VnV with Classical B method

Marielle Petit-Doche

Systerel

December 5th, 2013

Systerel

Contenu

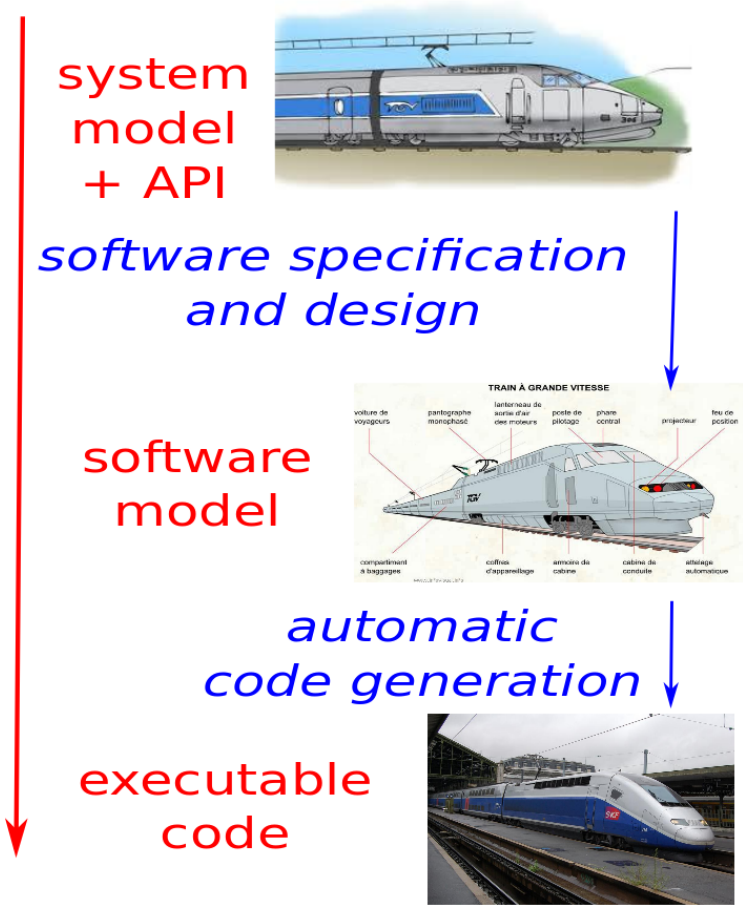
Contenu

Classical B

Classical B

Software Design :

- full functional description
- some safety constraints
- fully formal model
- refinement
- deterministic model to generate code (translation)



Highlights

- Mature approach
- *The B Book, Assigning Programs to Meanings* [Abrial1996]
- More than 20 years of practice in railway industry : ALSTOM, SIEMENS, AREVA,...
- Lots of application in urban railway domain : CBTC, ZC, CC, PMI,..
- Industrial tool : Atelier B (now free but partly closed source)
- Correct by construction approach
- Automatic code translation (C, Ada,...)
- Well adapted to develop critical software according to EN50128

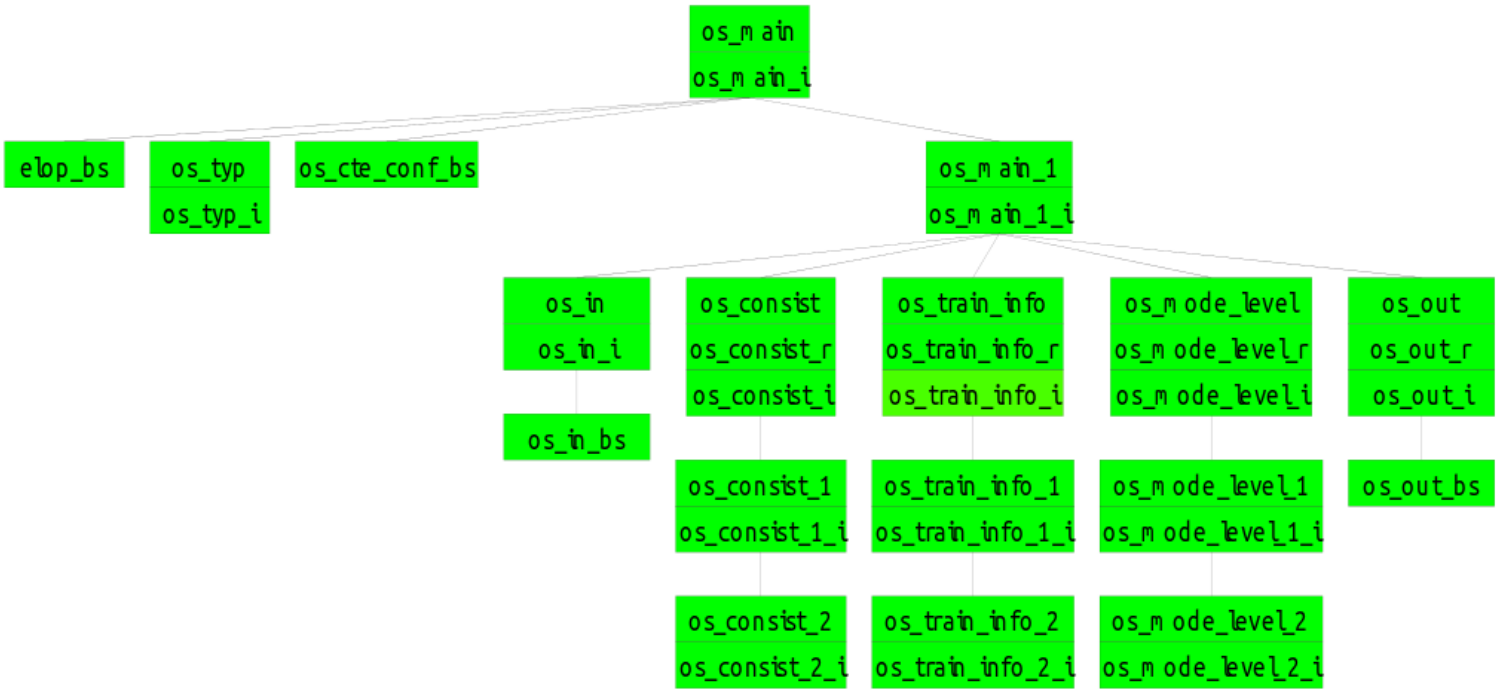
Structured models and Formal language

- Modules to describe the software architecture
  - Machine
  - Refinement
  - Implementation
- Links between modules : Sees, Import, Promotes,...
- Operations, Variables, Initialisation, Invariants,...
- Language : first order logic + set theory (inherited from Z notation, Hoare logic,...)

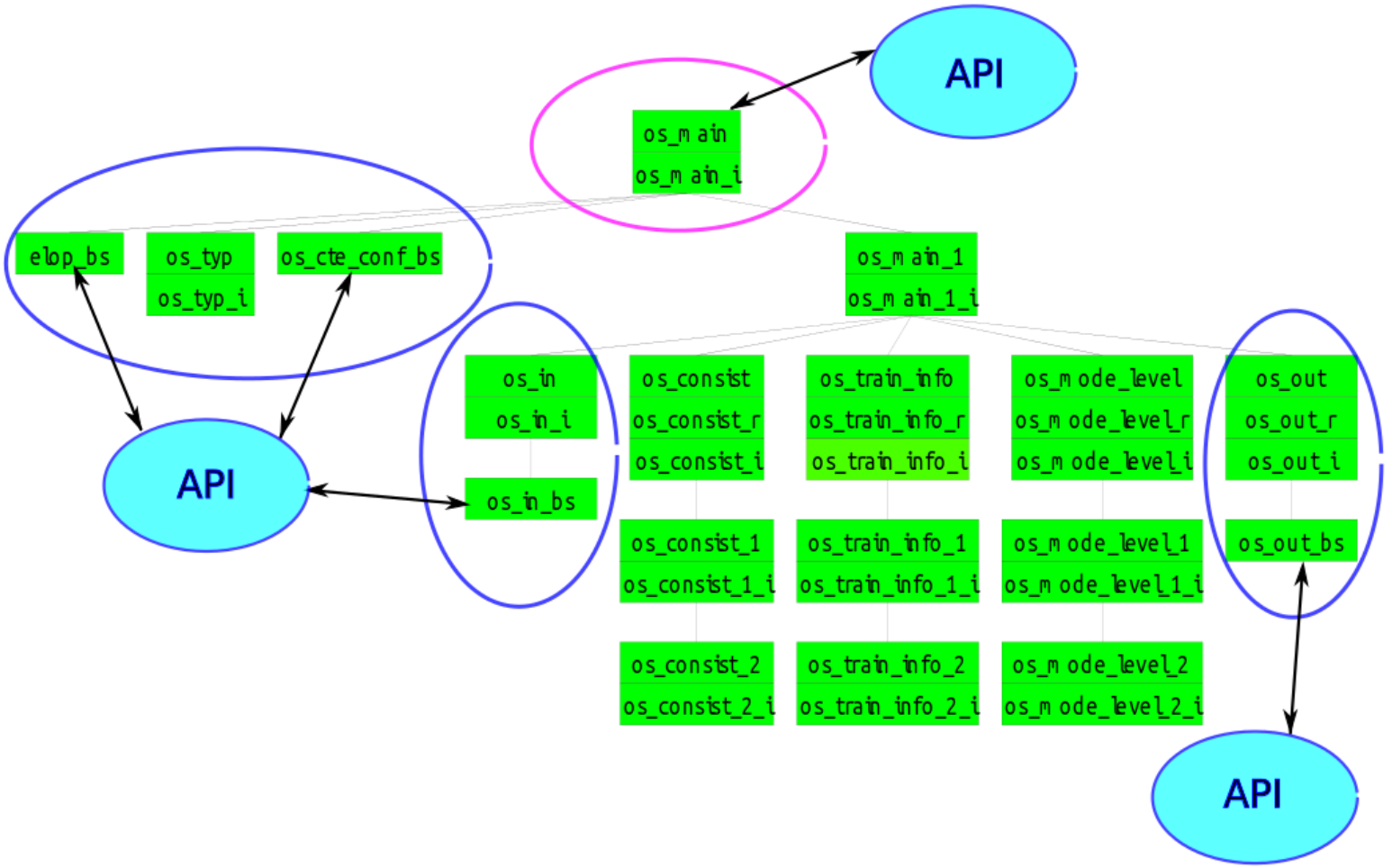
Systerel

Contenu

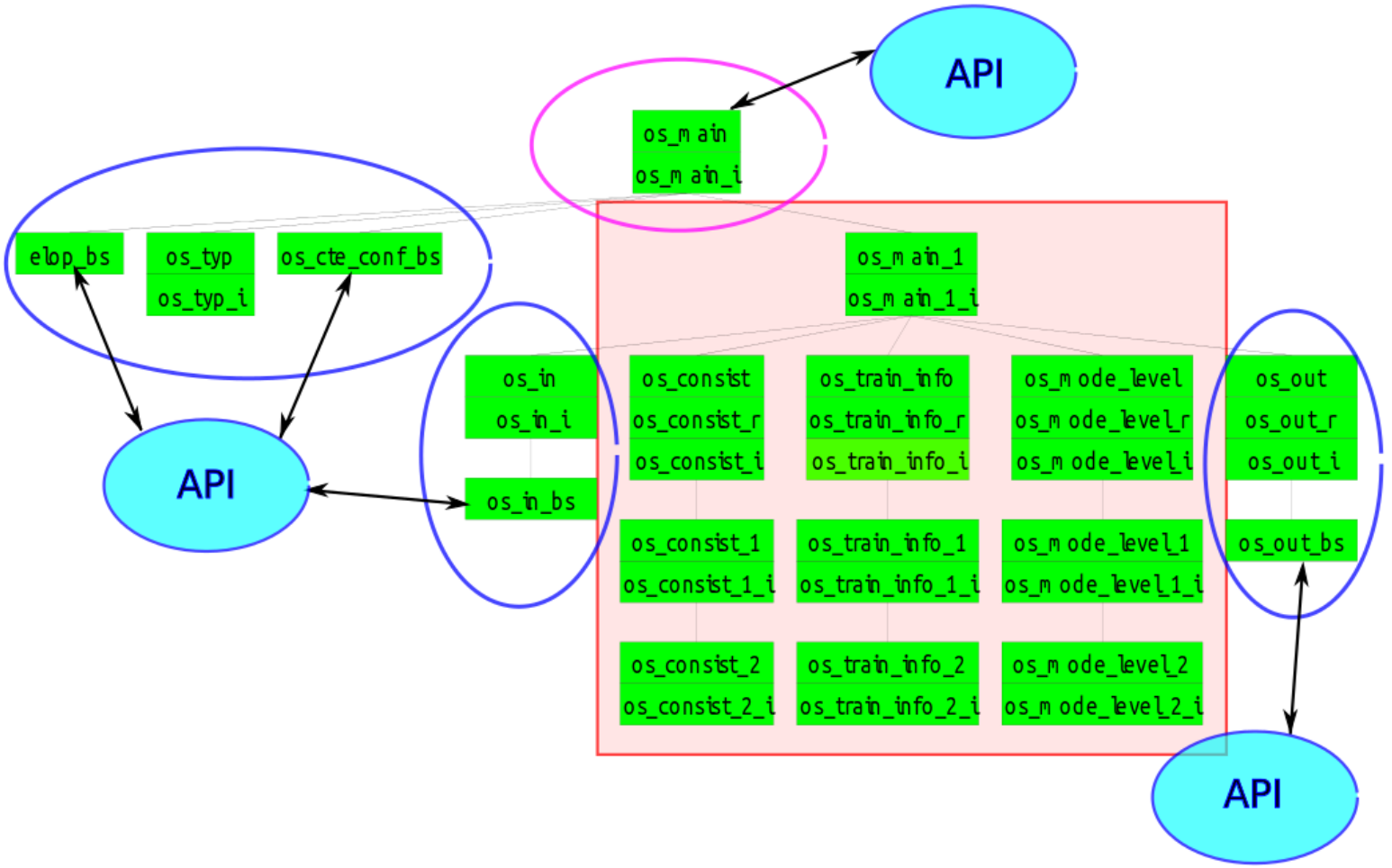
Example : Procedure On sight (1)



Example : Procedure On sight (2)



Example : Procedure On sight (3)



Example : Procedure On sight (4)

MACHINE
os_mode_level
ABSTRACT_VARIABLES
ack_request_os_mode
INVARIANT
ack_request_os_mode $\in$ **BOOL**
INITIALISATION
ack_request_os_mode $:\in$ **BOOL**
OPERATION
send_ack_request_driver =
BEGIN
ack_request_os_mode :(
ack_request_os_mode $\in$ **BOOL**)
END

Example : Procedure On sight (5)

REFINEMENT
os_mode_level_r
REFINES
os_mode_level
ABSTRACT_VARIABLES
ack_request_os_mode
INITIALISATION
ack_request_os_mode := **FALSE**
OPERATIONS
send_ack_request_driver =
IF
waiting_for_ack = **TRUE** $\wedge$
waiting_for_ack_prec = **FALSE**
THEN
ack_request_os_mode := **TRUE**
ELSE
ack_request_os_mode := **FALSE**
END

Example : Procedure On sight (6)

IMPLEMENTATION
os_mode_level_i
REFINES
os_mode_level_r
IMPORTS
os_mode_level_1
PROMOTES
read_waiting_for_ack , *read_waiting_for_ack_prec* , *write_ack_request_os_mode*
OPERATIONS
send_ack_request_driver =
VAR *l_waiting_for_ack* , *l_waiting_for_ack_prec*
IN
l_waiting_for_ack ← **read_waiting_for_ack** ;
l_waiting_for_ack_prec ← **read_waiting_for_ack_prec** ;
IF *l_waiting_for_ack* = **TRUE** ∧ *l_waiting_for_ack_prec* = **FALSE**
THEN
write_ack_request_os_mode(**TRUE**)
ELSE
write_ack_request_os_mode(**FALSE**)
END
END

Demonstration

Contenu

Translation principles

- Direct translation to C or Ada
- No code optimisation
- Same modular structure :
  - Machine : *.mch $\Rightarrow$ *.h
  - Implementation : *.imp $\Rightarrow$ *.c
- Links between modules :
  - Sees : *#include*
  - Import : *#include*
  - Promotes : *#define*
- Operations, Variables : same names
- Invariants are not translated

Demonstration

Contenu

Static and Type-checking

Aim : to check if the model is well written

- Static verification
- Verification of typing properties
- Data Typing defined as predicate
- Syntactical and lexical verification
- Type control of each expression
- Verification of the architecture of the model : no loop, no unlinked component,..

B0 checking

Aim : to check the B model can be automatically translated to C or Ada

- Verify the model is deterministic
- Verify the model shall be implemented

Well-definedness

Aim : to check if the model can be interpreted

- Verify interpretation of an expression
- Verify value of an expression is unique
- Examples :
  - domain definition of a function
  - division by zero

Model checking

Aim : to check some behaviours of the model

- Static semantic check
- Simulation/ Animation
- Finite sets, states and transitions limitations
- Tool ProB

Constraint-Based checking

Aim : to prove some properties on the model

- On going work
- symbolic and automatic proof of some properties
- SAT solvers
- BWare project : plugin SAT solveur with AtelierB

Formal Proof

Aim : to check the consistency of the model and prove some properties

- Powerful formal approach to check consistency of a model
- Automatic generation of proof obligation according well known rules :
  - Well-definition
  - Invariant preservation
  - Refinement verification
- Symbolic and partly automatic proof
- Easy maintenance
- Allow to suppress verification on generated code (unit and integration test)

Contenu

Questions ?