
  PBSE_Project::Process_Model_execution => Non_Interruptible ;
end Plane ;
```

```
system Using_Meta_Scheduler.impl
subcomponents
  process1: process ;
  process2: process ;
  process3: process ;
  p: processor ;
  scheduler: process EDFp {
    PBSE_project::Requires_Thread => (reference process1,
                                       reference process2,
                                       reference process3) ;
    PBSE_project::Requires_Processor => reference p ;
  };
end Using_Meta_Scheduler.impl;
```

# Part IV

## Conclusion and perspectives

# Conclusion

- *Proof-based system engineering* has to be considered for future practices in system development activities
- Thanks to its flexibility, AADL is able to support such innovative approaches
  - Our works take advantage of the current AADL developments
- PBSE + AADL appears at a promising solution to solve the “software and system crisis”
- A set of specific PBSE tools is currently under development to support this approach