Atelier B has turned 20 – ABZ 2016 Linz

Presentation · May 2016

DOI: 10.13140/RG.2.2.33148.16000

CITATIONS
4

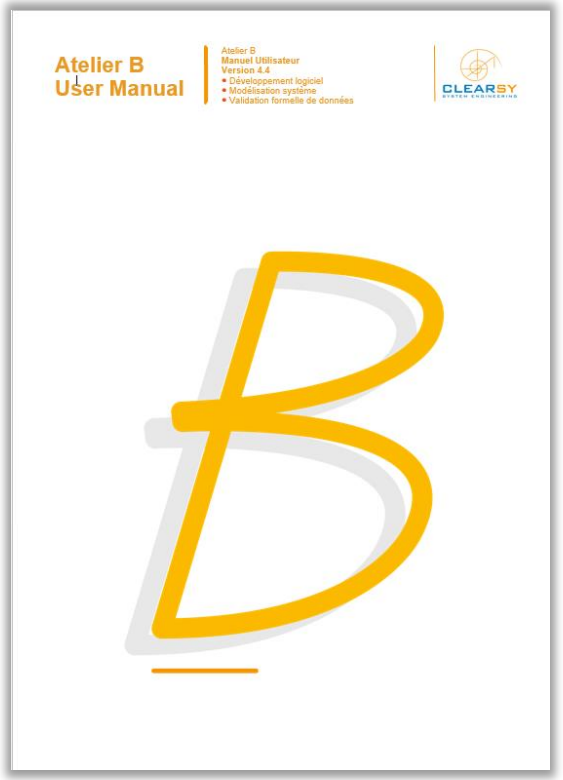
READS
26

1 author:

Thierry Lecomte
ClearSy System Engineering
54 PUBLICATIONS **418** CITATIONS

SEE PROFILE

Atelier B has turned twenty



Thierry Lecomte
ABZ 2016
Linz

CLEARSY
SYSTEM ENGINEERING

Agenda

What happened during the last 20 years ?

Focus on some aspects:

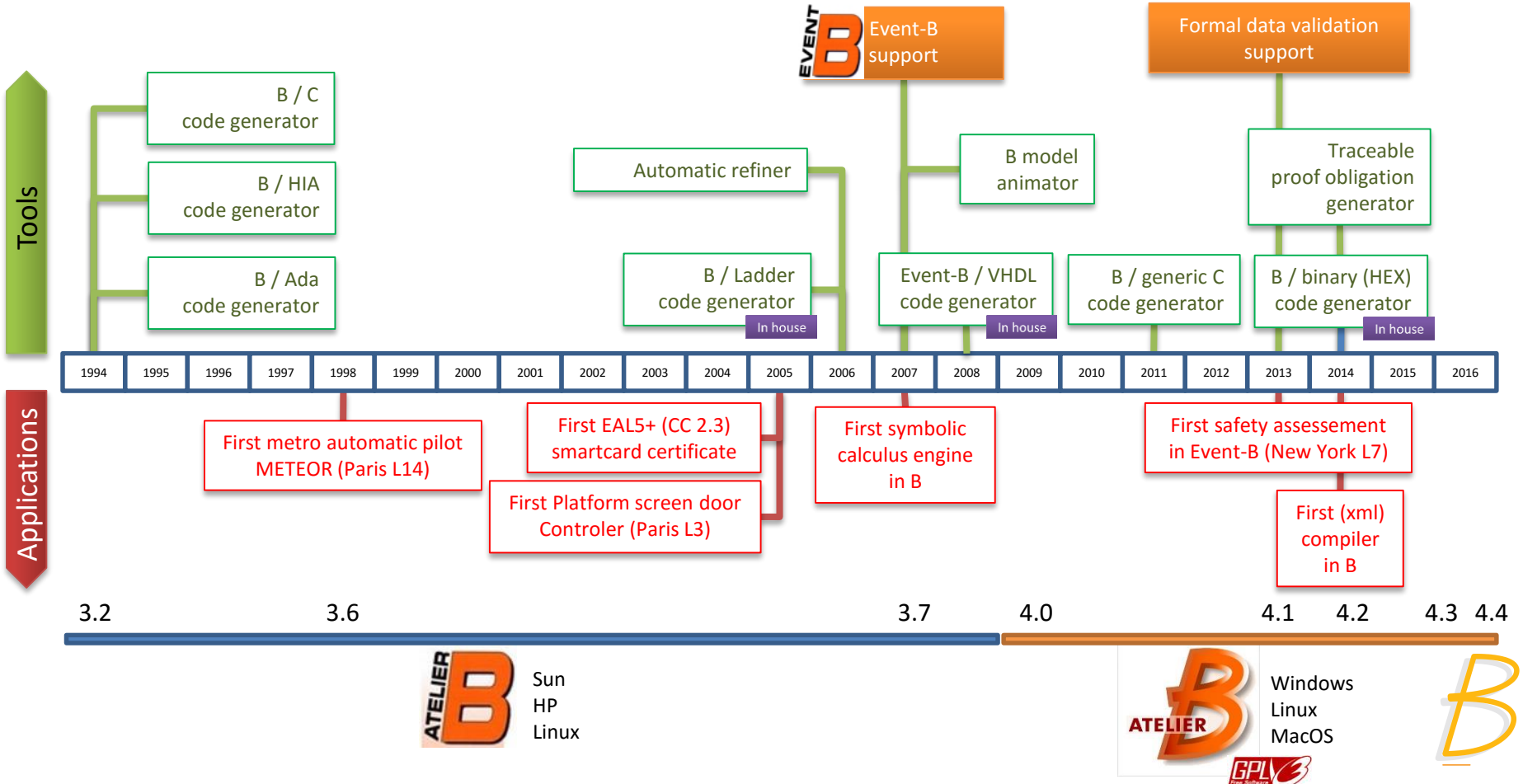
- Modelling
- Proving
- Generating code

Not a raw description of the tool

More how its usage has evolved

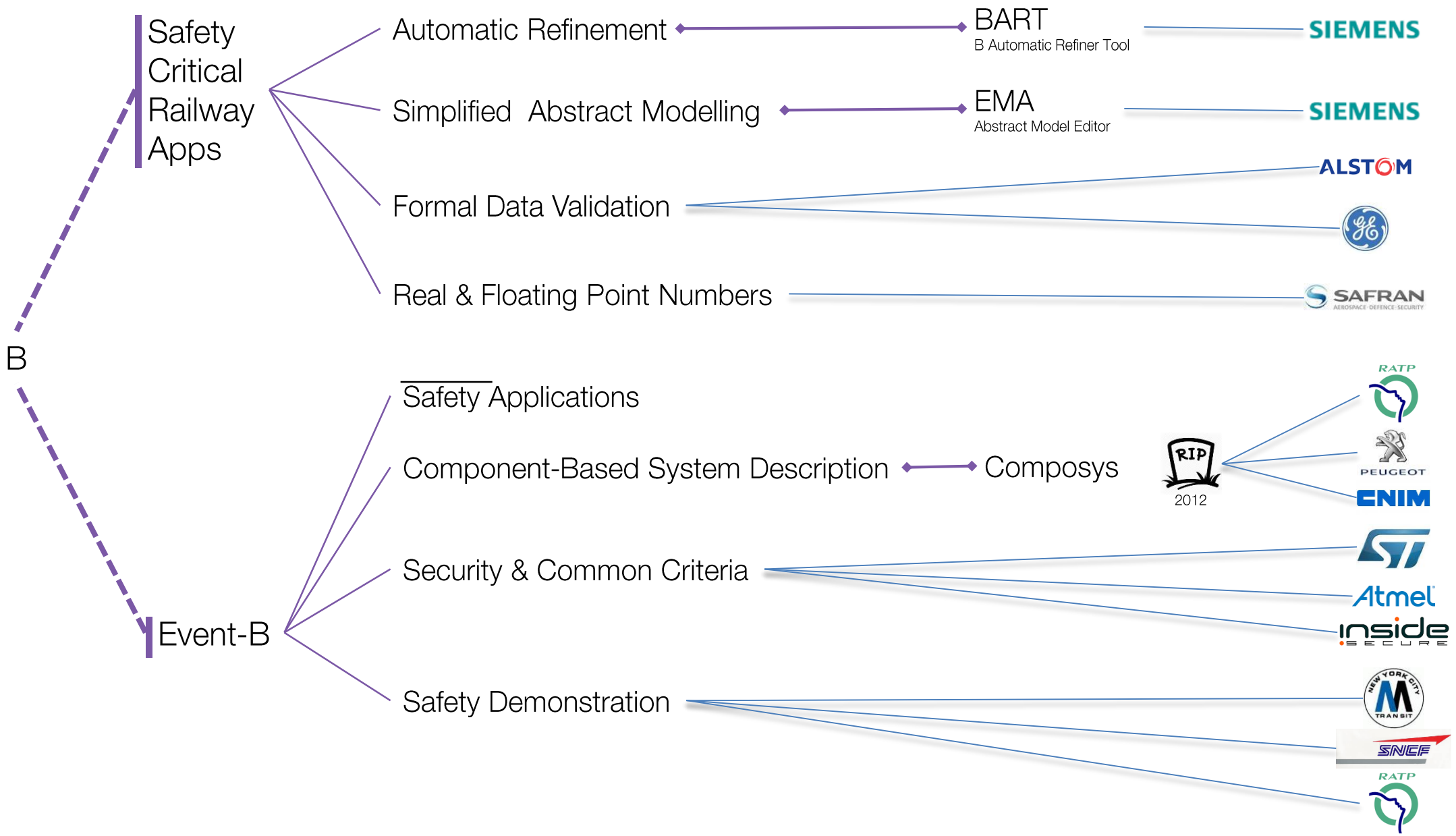
Driven by projects

- Atelier B initially developed by Fernando Mejia
  - Development funded for Driverless metro in Paris (L14)
    - First customer: University of Sherbrooke
      - METEOR released
        - Creation of ClearSy (full ownership)
          - Atelier B Community + Maintenance



Modelling

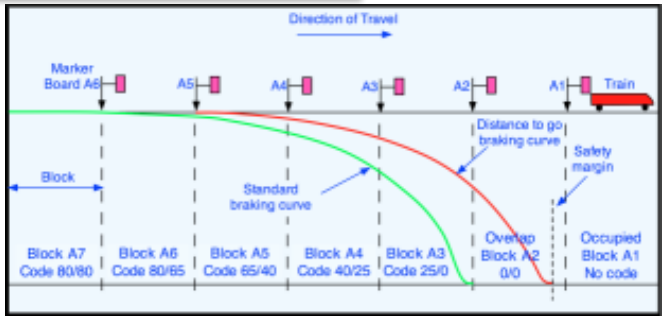
Modelling Random Walk



Safety Critical Railway Applications

≡ Reduce intervals between trains (from 120s to 90s / 75s)

- Passive security not sufficient (power off)
- Active security is required (trains have to brake when emergency)



Braking curves

≡ B instead of program proof for

- Embedded software (Automatic Train Pilot)
- Localization: graph-based algorithms
- Energy control: integer arithmetic (braking curve)
- Emergency braking: Boolean predicates
- Trackside software (Interlocking)

≡ +30% automatic metros in the world

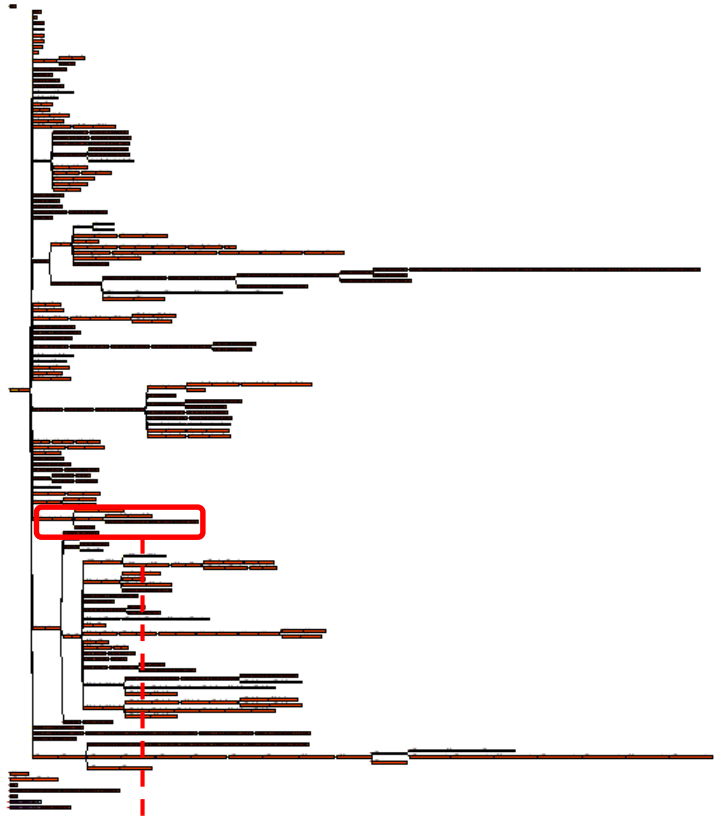
B initial success is due to urban women fertility

Confirm / Deny ?

Safety Critical Railway Applications

Top level implementation

- Imports 55 components
- Specify top level one-cycle function:
  - Compute location, manage kinetic energy, control PSD, trigger emergency braking, etc.



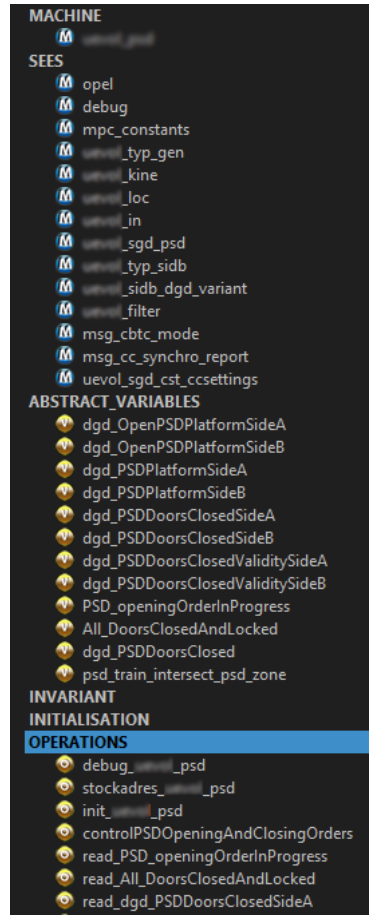
Modern Automatic Train Protection Software
(2015)

Metrics

- 233 machines, 50 kloc
- 46 refinements, 6 kloc
- 213 implementations, 45 kloc
- 3 000 definitions
- 23 000 proof obligations (83 % automatic proof)
- 3 000 added user rules (85 ?? % automatic proof)

Platform Screen Doors

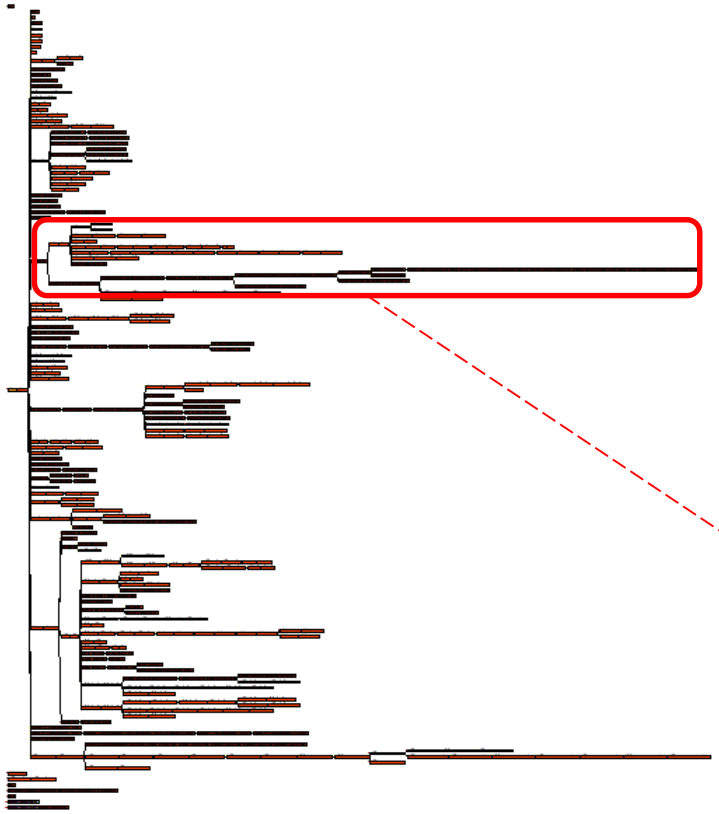
- Top component: 12 variables 14 operations
- 10 components, 2 kloc specification, 2kloc implementation
- Connection with I/O through basic machines



Safety Critical Railway Applications

Biggest function: Localization

where is the train ?



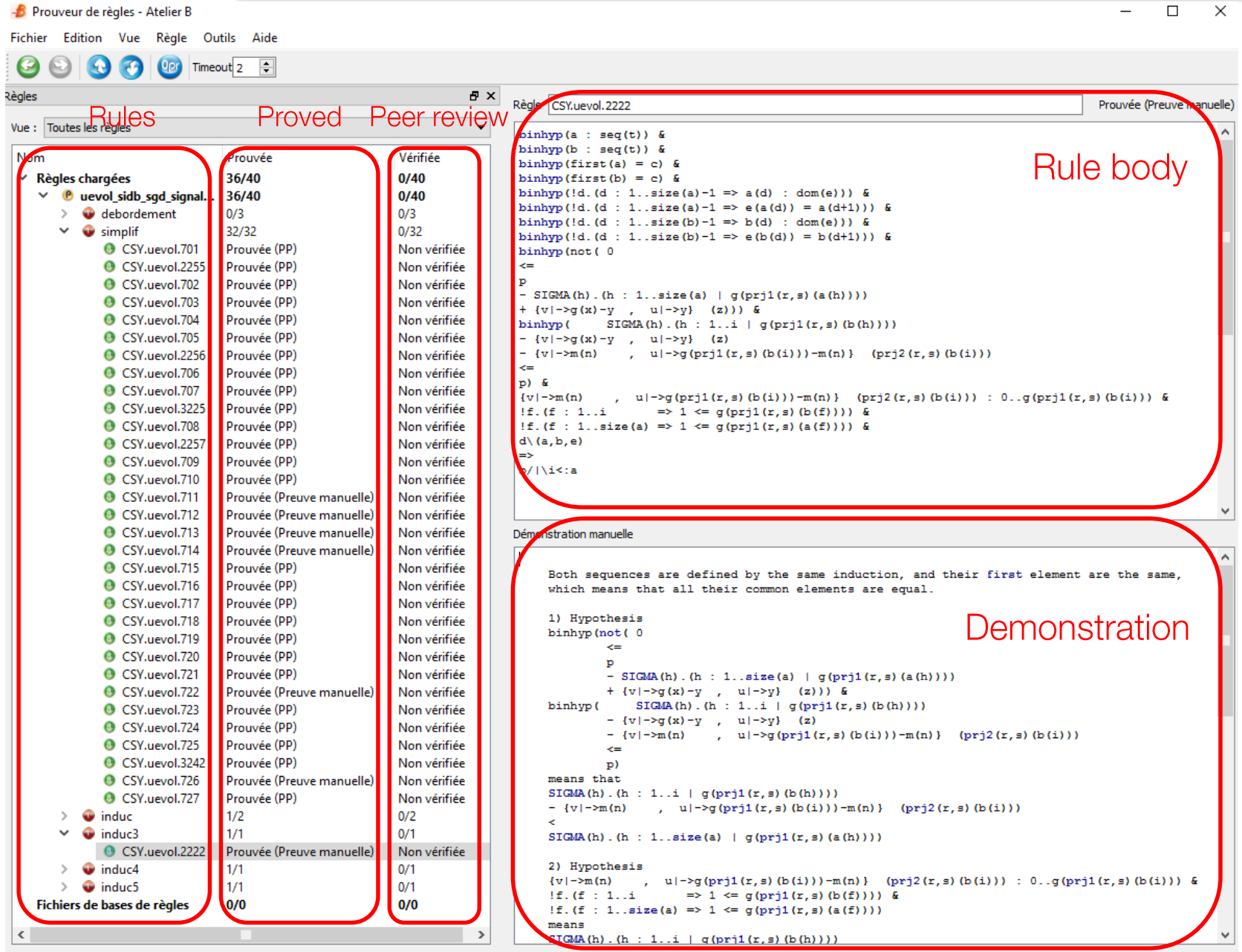
Modern Automatic Train Protection Software
(2015)

Post-condition of one operation:

variables become such as …

```
variables :(types &
    properties &
    properties_train &

    (loc_trainLocated = TRUE =>
        0 <= loc_locationUncertainty
        & kine_kineInvalid = FALSE
        & loc_train_track /= {}
        & first(loc_train_track) = loc_ext2Block |-> oppositeDirection(loc_ext2Dir)
        & !ii.(ii : 1..size(loc_train_track)-1 => loc_train_track(ii) : dom(sidb_nextBlock))
        & !ii.(ii : 1..size(loc_train_track)-1 => sidb_nextBlock (loc_train_track(ii)) = loc_
        & #aa.(aa : 1..size(loc_train_track) & prj1(t_block,t_direction)(loc_train_track(aa))


        & loc_rearBlock = { c_cabin1 |-> loc_ext2Block, c_cabin2 |-> loc_ext1Block, c_none |-
        & (loc_rearBlock = c_block_init
            =>loc_rearDir=c_up)
        & ( not (loc_rearBlock = c_block_init)
            =>  loc_rearDir    = { c_cabin1 |-> oppositeDirection(loc_ext2Dir), c_cabin2 |-> c
        & loc_rearAbs    = { c_cabin1 |-> loc_ext2Abs, c_cabin2 |-> loc_ext1Abs, c_none |-> 0
        & loc_frontBlock = { c_cabin1 |-> loc_ext1Block, c_cabin2 |-> loc_ext2Block, c_none |
        & loc_frontDir    = { c_cabin1 |-> loc_ext1Dir, c_cabin2 |-> loc_ext2Dir, c_none |-> c
        & loc_frontAbs    = { c_cabin1 |-> loc_ext1Abs, c_cabin2 |-> loc_ext2Abs, c_none |-> s
    ))
```

Safety Critical Railway Applications



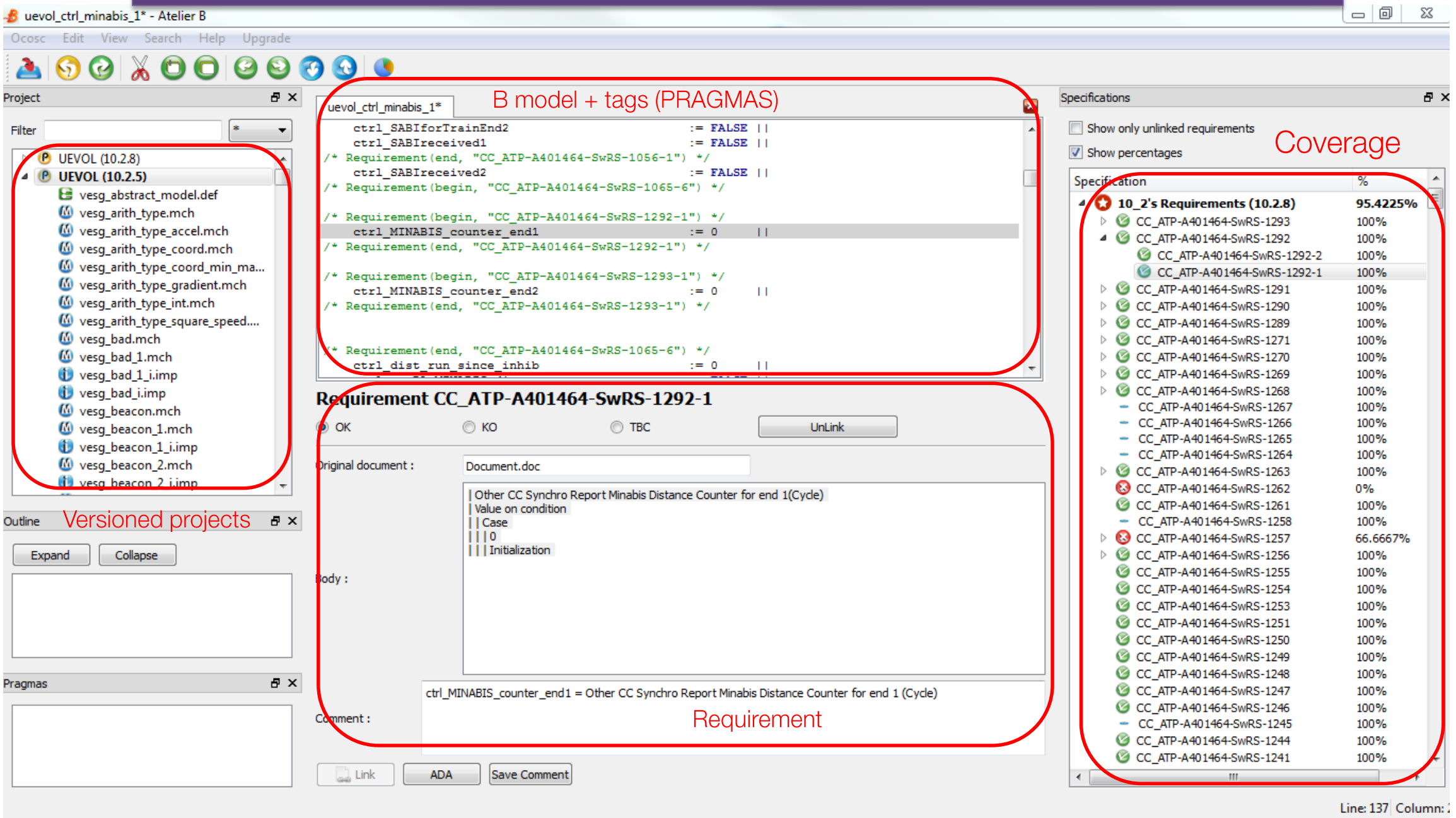
User added rules to bootstrap the main prover:

- THEORY language (ako PROLOG)
- Pattern-matching
- Backward, forward, rewrite rules
- Added at project level or component level
- Partly reused from project
- Have to be validated to confirm that a project is 100% proven

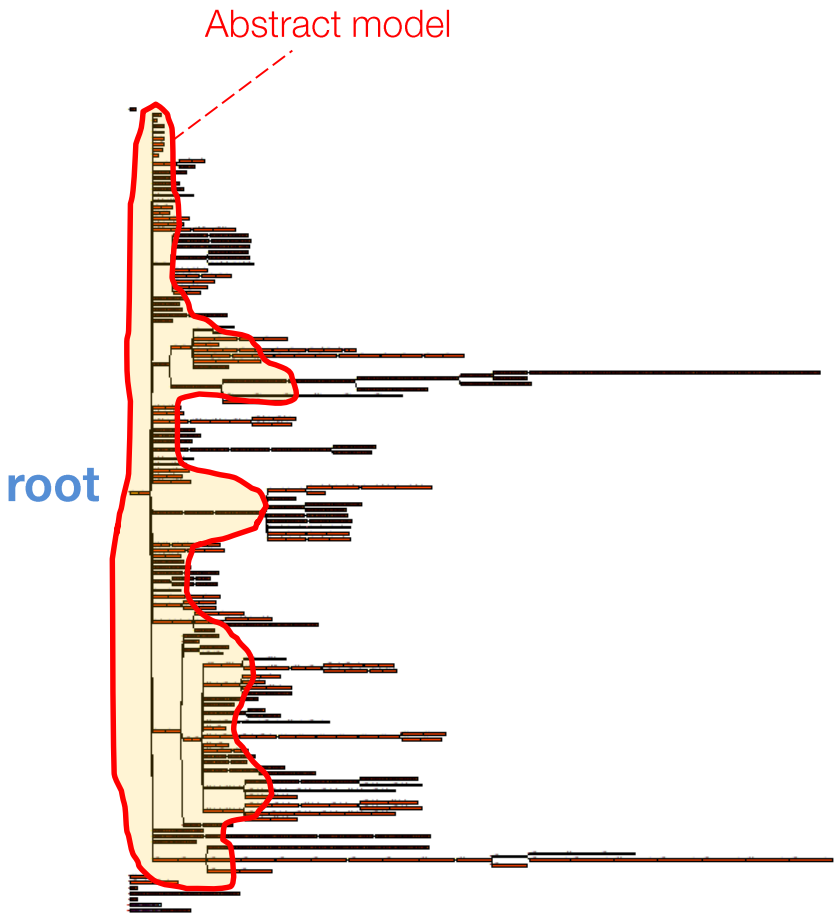
Specific tool & GUI to validate user added rules (maintenance version)

- Translation to predicates
- Predicate prover
- Peer review
- Report

Safety Critical Railway Applications



Automatic Refinement

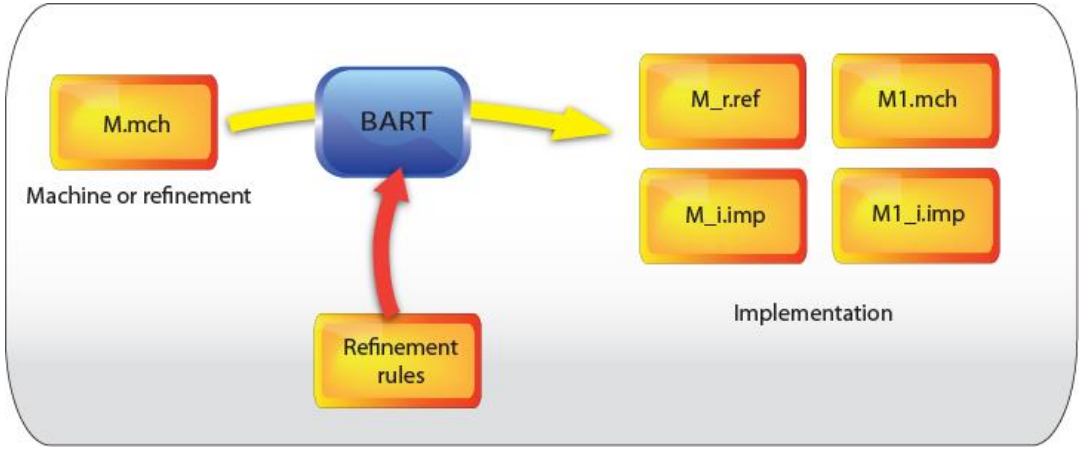


Modern Automatic Train Protection Software
(2015)

≡ Progressive transformation of a model containing all design decisions into a model able to be translated into an imperative language

≡ Initially invented by Matra Transport

- To support their own development methodology (abstract model first)
- In-house tools (EdithB + Bertille) to obtain an efficient process
- Refinement engine based on refinement rules



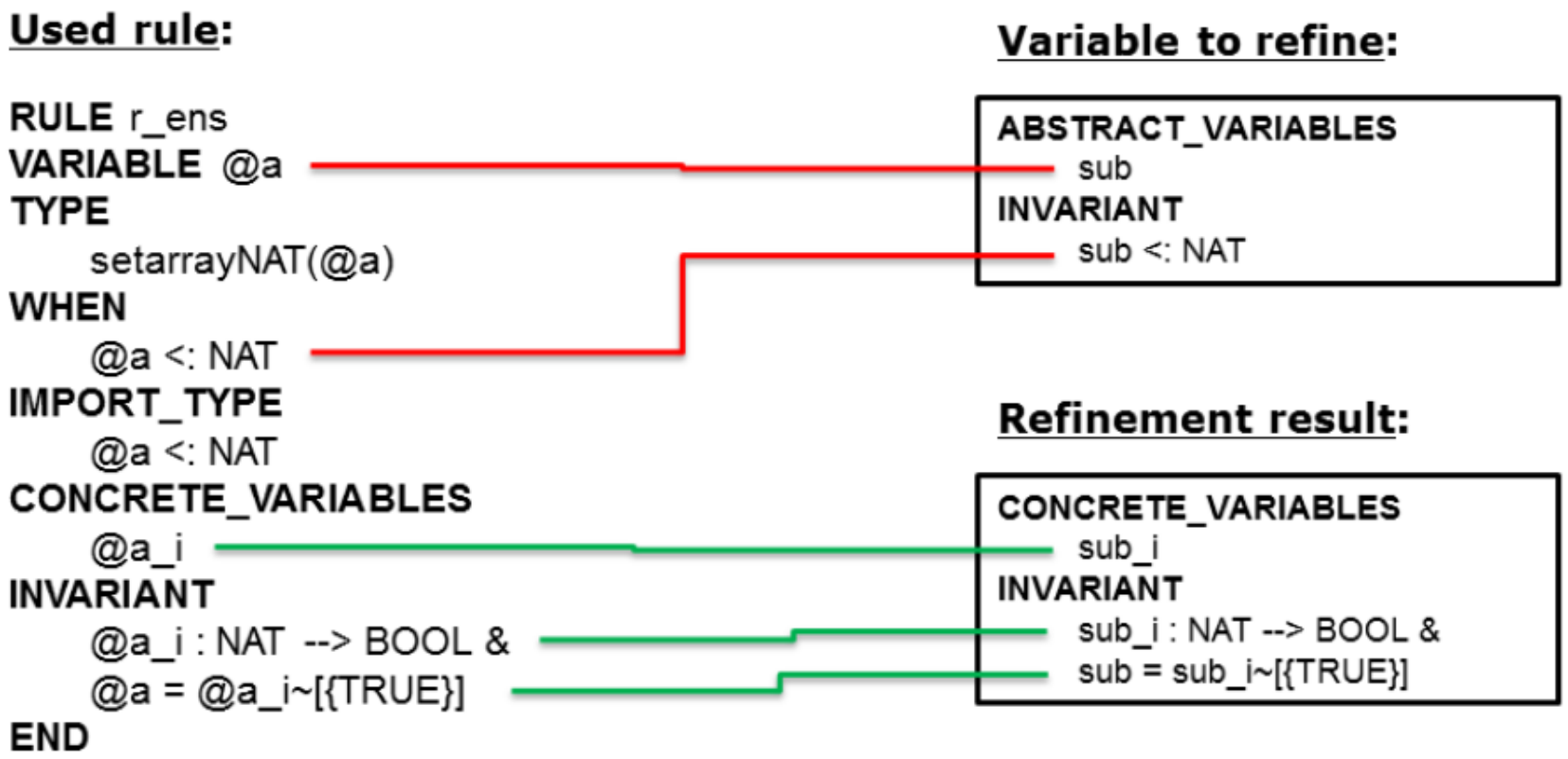
Automatic Refinement, L. Burdy, J.M. Meynadier, FM 1999
Return of Experience on Automated Refinement in B, T. Lecomte, SETS 2014

Automatic Refinement

≡ Not a push-button tool

- Automate what the modeller does manually
- Refine variables then substitutions
- Adding rules is mandatory when automatic refinement fails (iterations)
- Rich language: type information, operation declaration, etc.

```
RULE assign_a_bool_belongs_b_c_16
REFINES
    @a := bool(@b|->@d : @c*@e)
REFINEMENT
    @a := bool(@b:@c & @d:@e)
END;
```



Used rules:

```
RULE assign_a_union_a_b
REFINES
    @a := @a ∨ @b
WHEN
    setarrayNAT(@a)
IMPLEMENTATION
    TYPE_ITERATION(index => #1,
        type => NAT,
        body => (IMPORTED_OPERATION(out => (),
            in => (#1),
            pre => (#1 : NAT),
            body => (@a := @a ∨ ({#1}/\(@b)))),
        invariant => (@a=@a$0∨(NAT_treated/\(@b))))
END
```

```
RULE add_el_to
REFINES
    @a := @a ∨ {@b}
WHEN
    setarrayNAT(@a) &
    B0EXPR(@b)
IMPLEMENTATION
    @a_i(@b) := TRUE
IMPLEMENT
    @a
END
```

```
ABSTRACT_VARIABLES
    sub1, sub2, sub3
INVARIANT
    sub1 <: NAT &
    sub2 <: NAT &
    sub3 <: NAT
OPERATIONS
op1(ee) = PRE
        ee : NAT
    THEN
        sub1 := sub1 ∨ sub2 ||
        sub3 := sub3 ∨ {ee}
END
```

Generated components:

```
CONCRETE_VARIABLES
    sub3_i, vg_loop
INVARIANT
    sub3_i : NAT --> BOOL &
    sub3 = sub3_i~[ {TRUE}]  &
    vg_loop : BOOL
OPERATIONS
op1(ee) =   VAR I_1 IN
    vg_loop <-- init_iteration_NAT ;
    WHILE vg_loop = TRUE DO
        vg_loop, I_1 <-- continue_iteration_NAT ;
        /*? sub1 := sub1 ∨ ({I_1} /\ sub2) ?*/
        op1_1(I_1)
    INVARIANT
        […] &
        sub1 = sub1$0 ∨ (NAT_treated /\ sub2)
    VARIANT
        card(NAT_to_treat)
    END ;
    sub3_i ( ee) := TRUE
END
```

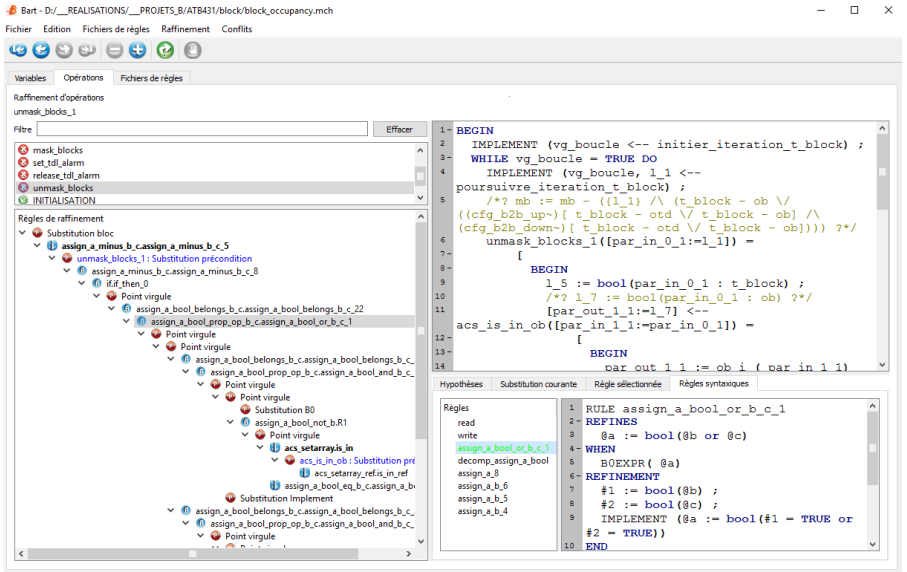
Automatic Refinement

≡ No need to certify the tool

- Whatever the refinement rules, in the engine or added by users
- If the refined model is incorrect, it can't be proved

≡ Impact on projects

- Siemens claimed to divide development costs by 2 when the process is stabilized
- More little steps, more decomposition levels, more function calls, more lines of code
- Refinement patterns ease deployment of proof strategies



B Automatic Refinement Tool (BART) added to Atelier B in 2007, in accordance with Siemens T.S.

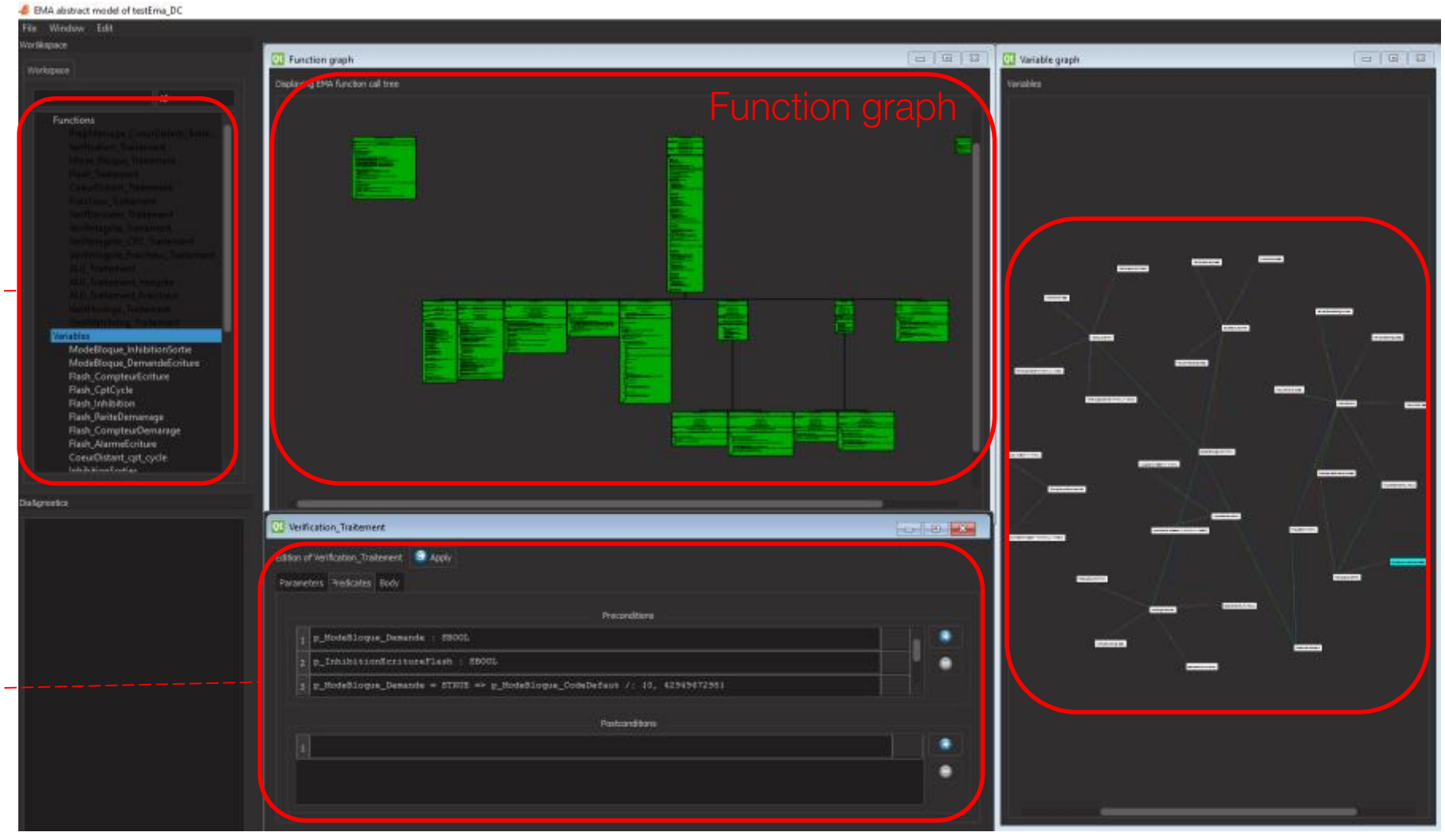
≡ Industrial applications

[2004] Canarsie line (ATP):	*38k lines of handwriten B, 115k lines of generated B*
[2007] Roissy airport shuttle (ATP+ZC):	*40k lines of handwritten B*
	225k lines of generated B
[2007] Symbolic Calculus Engine:	*200k lines of generated B*
[2015] SysML Compiler :	*300k lines of generated B*

Simplified Abstract Modelling

≡ Propose a new way of modelling

- User is incited to think in terms of functions
- Model creation, decomposition, variables allocation is left to the computer



Outline
- Functions
- Variables
- Constants

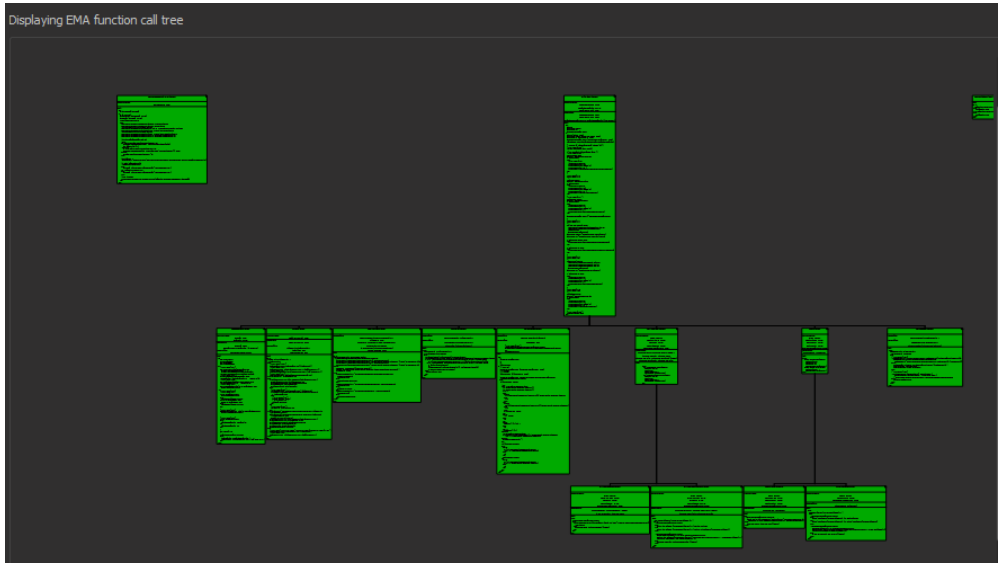
Function
- Parameters
- Predicates
- Body

Variables graph
- What is produced
- What is consumed

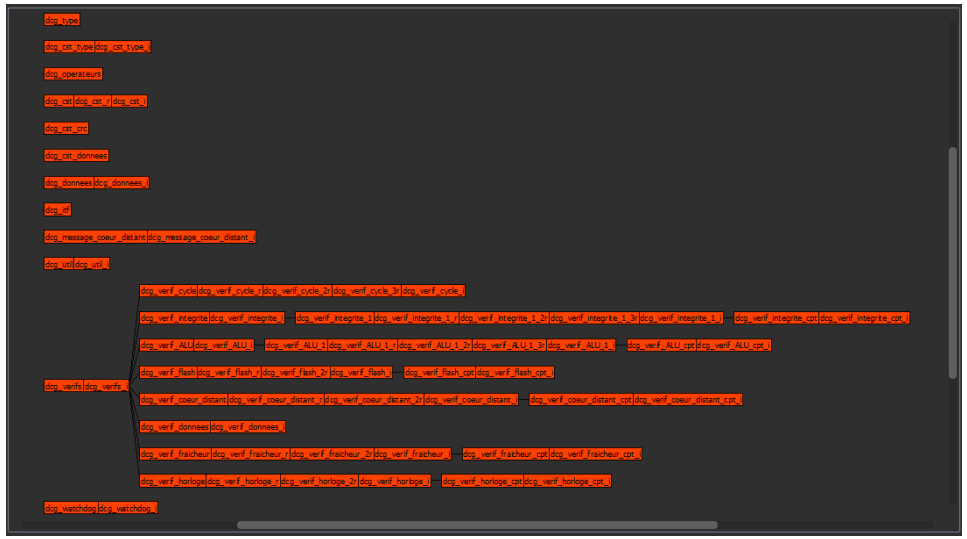
Simplified Abstract Modelling

≡ Two Modelling Views for the Same Project

- Projects can me edited what ever the view
- Providing a clear distinction between abstract model and concrete model
- Corollary: not all projects can be analysed
- Direction continuation of the Automatic Refinement approach



Abstract Modelling Editor (EMA) to appear in Atelier B 4.4



Atelier B current component view

Formal Data Validation

≡ Proving parameters (constants)

- What is the use of a formally proven software if some of its (non trivial) parameters are wrong ?
- Initially metro line static data used by the automatic pilot (software) to drive safely

≡ Data Validation

- Automatic check of large data sets against properties
- Properties : international standards, national regulations, manufacturer habits, customer requirements
- Initially metro line static data used by the automatic pilot (software) to drive safely
- Model-checking applied to

	A	B	C	D	E	F	G	H	I
1	Name	ID	IP	Type	UpLink	DownLink	Length	GPS 1	GPS 2
2	Route_tx_001	243		R	Route_tx_005	Route_vx_002	345		
3	Route_vx_002	128		R	Route_vx_002	EndLine_000	128		
4	Switch_w_003	256	192.16.4.55	S	Route_vx_128	Route_tx_006	23		
5	Relay_s_004	12	192.16.4.10	Y				N 50.85 963	O 6.84 201
6	Route_tx_005	3		R	Route_tx_006	Route_vx_128	291		
7	Relay_s_001	55	192.16.4.125	Y					
8	Route_tx_006	22		R	EndLine_001	Route_vx_002	110		
9	Route_vx_128	127		R	Route_tx_006	Route_vx_002	145		
10	Switch_w_009	242	192.16.4.10	S	Route_vx_128	Route_tx_005	34		
11	EndLine_000	0		E		Route_vx_002	1		
12	EndLine_001	1		E	Route_vx_002		1		
13	Signal_xs_002	32	192.16.4.12	G	Route_vx_128		22		
14	Signal_xs_003	33	192.16.4.13	G	Route_tx_006		51		
15	Balise_b_001	301		B	Route_vx_128		0	N 50.85 933	O 6.84 508
16	Balise_b_002	302		B	Route_tx_005		0	N 50.86 123	O 6.84 550

Up to 100,000+ raw data chunks

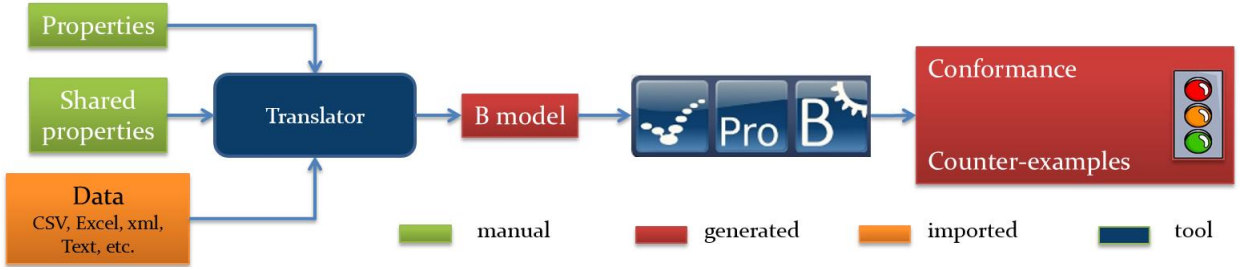
Are they
- Consistent ?
- Correct ?
- Safe ?

Formal Data Validation in the Railways, T. Lecomte, SSS 2016
Using B and ProB for Data Validation Projects, D. Hansen, D. Scheiner, M. Leuschel, book chapter 2016
From Animation to Data Validation: The ProB Constraint Solver 10 Years On, M. Leuschel, J. Bendisposto, I. Dobrikov, S. Krings, D. Plagge, book chapter 2014

Formal Data Validation

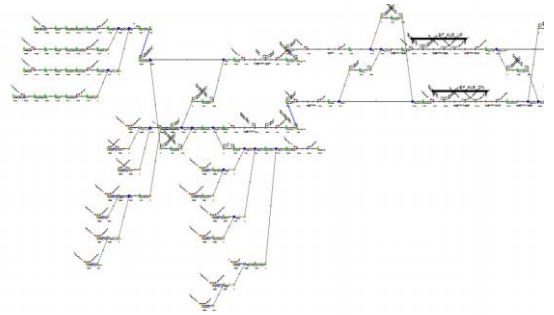
≡ Consistency, correctness, safety

- Expressed with the mathematical language of B
- Work well with graph-based properties
- Provide counter examples when errors found



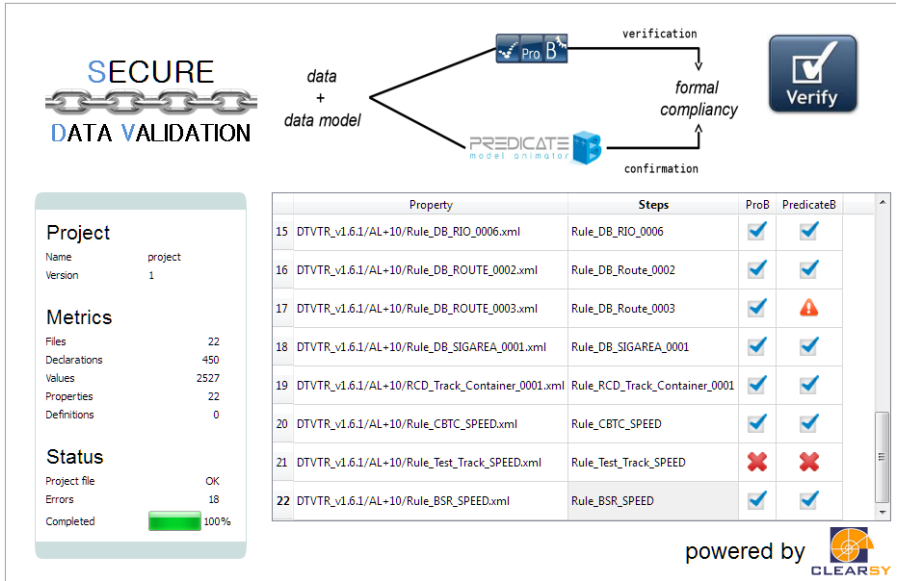
≡ Model-checking

- Performed with ProB
- Rodin (Siemens) and Alstom have funded development and validation to obtain mature tool
- Replace months of (boring) engineer work by hours of computer verification
- Engineer models properties (1 000/line)



≡ Industry-Ready **[PUSH-BUTTON !]**

- Deal with permanent changes in data and properties
- Redundant tools to obtain diversity
- Rules reused from one project to another
- More than 30 sites verified including:
  - Singapore, Panama, Ryad, Mecca



Formal Data Validation

≡ Example of Properties

- Domain Specific Language to express properties
- Direct translation from the DSL to B
- This rule is simple

```
FOR
        sig, ixl
WHERE
        sig : sys_sud_er::Signal &
        sig : dom(sys_sud_er::Signal__dptId) &
        sig : dom(ic::sys_sud_er::signal_geopoint) &
        ic::sys_sud_er::signal_geopoint(sig) : ic::sys_sud_er::zone_GPZone (sys_sud_er::IXL_Core__singleZone(ixl))
THEN
        VERIFY
                sys_sud_er::Signal__dptId(sig) : ran(sys_sud_er::IXL_Core__signal(ixl))
        MESSAGE
                «The signal %1 belongs to IXL_Core %2 territory but is not referenced among its signals.»
                ARG sys_sud_er::Signal__name(sig) TYPE STRING
                ARG sys_sud_er::IXL_Core__name(ixl) TYPE STRING
        ENDVERIFY
ENDFOR
```

Formal Data Validation

≡ Verbosity, implicit made explicit

For each GradientTopology (GradientTopology.BOT-Zone) totally included in a segment, a Gradient (Gradient.BOT-Zone) is created with the same attributes.
For GradientTopology intersecting different segments, several Gradients (Gradient.BOT-Zone) are created so that each of them is located in only one segment.

When the gradient is constant (GradientTopology.isConstant = Yes):
- the variable gradient information (Gradient.VariableGradient) is not set.
- the constant gradient information is set with the same information of GradientTopology for both parts.
- the elevationDifference.elevationEnd of the part1 and elevationDifference.elevationStart of the part2 (reference to the above figure) are equal to elevationStart + gradient*Length1.
- the information isConstant is set to Yes for both parts.

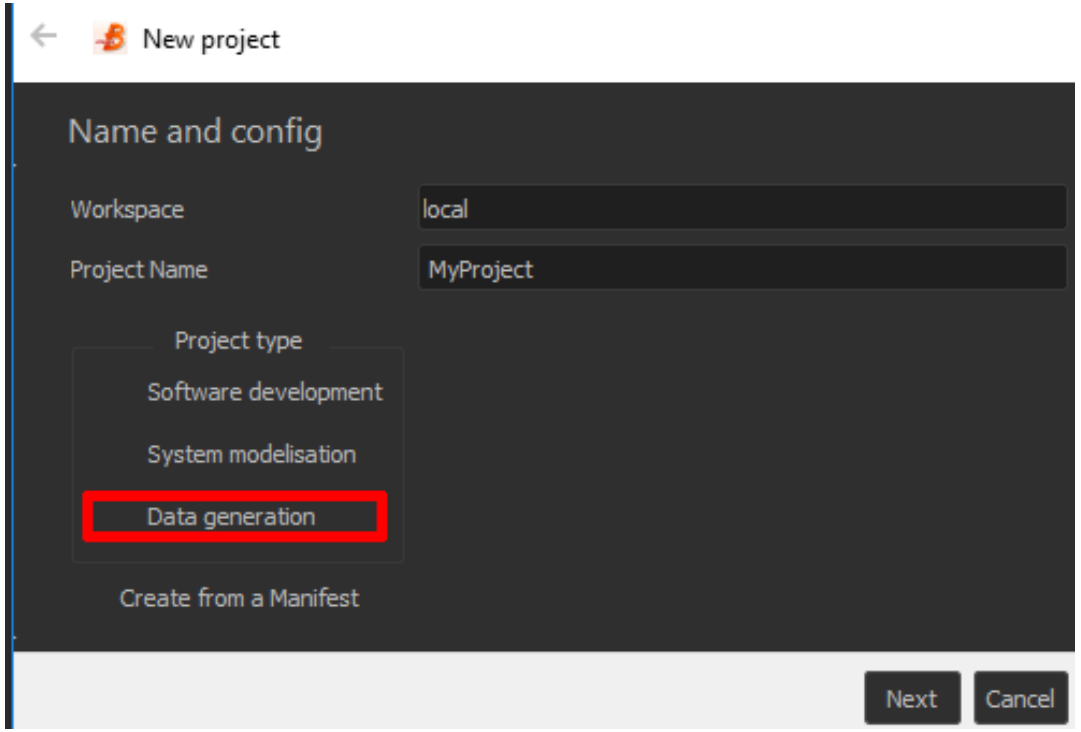
When the gradient is not constant (GradientTopology.isConstant = No):
- the constant gradient information (ConstantGradient) is not set.
- the elevationDifference.elevationEnd of the part1 and elevationDifference.elevationStart of the part2 (reference to the above figure) are equal to elevationStart +2*radius*sin(Length1/ (2*radius))*sin(gradientStart +Length1/ (2*radius)).
- the information radius and transitionCurveType of the variableGradient information are the same for both parts (as initial GradientTopology information) .
- the information gradientEnd for part1 and gradientStart of part2 for variableGradient information are set to (gradientEnd-gradientStart)/(Length1 +Length2)*Length1 + gradientStart.
- the information isConstant is set to No for both Part.

40 lines

Formal Data Validation

≡ New feature

- Added to Atelier B 4.2
- Requires ProB to be installed
- Able to validate constants against properties
- Able to generate/compute missing values



Event-B

≡ Involved in Several R&D projects

- French proposal rejected “because Event-B mathematics do not allow to model systems” (*sic*)

- FP5 **MATISSE** (Methodologies And Technologies for Industrial Strength Systems Engineering)
  - First language draft specification
  - Software-based systems
  - Event-B to B translation to experiment
  - UML-B

- FP5 **PUSSEE** (Paradigm Unifying System Specification Environments for Proven Electronic Design)
  - Electronic-based systems
  - VHDL code-generation

- FP6 **RODIN** (Rigorous Open Development Environment for Complex Systems)

- FP7 **DEPLOY** (Industrial deployment of system engineering methods providing high dependability and productivity)

Non Safety Related Applications

≡ Event-B applied to a number of diverse applications

- Banking (Société Générale)
- Ariane 5 Flight Sequencer Software (Délégation Générale de l'Armement) – *"When we fire a nuclear missile underwater, we want to be sure that the 'door' is open"*
- Micro-Satellites Constellation Flight (Centre National d'Etudes Spatiales)
- Air Traffic Control (Eurocontrol, CEAT) – *"An Air-France flight landed in Brussels with guidance from Zurich airport"*
- Press (Caisse Nationale d'Assurance Maladie)
- De-icing Helicopters (Eurocopter)
- Nuclear Plant (Electricité de France) – *Trust me: you don't want to know*
- Multiplexed Cars (Peugeot, Renault)
- Smartcard (STMicrolectronics)
- Etc.

≡ Studies mostly **for free !**

- Bootstrap activity
- Studies mainly performed with Atelier B (tool "sufficient to handle problems")

≡ No positive conclusion

- understanding,
- compliance with existing development cycle & methods,
- existing teams,
- standards,
- etc.

≡ Except

- Vehicles (diagnosis, assistance to integration)
- Smartcard (certification, proven product)
- Railways (safety critical systems)

Component-Based System Description

≡ Flat Behavioural Specification (maintenance, failure diagnosis)

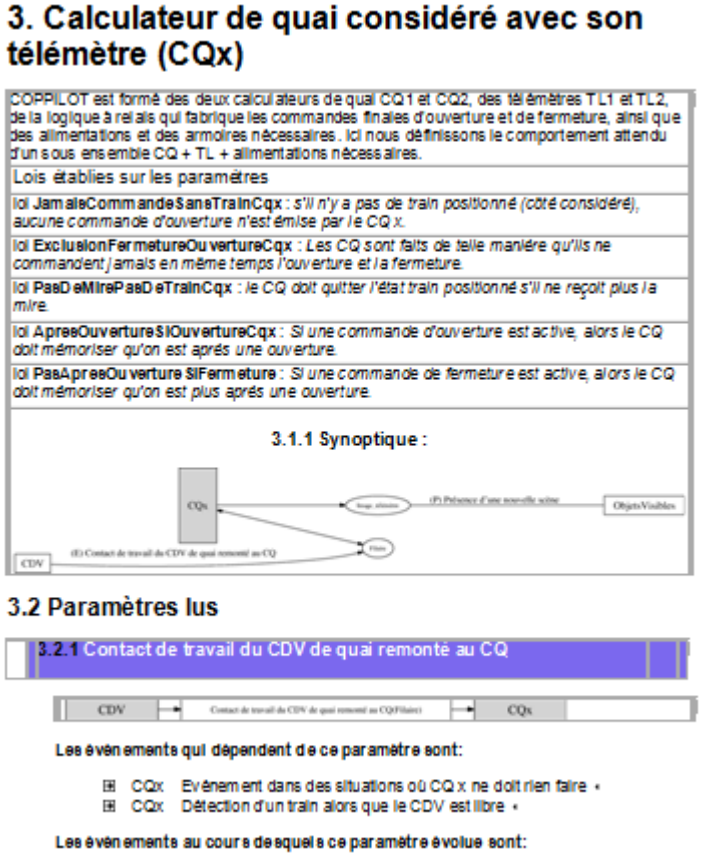
- Event-B to describe behaviour, light proof
- Find ambiguities in the specification documents (thousands pages)
- Events are allocated to components
- Analyse model structure to deduce dependencies: where is produced, where is consumed
- Generate documentation from model and additional information

B model

```
F_Opening =
SELECT
   CommandOpening = TRUE
THEN
   PSDOpen := TRUE
END;
```

Dictionnary

```
VARIABLE PSDOpen
MEDIUM Internal
TRANSLATION
   Status of the non locked PSD: open or not
END

TRANSLATION PSDOpen:=TRUE
   PSD are opening
END
```



3. Calculateur de quai considéré avec son télémètre (CQx)

COPILOT est formé des deux calculateurs de quai CQ1 et CQ2, des télémètres TL1 et TL2, de la logique à relais qui fabrique les commandes finales d'ouverture et de fermeture, ainsi que des alimentations et des armoires nécessaires. Ici nous définissons le comportement attendu d'un sous ensemble CQ + TL + alimentations nécessaires.

Lois établies sur les paramètres

loi **JamaisCommandeSansTrainCqx** : *s'il n'y a pas de train positionné (côté considéré), aucune commande d'ouverture n'est émise par le CQ x.*

loi **ExclusionFermetureOuvertureCqx** : *Les CQ sont faits de telle manière qu'ils ne commandent jamais en même temps l'ouverture et la fermeture.*

loi **PasDeMirePasDeTrainCqx** : *le CQ doit quitter l'état train positionné s'il ne reçoit plus la mire.*

loi **ApresOuvertureSiOuvertureCqx** : *Si une commande d'ouverture est active, alors le CQ doit mémoriser qu'on est après une ouverture.*

loi **PasApresOuvertureSiFermeture** : *Si une commande de fermeture est active, alors le CQ doit mémoriser qu'on est plus après une ouverture.*

3.1.1 Synoptique :

3.2 Paramètres lus

3.2.1 Contact de travail du CDV de quai remonté au CQ

Les événements qui dépendent de ce paramètre sont:

- CQx Evénement dans des situations où CQ x ne doit rien faire •
- CQx Détection d'un train alors que le CDV est libre •

Les événements au cours desquels ce paramètre évolue sont:

≡ Modelling Main Issue

- What could be done when millions hours have been spent designing a car ?

Component-Based System Description

≡ Tooling

- Scripts developed during modelling sessions turned into a standalone tool: Composys
- Based on Atelier B BCompiler

Killed In Action 2012

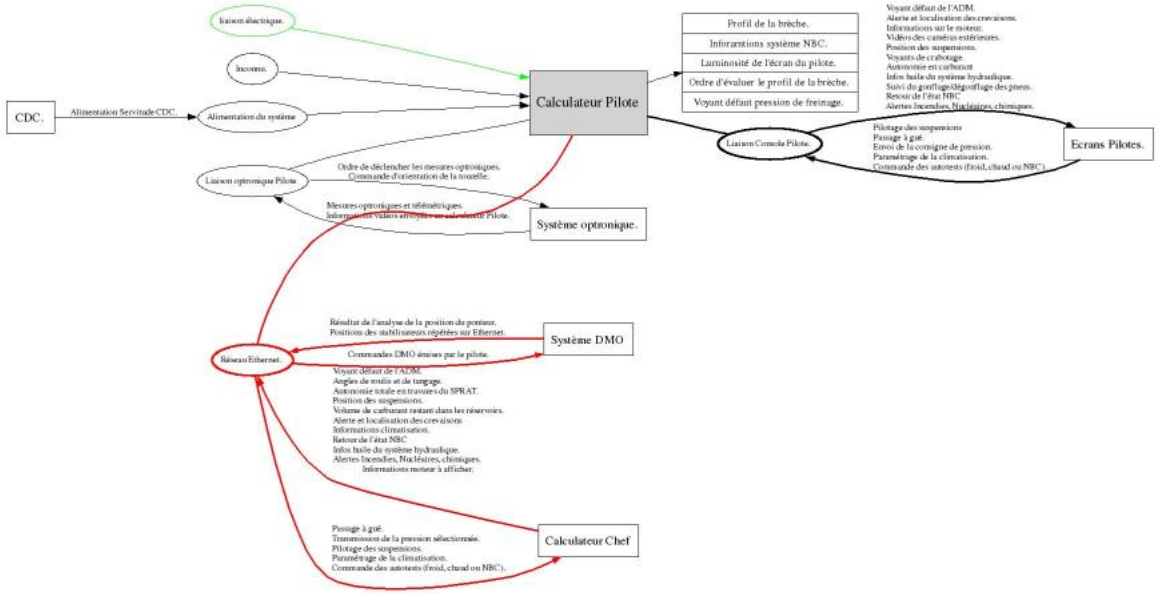
≡ Cars Modeling

- Applied to the failure modelling of 4 Peugeot cars in order to locate faulty ECU(s)
- To help mechanics to diagnose failures
- Stopped in 2005, Peugeot switched back to only provide error codes without further analysis

```
Reponse_ID =
SELECT
    vl_couplage = normal &
    Attente_reponse = TRUE &
    PresenceIDCoteDroit = TRUE
THEN
    ID_Reponse :: {REP_pirate, REP_PSA} ||
    vl_couplage  := ph_Emission_reception

END
```

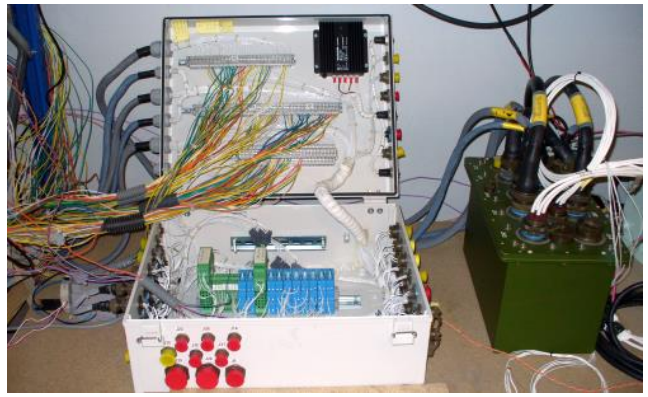
≡ Military Vehicule Modeling

- Applied to the design of unique vehicle designed from scratch
- To help integrating tons of systems



Component-Based System Description

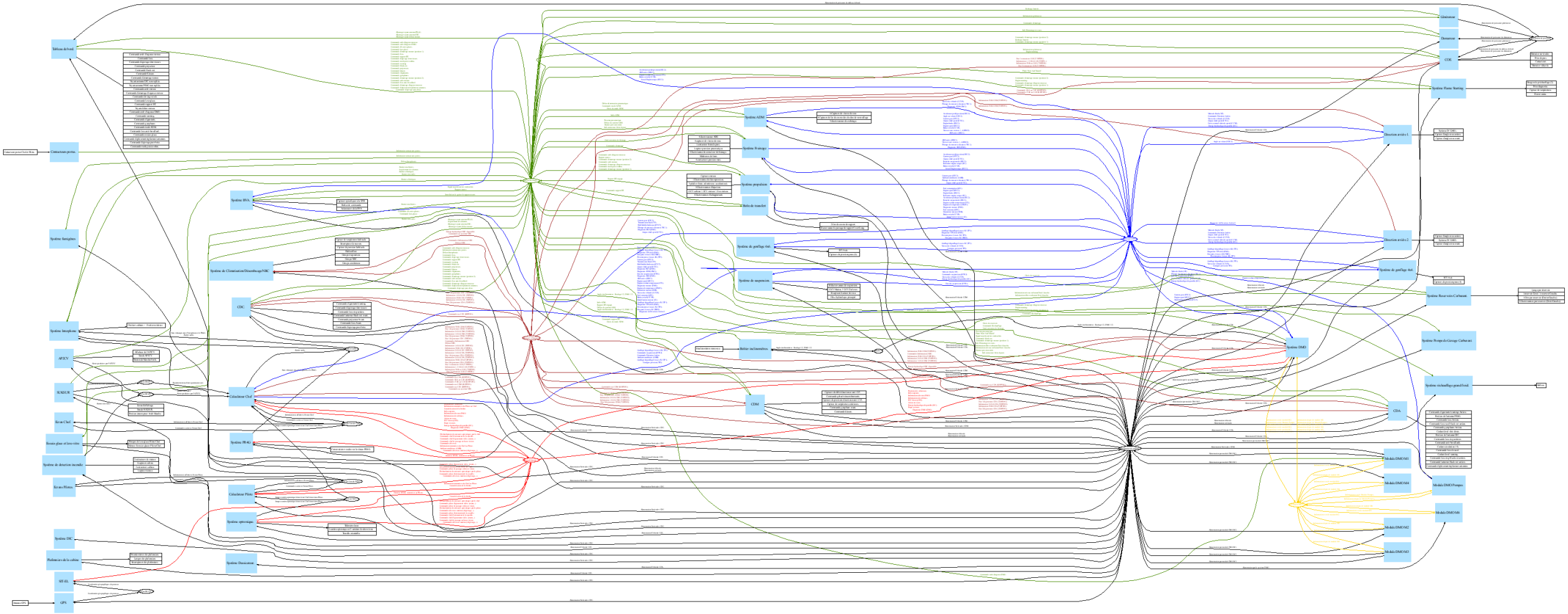
≡ On Going Integration



≡ The Resulting Vehicle

Component-Based System Description

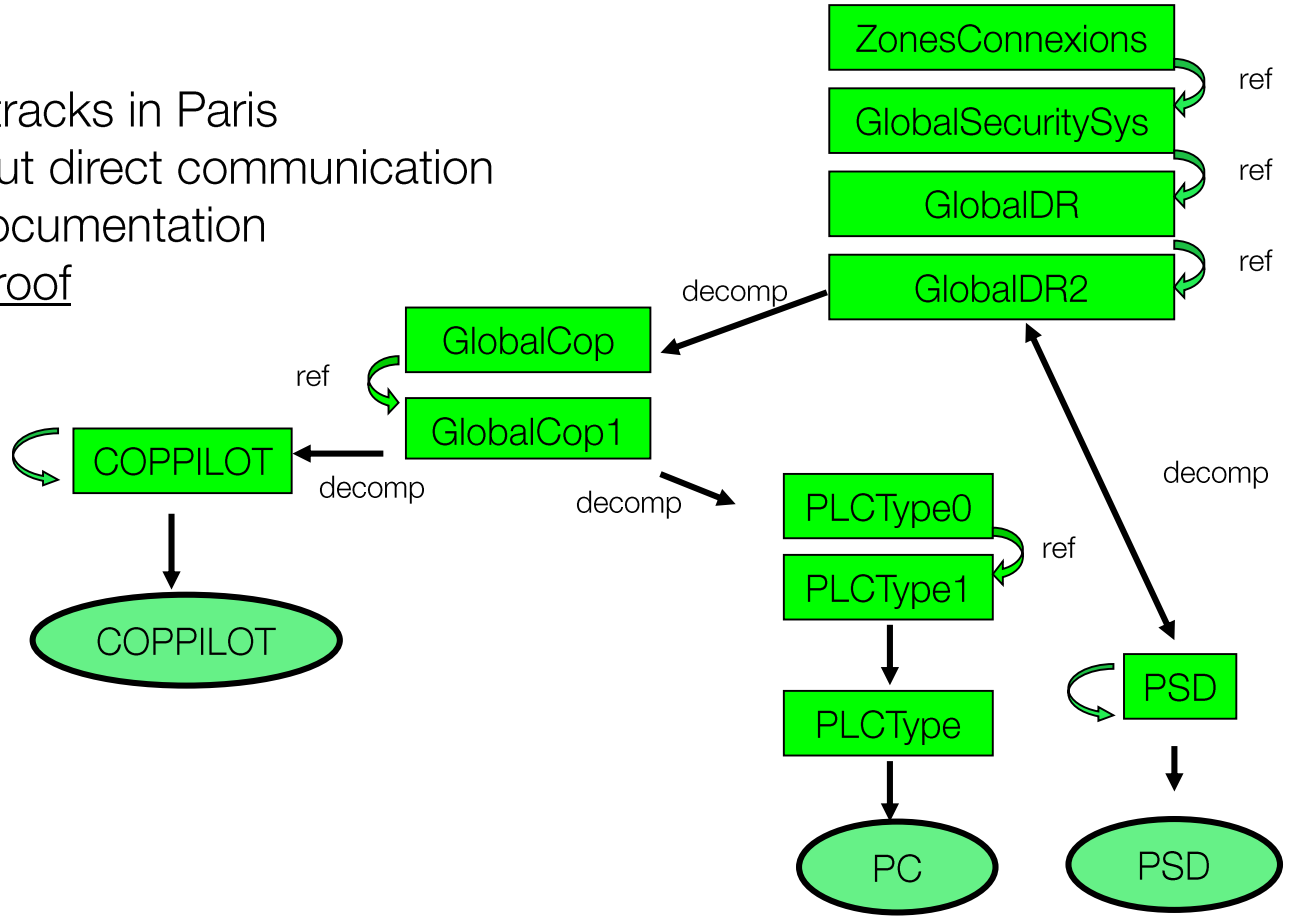
≡ Complex Relationships



Component-Based System Description

≡ Platform Screen Doors (railways)

- New system to avoid people to die (pushed) on tracks in Paris
- Trick: detect train arrival and door opening without direct communication
- Composys to analyse the system and provide documentation
- Finally an Event-B model to ensure safety with proof



≡ Real System on Paris L13 (proof of concept)

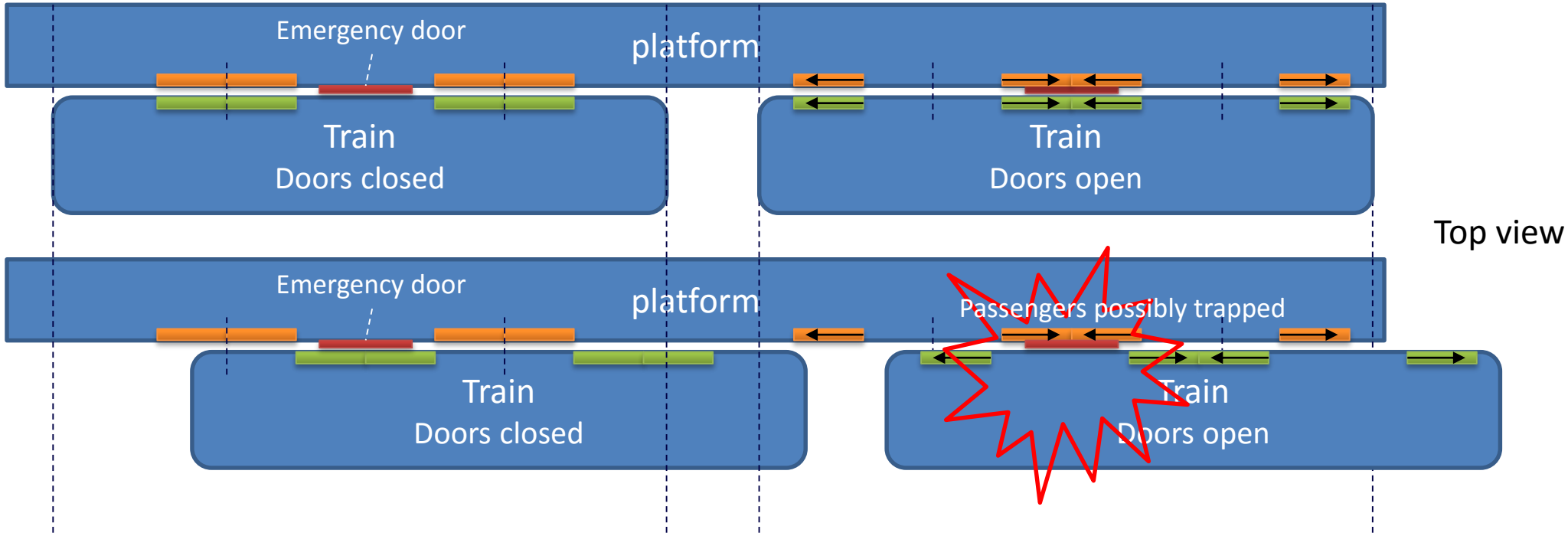
- Developed from scratch in 6 months
- Lot of missing skills to acquire
- Software developed manually from formal specification
- SIL4: one failure every 10 000 years
- 99,999% reliability: one train max missed per year
- one year successful experiment: no death, no train missed

Safe and Reliable Metro Platform Screen Doors Control/Command Systems, T. Lecomte, SBMF 2008

Component-Based System Description

≡ Proved but still unsafe …

- Proven Event-B model presented to RATP
- But major issue uncovered by RATP technician



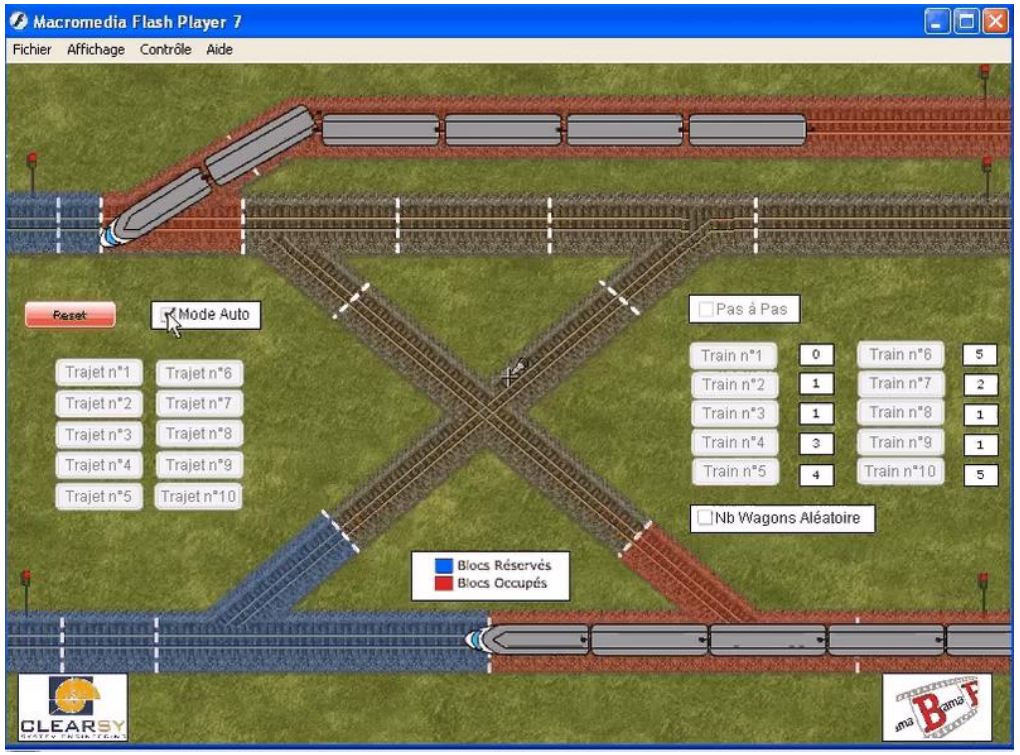
≡ Specification changed: “do not open PSD in case of incorrect train position”

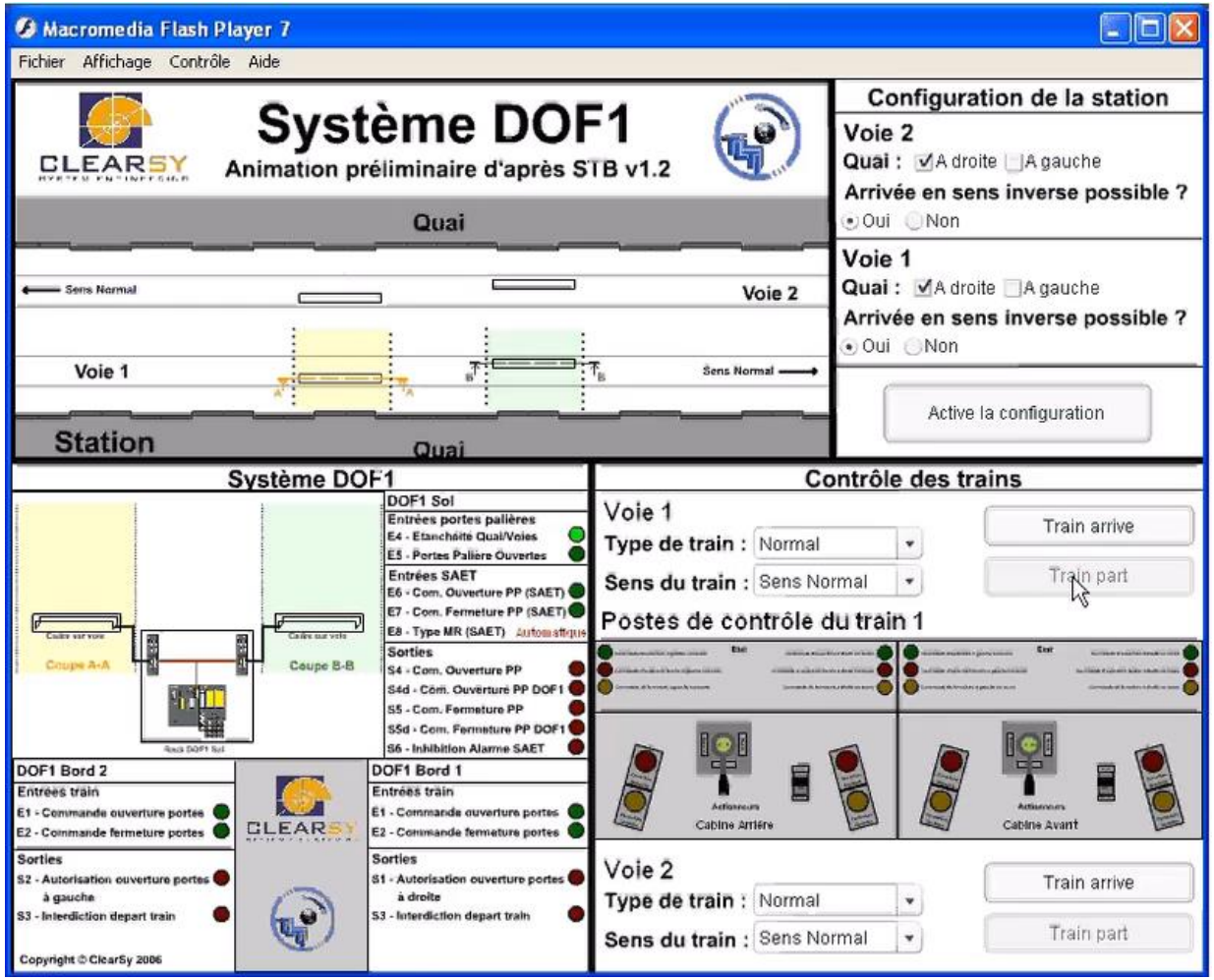
- Did we select the “correct” dimensions to analyse the system ?
- What about Japanese tourists taking pictures on the other platform, birds flying through the tunnels, etc. ?

The model

≡ For the next call for tender (Paris line 1), we were ready

≡ Brama Model Animator (Rodin)

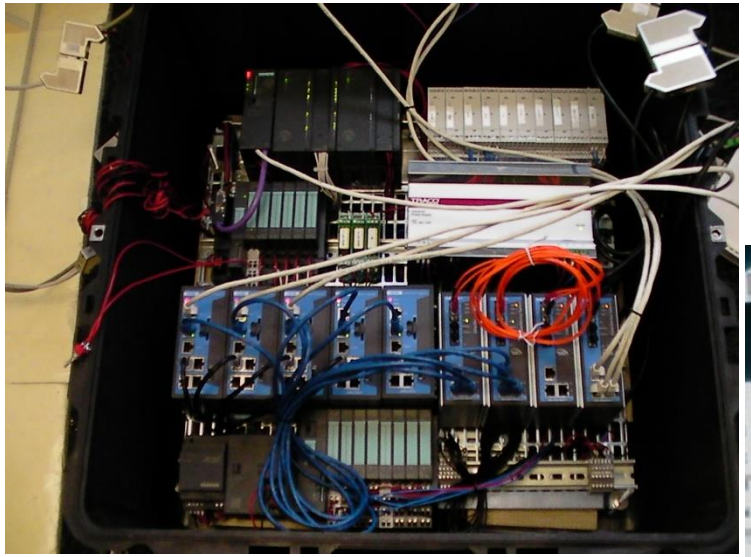


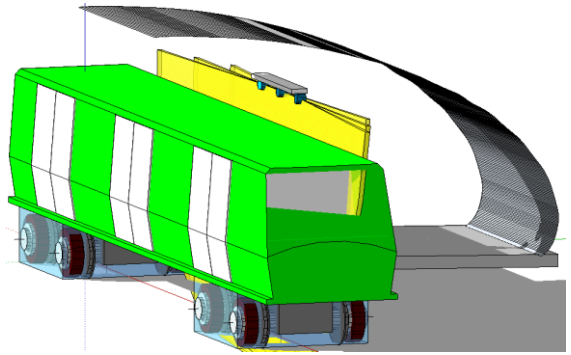


The model

≡ When it is the first time, it is the first time !

Scale 1 DIY model





The model

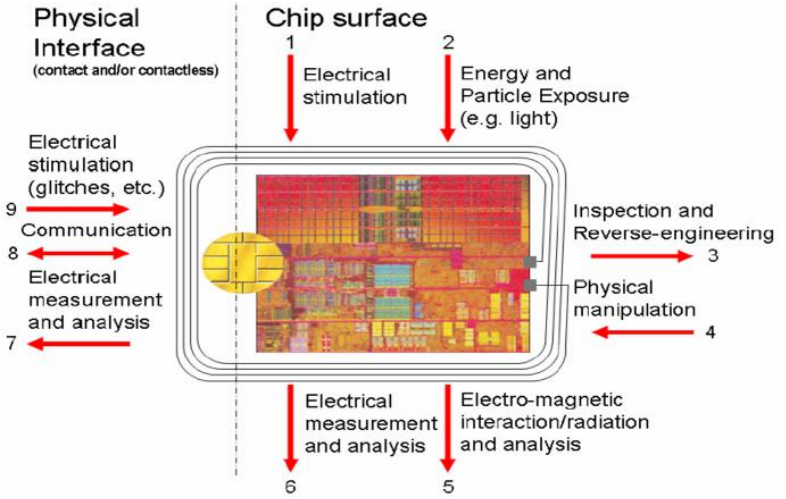
≡ Connecting things with the real station

Et voilà !

Security & Common Criteria

≡ Smartcard industry demands for higher security levels

- … for marketing reasons
- First attempt:
  - Model API functions (library available to third-party)
  - Model Security Policy (define what access conditions are)
  - Prove that Security Policy is enforced by the functional specification
  - Check (traceability) that specification is implemented
  - Compliant with EAL5+ (Common Criteria 2.2)
  - 1 Event-B model, 4 levels of refinement
- Replayed several times for different companies and products
- Demonstration is limited to non-disturbed behaviour *"When it works, it works"*

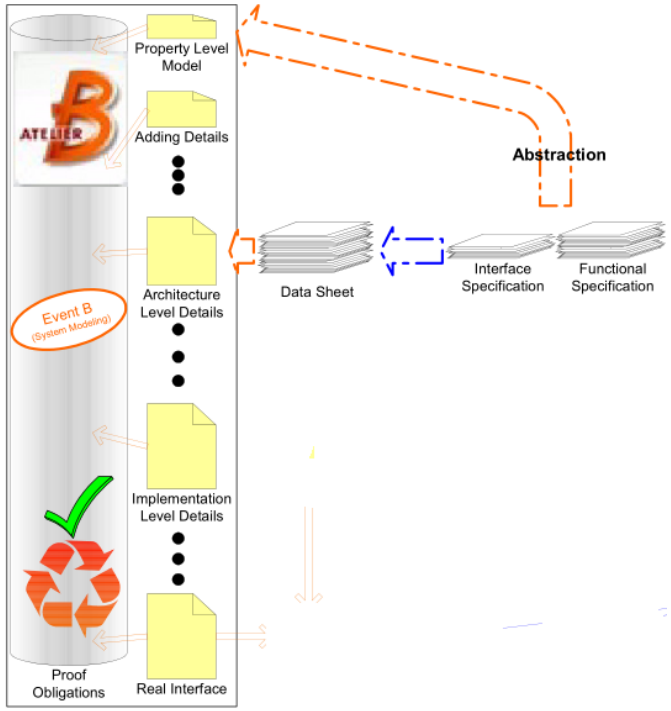


≡ Down to the hardware

- Same approach than above, ***Rodin not op***, Event-B support added to Atelier B
- Component to study: Memory Protection Unit
- 1 Event-B model, 18 levels of refinement, ready for VHDL code generation
- Development: 6 months, 183 versions, 5 major refactoring
- Final product: 20 000 gates

18 levels for an IF-THEN-ELSE ???

How many levels for Intel Core i7 ?

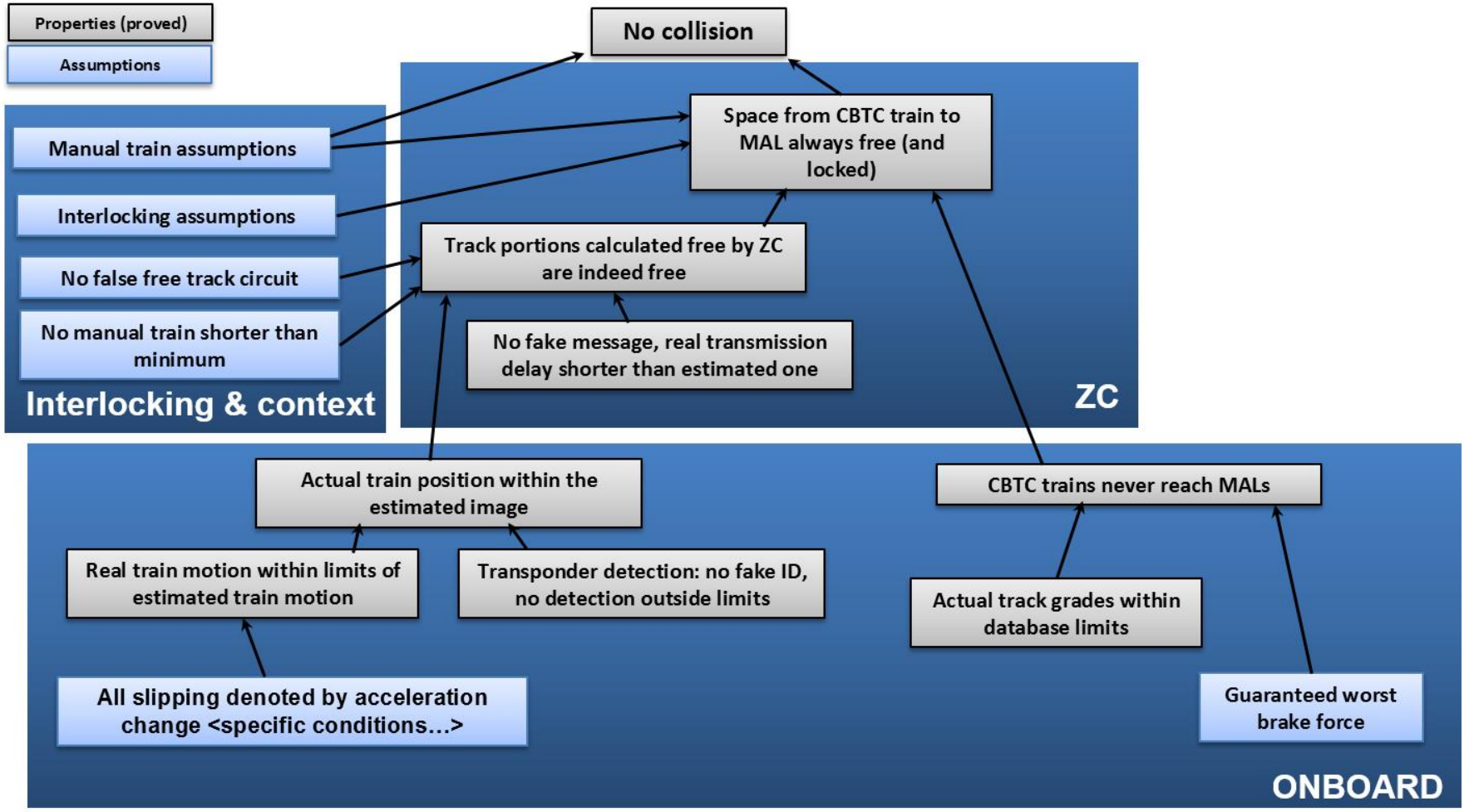


A « Correct by Construction » Realistic Digital Circuit, M. Benveniste, RIAB/FM 2009

On Using B in the Design of Secure Micro-controllers: An Experience Report, M. Benveniste, B 2011

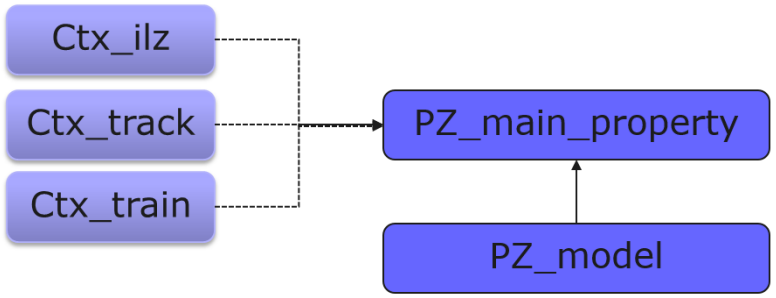
Safety Demonstration

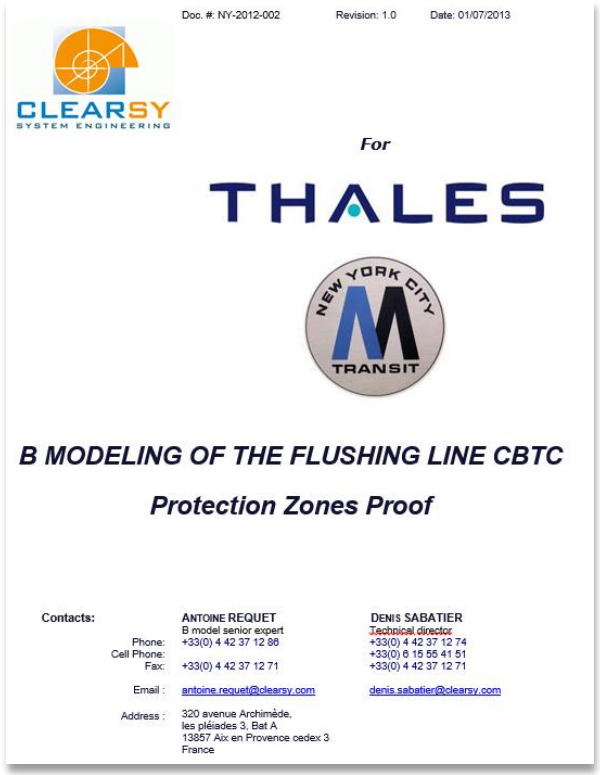
- 18 levels of refinement for a 20k gates micro-electronic component
- How many zillions levels required to model a metro line **and** address very low level details ?
- Challenge: proving that a new CBTC (Thales) enforces safety on New York L7 metro
  - No structural “model of everything”
  - End-product readable and understandable by customers
  - Reusable on other lines



Safety Demonstration

- Protection zone proof: 35 assumptions, 2 subproofs, 30 events, 1400 loc model
- 2 years to complete
- NYCT analysed positively the proof documents
- Reused on Culver line (50% effort reduction)
- Experimented on:
  - ERTMS Regional
  - Paris metro (xx lines automated until 2030)




Doc. #: NY-2012-002 Revision: 1.0 Date: 01/07/2013

CLEARSY SYSTEM ENGINEERING

For

THALES

NEW YORK CITY TRANSIT

B MODELING OF THE FLUSHING LINE CBTC

Protection Zones Proof

Contacts:

ANTOINE REQUET
B model senior expert
Phone: +33(0) 4 42 37 12 86
Cell Phone:
Fax: +33(0) 4 42 37 12 71
Email : antoine.requet@clearsy.com
Address : 320 avenue Archimède, les pléiades 3, Bat A 13857 Aix en Provence cedex 3 France

DENIS SABATIER
Technical director
+33(0) 4 42 37 12 74
+33(0) 6 15 55 41 51
+33(0) 4 42 37 12 71
denis.sabatier@clearsy.com

Proof document: 90 pages

Reference:
D. Sabatier, L. Burdy,
New York Metro Flushing line: System level formal verification , June 2013

Real & Floating Point Numbers

≡ Following Air France Rio-Paris flight crash

- Requests for “proven software” for inertial centres
- Raw requirement: attitude matrix should keep a unit norm with a bounded error **ε**

≡ New types: REAL, FLOAT

- FLOAT: new B0 type
- REAL: to specify ideal algorithms without considering precision
- Floating point literals are not accepted

≡ New operators

- Floating point literals are not accepted
- No predefined operator for converting float into real and float into integer (and vice-versa).

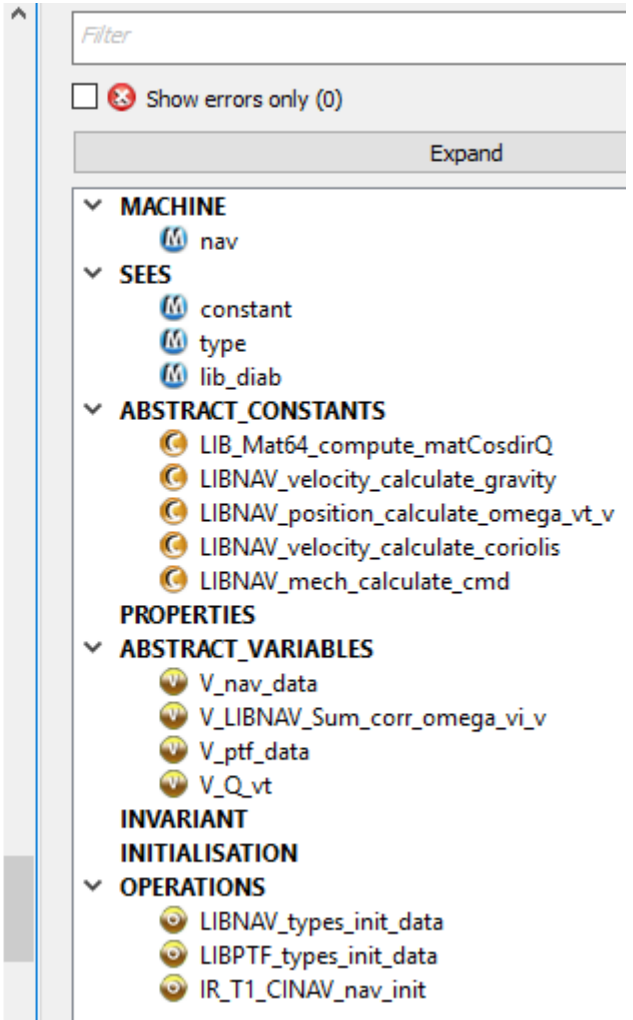
Unified	Integer	Real	Float
`x <= y`	`x <= y`	`x rle y`	`x <=. y`
`x < y`	`x < y`	`x rlt y`	`x <. y`
`x >= y`	`x >= y`	`x rge y`	`x >=. y`
`x > y`	`x > y`	`x rgt y`	`x >. y`
`x + y`	`x + y`	`x rplus y`	`x +. Y`
`- x`	`- x`	`0.0 rminus x`	`-. X`
`x - y`	`x - y`	`x rminus y`	`x <=. y`
`x * y`	`x * y`	`x rmul y`	`x *. Y`
`x / y`	`x / y`	`x rdiv y`	`x /. y`
`x ** y`	`x ** y`	`x rpow y`	***Invalid***
`min(x)`	`min(x)`	`rmin(x)`	***Invalid***
`max(x)`	`max(x)`	`rmax(x)`	***Invalid***
`SIGMA(x).(y | z)`	`SIGMA(x).(y | z)`	`rSIGMA(x).(y | z)`	***Invalid***
`PI(x).(y | z)`	`PI(x).(y | z)`	`rPI(x).(y | z)`	***Invalid***

Real & Floating Point Numbers

≡ New proof obligations

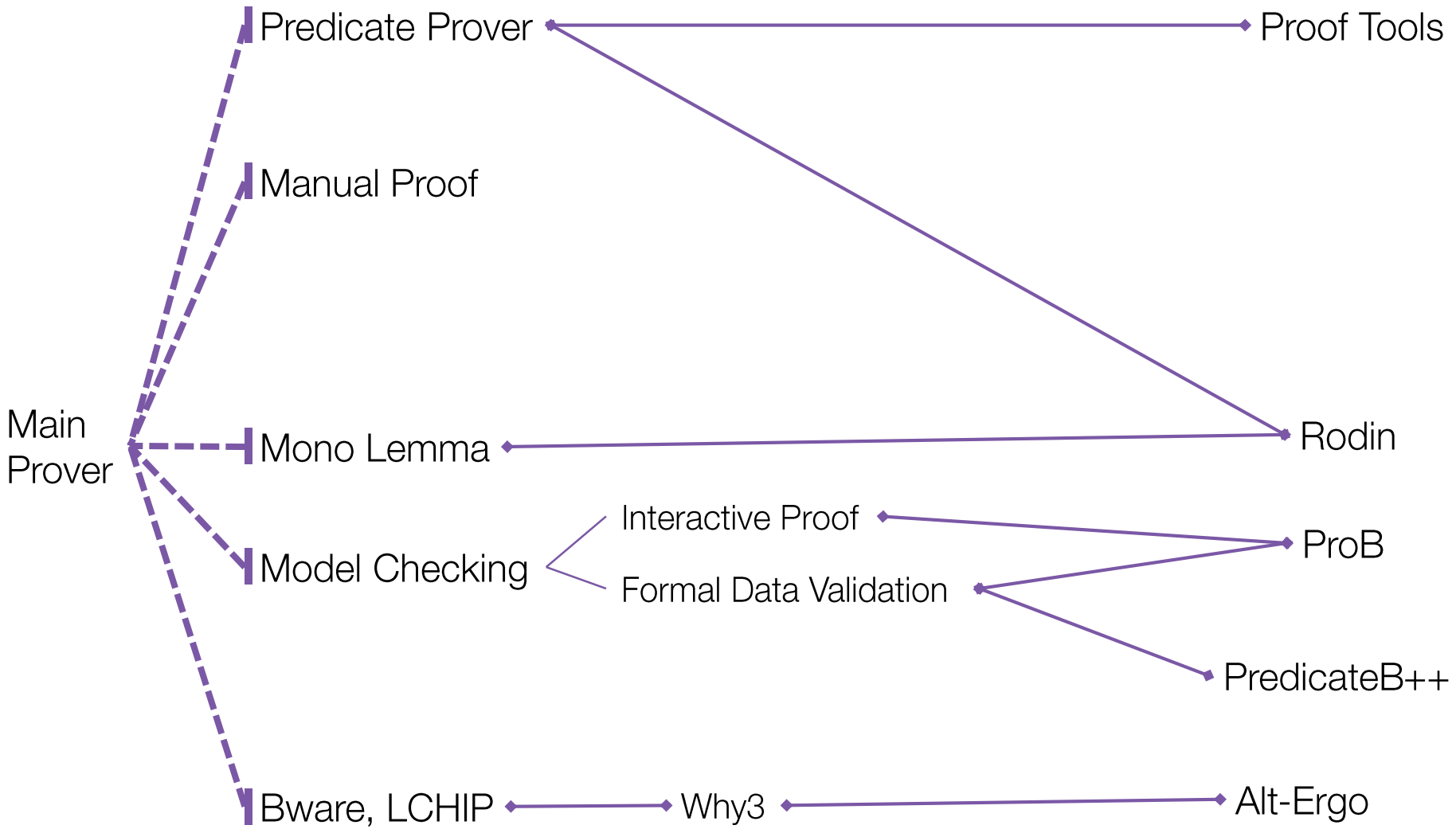
- Not currently handled by any prover
- Bware project would contribute to it

```
    LIBPTF_types_init_data =
    BEGIN
        /* Init pure inertial attitudes */
        V_ptf_data := rec ( pure_inertial_attitudes : rec ( Roll : 0.0 ,
Pitch : 0.0 , Azimuth : 0.0 ) ,
            T_vm : % xx . ( xx : 0 .. 2 | ( 0 .. 2 ) * { 0.0 } ) ,
            T_vb : % xx . ( xx : 0 .. 2 | ( 0 .. 2 ) * { 0.0 } ) ,
            deltaV_v : ( 0 .. 2 ) * { 0.0 } ,
            q_vm : ( 0 .. 3 ) * { 0.0 } )
    END ;

    IR_T1_CINAV_nav_init ( p_Velocity_InitVALUE , p_Position_InitVALUE ,
        v_vertical_velocity , v_altitude , v_deltaT ) =
    PRE
        v_deltaT : tFloat32 &
        p_Velocity_InitVALUE : tVect3_64 &
        p_Position_InitVALUE  : tVect3_64 &
        v_altitude : tFloat64 &
        v_vertical_velocity : tFloat64
    THEN
        LET
            pa_coriolis_correction ,
            v_vect_pos_init
        BE
            /*! Init Velocity */
            pa_coriolis_correction = ( 0 .. 2 ) * { 0.00 } &
            v_vect_pos_init = ( 0 .. 2 ) * { 0.00 }
        IN
```



Proving

Proving


Predicate Prover
Proof Tools
Manual Proof
Main Prover
Mono Lemma
Rodin
Interactive Proof
ProB
Model Checking
Formal Data Validation
PredicateB++
Bware, LCHIP
Why3
Alt-Ergo

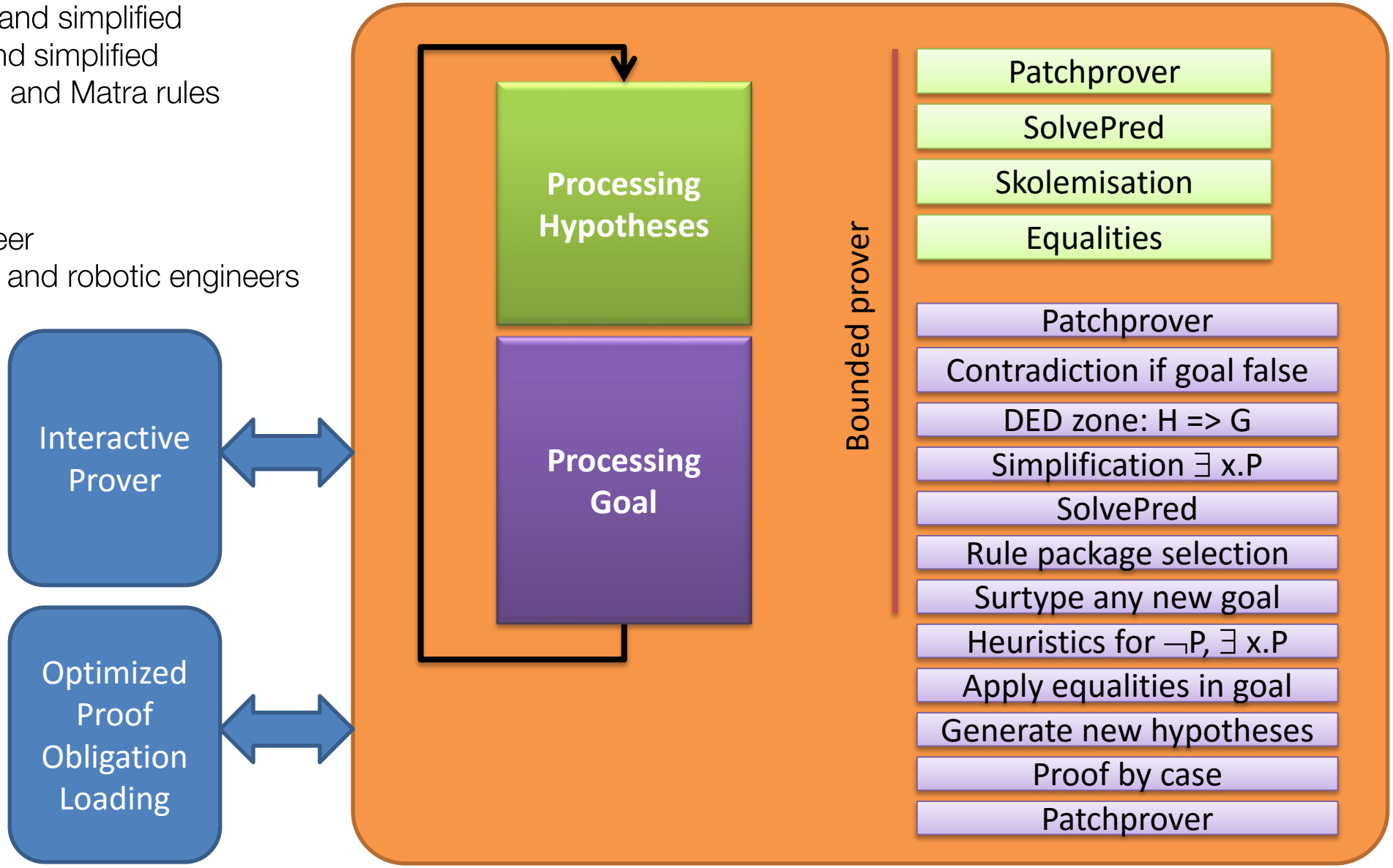
Main Prover

≡ Rule-based Proof Engine

- Predicates are decomposed and simplified
- Hypotheses are generated and simplified
- Set of rules includes Alsthom and Matra rules

≡ State of the Art

- Invented by a signalling engineer
- Improved by electrotechnician and robotic engineers



Main Prover

≡ Ad-hoc programming language: THEORY language

- Prolog-like
- Built-in B parsers
- Programs transformed in bytecode programs executed by a VM
- Integrated to Rodin as a plugin (Mono Lemma version)

```
IDENT(InRelationXY.11)
binhyp(r: s <-> t) &
binhyp(q: s <-> t)
=>
(r/\q: s <-> t);
```

Leaf rule

```
IDENT(InRelationXY.7)
(dom(b)<<|a: s <-> t) &
(b: s <-> t)
=>
(a<+b: s <-> t);
```

Equivalence rule

```
THEORY SimplifyDisjX IS

    bcall(WRITE: bwritef("Impossible
    =>
    p;

    SimplifyDisjG(A | a | R);

    bcall(RES: bresult(R))
    =>
    SimplifyDisjG(a | a | R);

    bsearch(a, (A or bfalse), B) &
    bcall(RES: bresult(R or B))
    =>
    SimplifyDisjG(A | a | R);

    SimplifyDisjG(A | a | R)
    =>
    SimplifyDisjG(A or a| a | R);

    bsearch(a, (B or bfalse), C) &
    SimplifyDisjG(A | a | R or C)
    =>
    SimplifyDisjG(A or B | a | R)

END
```

≡ Fixed point

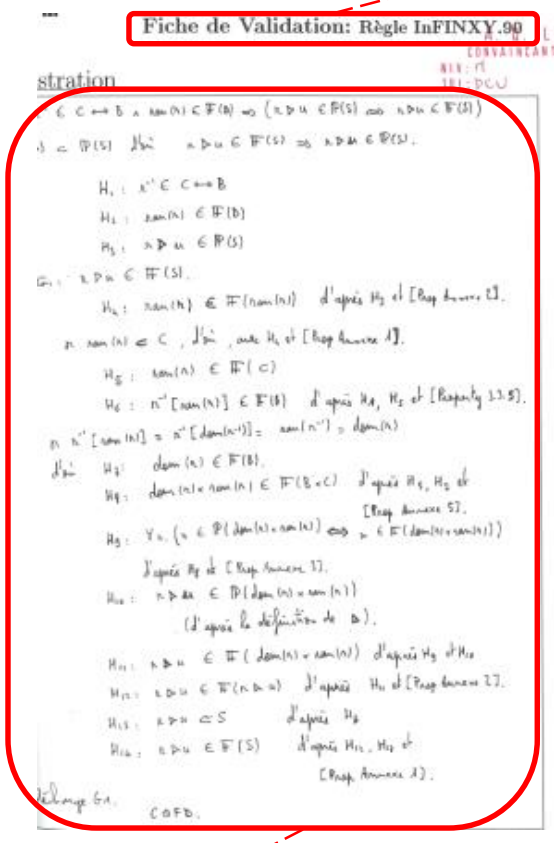
- 1 proof obligation to demonstrate manually == 35 €
- No proof regression allowed with atelier B releases
- Since 1998, evolutions are new interactive commands and trigger-able sets of rules

Main Prover

≡ 2 700 rules to validate

- Manual demonstrations
- Cross-verification
- Third-party verification

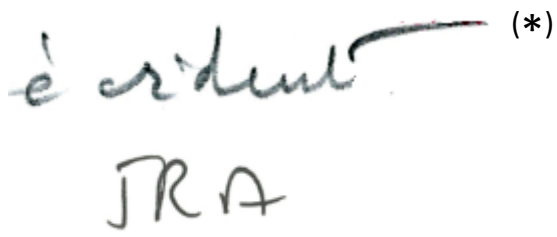
Rule name



Manual demonstration

```
hyp2   f~: s +-> NATURAL
hyp3   e: s
hyp4   not(e: ran(f))
but    (f<-e)~: s +-> NATURAL

hyp5   f<-e = f\/{(size(f)+1)|->e}                    DR11 sur L (Hyp1 Hyp3)

but    (f\/{(size(f)+1)|->e})~: s +-> NATURAL
  <=   f~\/{(size(f)+1)|->e}~: s +-> NATURAL          Inverse Properties
  <=   f~\/{e|->(size(f)+1)}: s +-> NATURAL           Inverse Properties
1.1 <= f~: s +-> NATURAL                              Membership Properties
1.2   & {e|->(size(f)+1)}: s +-> NATURAL
1.3   & dom({e|->(size(f)+1)}<|f~ = dom(f~)<|{e|->(size(f)+1)}

1.1   decharge par Hyp2

1.2   decharge par Hyp3

1.3   dom({e|->(size(f)+1)})<|f~
    = {e}<|f~                                         Domain Properties
    = (f|>e)~                                         Inverse Properties
    = {}~                                             Restriction Properties et Hyp4
    = {}                                              Inverse Properties
    = ran(f)<|{e|->(size(f)+1)}                       ran(f)/\{e}={} (Hyp4)
    = dom(f~)<|{e|->(size(f)+1)}                      Domain Properties
1.1, 1.2 et 1.3 dechargent le but.

2)    On applique DED:
hyp2   (f<-e)~: s +-> NATURAL
2.1    f~: s +-> NATURAL
2.2    e: s
2.3    not(e: ran(f))

hyp3   e: s                                           type-checking de Hyp2
hyp4   f<-e = f\/{(size(f)+1)|->e}                    DR11 sur L (Hyp1 Hyp3)
hyp5   (f\/{(size(f)+1)|->e})~: s +-> NATURAL  EQL1 sur Hyp2 avec Hyp4
hyp6   f~\/{(size(f)+1)|->e}~: s +-> NATURAL   Inverse Properties
hyp7   f~\/{e|->(size(f)+1)}: s +-> NATURAL    Inverse Properties
hyp8   f~: s +-> NATURAL                              par contradiction avec Hyp7

2.1    decharge par Hyp8

2.2    decharge par Hyp3

2.3    par contradiction (Rule 5), on suppose e: ran(f) et on montre:
```

A detailed demonstration



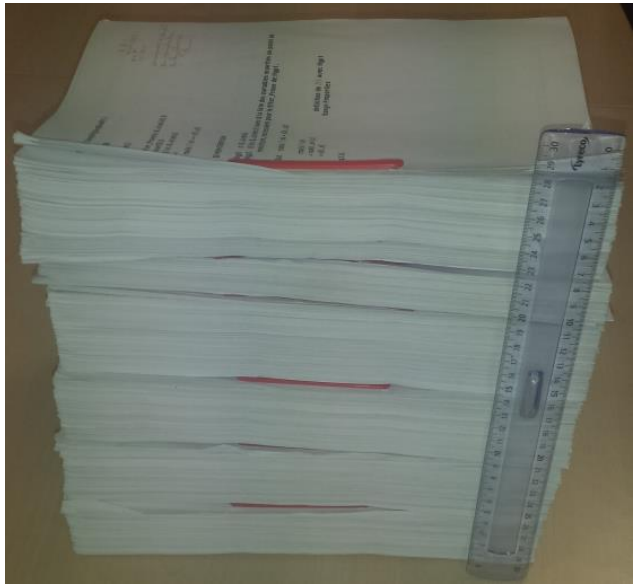
A concise one !

(*) « Obvious »

Testimony:
a rule modified 9 times by 2 experts
while being constantly wrong

≡ The resulting validation forms

- 28 cm



Predicate Prover

≡ Tableau method

- Invented by JR Abrial to prove Atelier B rules
- Used to validate the majority of the 2700 rules of the Main Prover
- Became an interactive command with means to select reduced set of hypotheses
- Integrated to Rodin as a Plugin

```
/** BOOL31 **/
        INFRULE(BOOL31)
        ((v = FALSE) => Q)
=>
        (not(v = TRUE) => Q);
```

≡ Tableau method

- Hypotheses combined with not(goal) to obtain a contradiction
- Heuristics for wise instanciation
- Limited number of rules (116)
- Efficient when the number of hypotheses is low

```
/** ECTR6 **/
        INFRULE(ECTR6)
        binhyp(l=m) &
        bnot(bgoal(not(L)=>M)) &
        bnot(bgoal(!x.H=>N)) &
        band(binhyp(F=E),
        band(bsubfrm(E,F,P,R),
        binhyp(not(R))
       ))
=>
        (P => Q);
```

≡ Added value

- Quickly identify errors

Interactive Proof

≡ Interactive Automatic Proof or Automatic Interactive Proof

- Remember: 1 PO == 35 €
- Improve demonstrations efficiency
  - Abstract and reuse demonstrations
  - Fine grained tactics
  - Motto: do not lose any proof work

Patchprover

User Rule

User Tactics

Rule packages

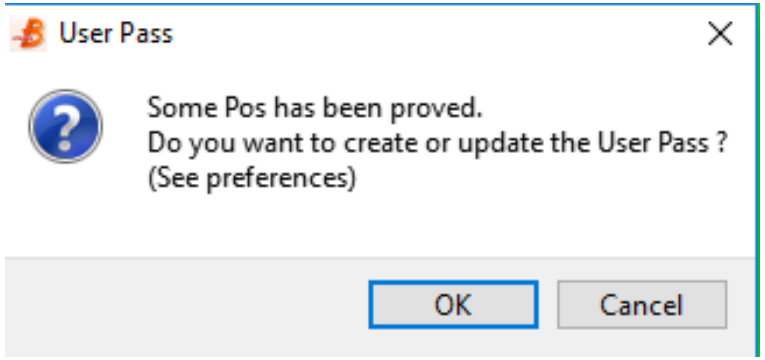
Backward

Forward

Rewriting

Operation(AssertionLemmas) & Pattern(ST_7 <: E) & dd & ah(Mhyp(ST_7: F)) &p0

Operation filter | Goal filter | Interactive commands



Do not lose any proof work

UserPass creation

preview

```
1  Operation(control) & Pattern(bool(a : {b}\/{c}\/{d}) =
   bool(bool(bool(a = b) = a or bool(a = c) = a) = a or
   bool(a = d) = a))
2      & ff(0) & dd & ah(e0 = e0$1) & ss & pp(rt.1)
```

Generated tactic from successful proof

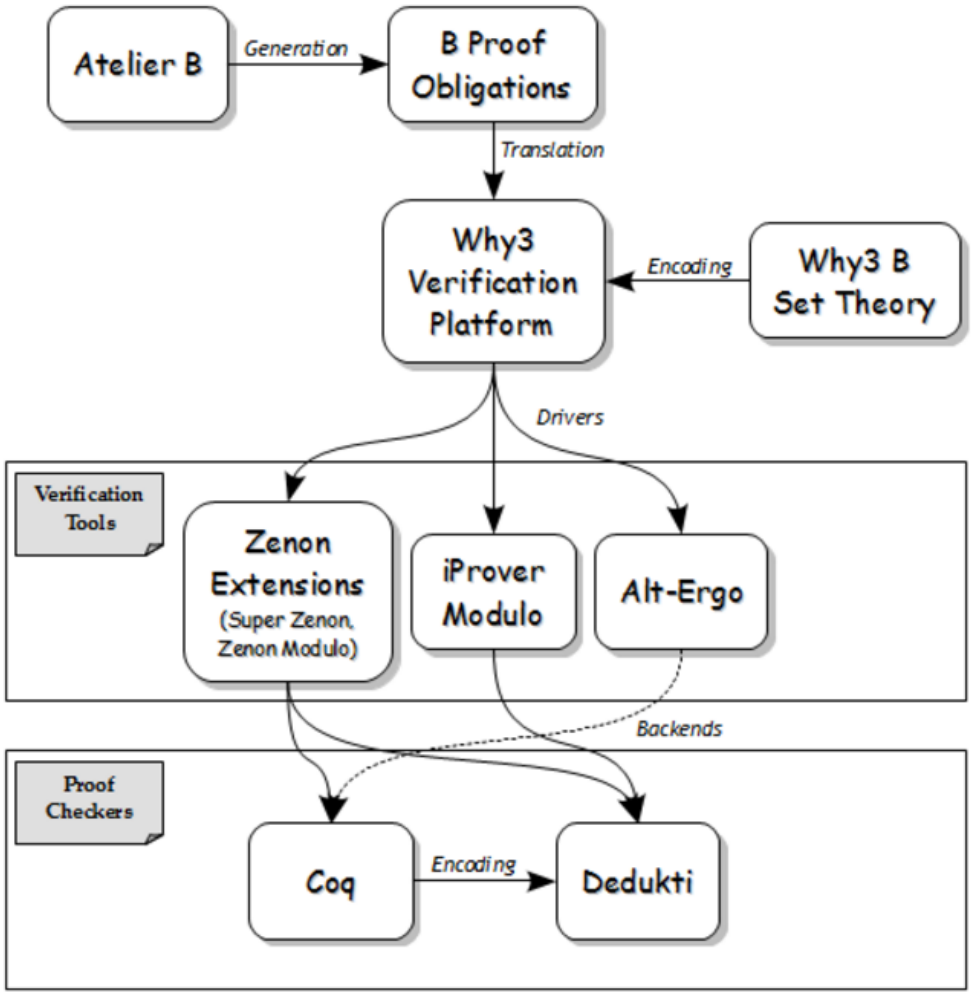
Connecting New Provers

≡ ProB

- Added to Atelier B 4.3.1 as an interactive command.
  - Syntax: prob(n) or prob(n | T).
    - T: timer in seconds,
    - n=1 selects all hypotheses that have a symbol in common with the goal
  - Add resource: ATB*PR*ProB_Path:C:\<path>\probcli.exe
- Funded by Alstom

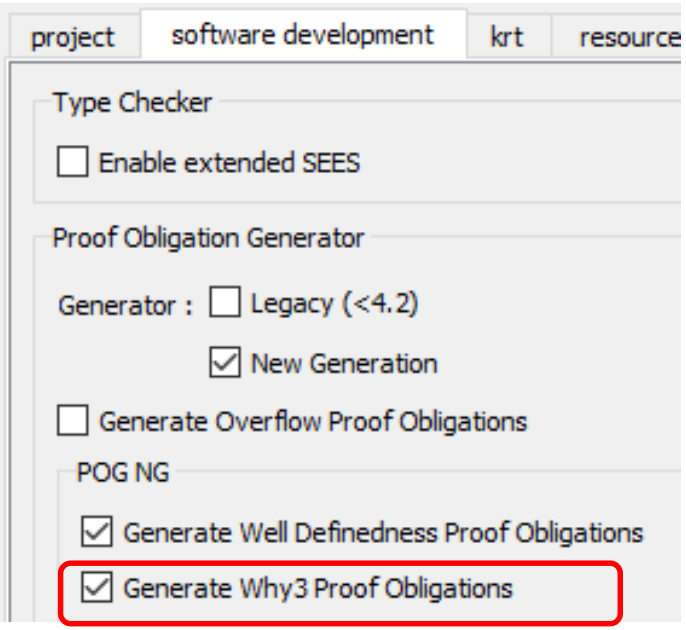
≡ Alternate provers

- Bware research project (http://bware.lri.fr/)
- Connected through Why3
- +100 k Proof obligations (obfuscated) for benchmark
- 78.% -> 99% automatic proof for one significant project



Increasing Proofs Automation Rate of Atelier-B Thanks to Alt-Ergo, S. Conchon, M. Iguernlala, RSSR 2016

Connecting New Provers




```
goal g_0 :
   forall obf_1: set int, obf_2: set int, obf_3: set int, obf_4: (set int), obf_5: int, obf_6: (set int),
   obf_7: int, obf_8: (set int), obf_9: int, obf_10: (set (int,int)), obf_11: (set (int,int)), obf_12: (
   set (int,int)), obf_13: (set (int,int)), obf_14: (set (int,int)), obf_15: (set (int,int)), obf_23: (
   set int), obf_25: (set int), obf_26: (set int), obf_16: (set int), obf_17: (set int), obf_18: (set int)
   , obf_19: (set int), obf_20: (set int), obf_21: bool.
   (define1 ) /\ (define2 obf_1 obf_2 obf_3 obf_4 obf_5 obf_6 obf_7 obf_8 obf_9 obf_10 obf_11 obf_12
   obf_13 obf_14 obf_15 ) /\ (define6 ) /\ (define7 ) /\ (define9 ) /\ (define10 ) /\ (define11 obf_23
   obf_4 obf_25 obf_26 ) /\ (define3 obf_16 obf_4 obf_17 obf_6 obf_18 obf_8 obf_19 obf_20 obf_21 ) /\ (
   lh_2_0 )
   ->
   true /\
((lh_2_1 obf_4 ) -> (exists obf_27: (set int).(mem obf_27 (power obf_4))))
```

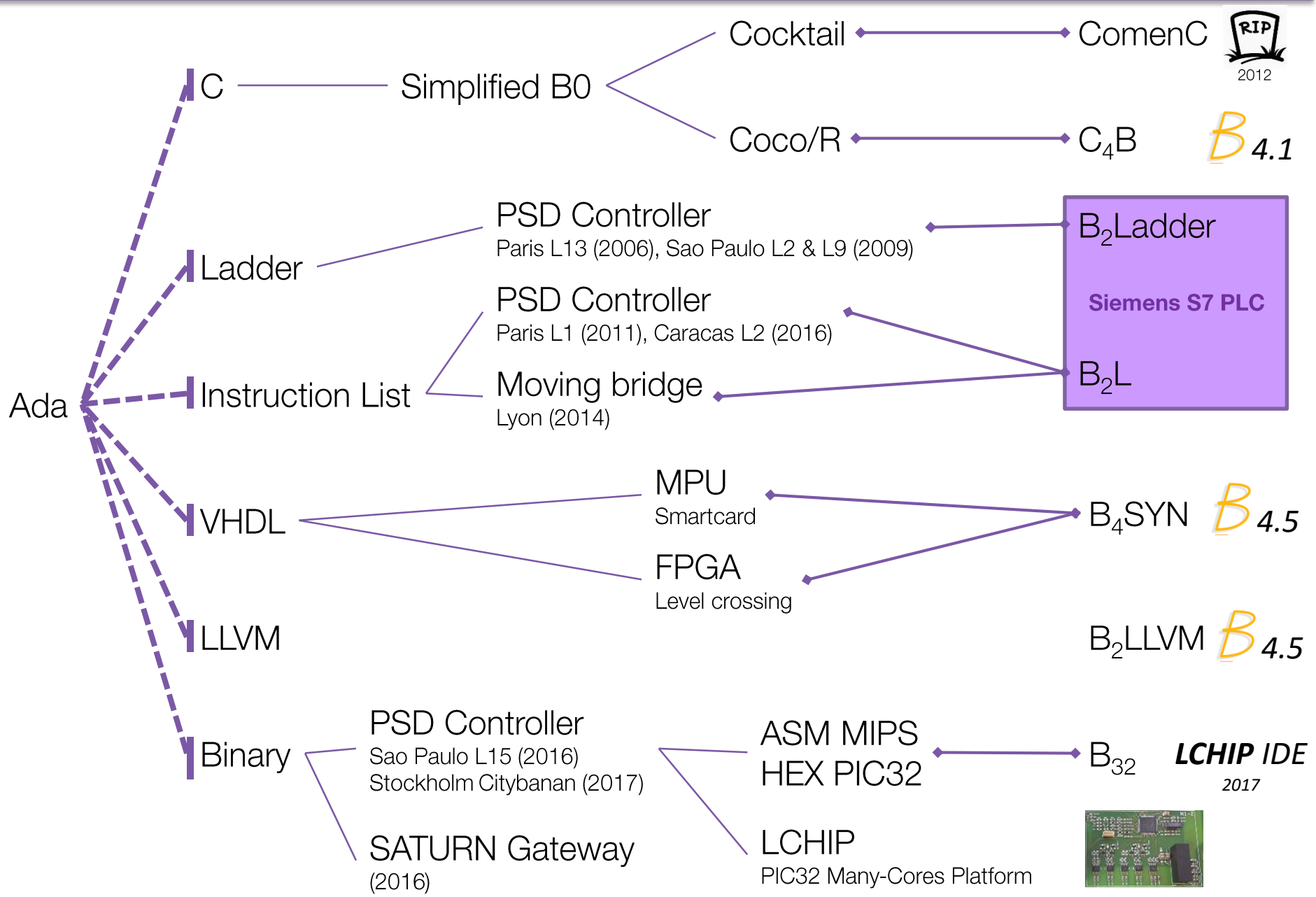
Example of Why3 fragment

≡ Alternate provers (bis)

- LCHIP research project (to start in Q4 2016)
- Connected through Why3, still
- Alt-Ergo integrated to Atelier B, Cubicle model-checker for evaluation

Generating

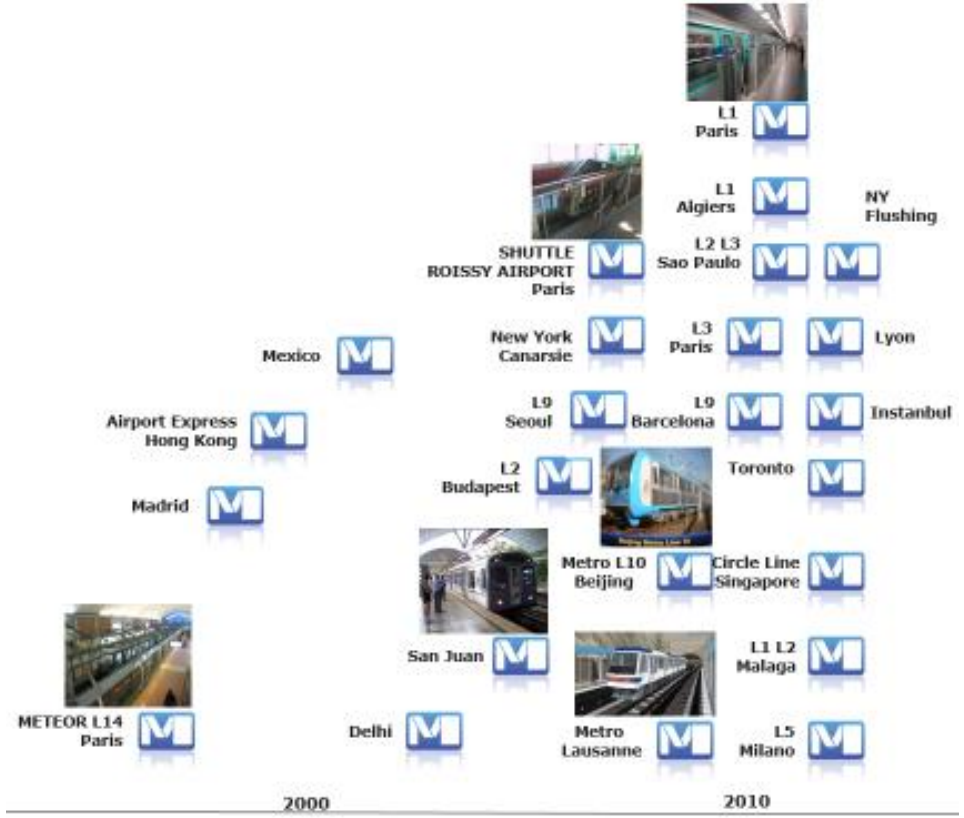
Generating Code



Ada

≡ Long sequence of implementations

- Paris L14 (1998) : 80 kloc
- Paris Roissy Airport Shuttle (2006): 186 kloc onboard, 50 kloc trackside
- Lille (2015): 300 kloc xml compiler

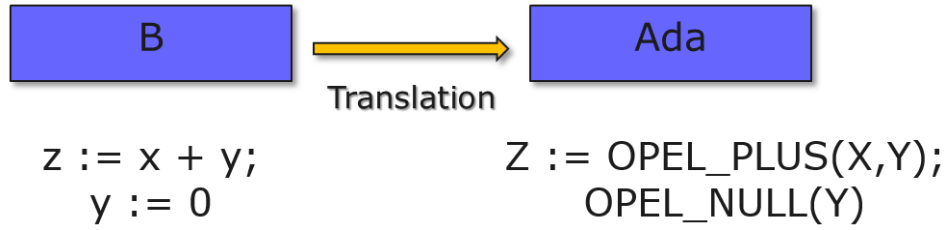


≡ Using different technologies (redundancy)

- encoding (FIDARE): 2 instances
  - inputs, outputs, variables, instructions are encoded
  - the coding key is different for each instance

- diversity (inverse mirror): 2 instances
  - one instance uses big endian model,
  - the other one executes little endian model

- specific hardware (coded secure processor): 1 instance
  - Variables have two fields: value and code. Instructions (OPELS) modified both accordingly. Mismatching value and code indicates a memory corruption.
  - Compensation tables contains tags for all possible path (masks). An unexpected value indicates a program counter corruption



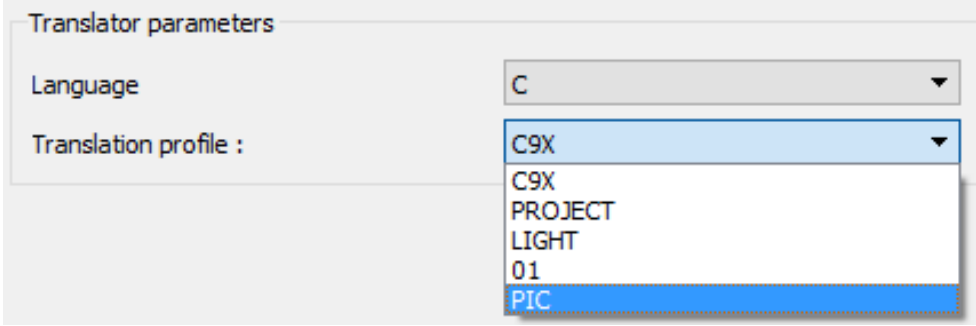
C

≡ Privileged code generation path

- Code generator that produces:
  - Readable code
  - Compliant with (railways) standards: limited use of pointers
  - Restrictions on accepted B0 (no multi instance)

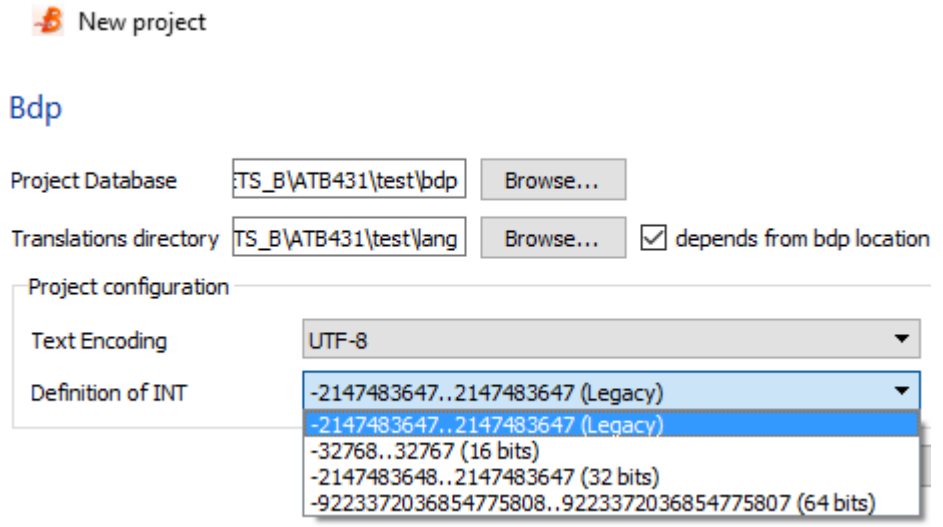
≡ Modes

- Code generation for a:
  - Machine: skeleton for the component
  - Implementation: code for the component
  - Project: makefile and code for the components
- Profiles:
  - Code generation is highly dependant on the target
  - Predefined translation profiles allow adapt generated code



≡ INT

- INT can be redefined at project level to adapt to algorithm precision



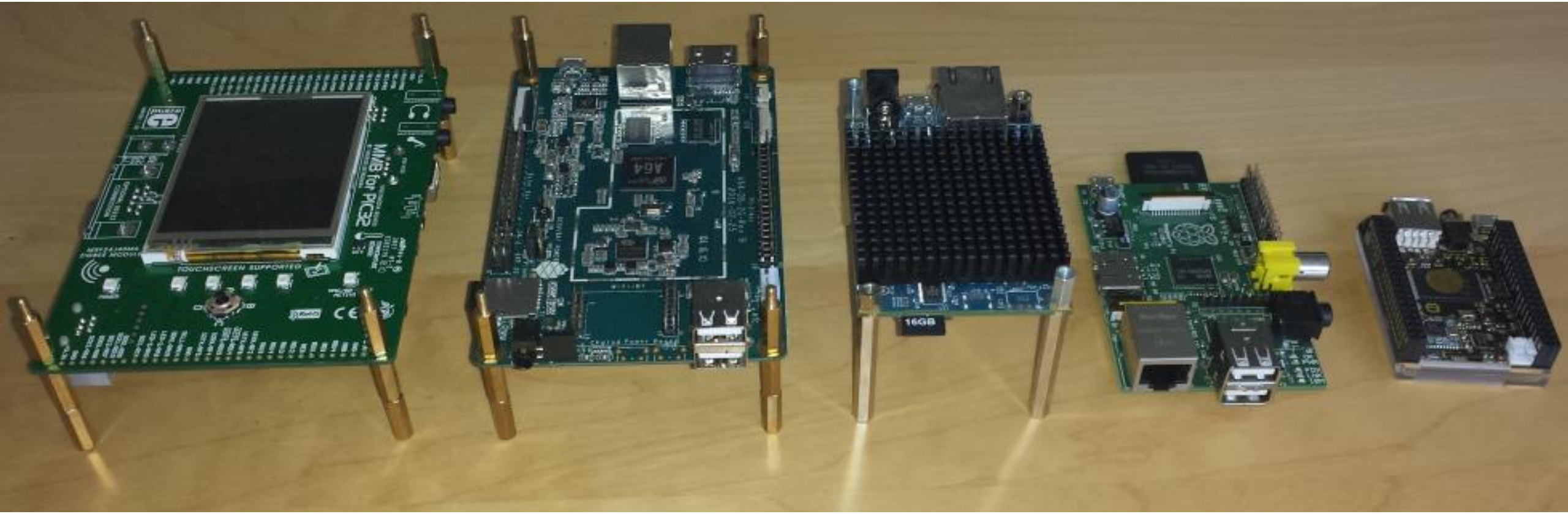
C

≡ Applications

- C code embedded at Sao Paulo L15 and Stockholm Citybanan

≡ Embedded systems

- Aimed at low cost hardware (from 6€ to 100€)
- Resources available in the future at https://github.com/TProver/TRBM: Training Resources for the B Method



MicroChip PIC32 | Pine 64 | Parallella | Raspeberry Pi | CHIP

Ladder

≡ Programming PLCs

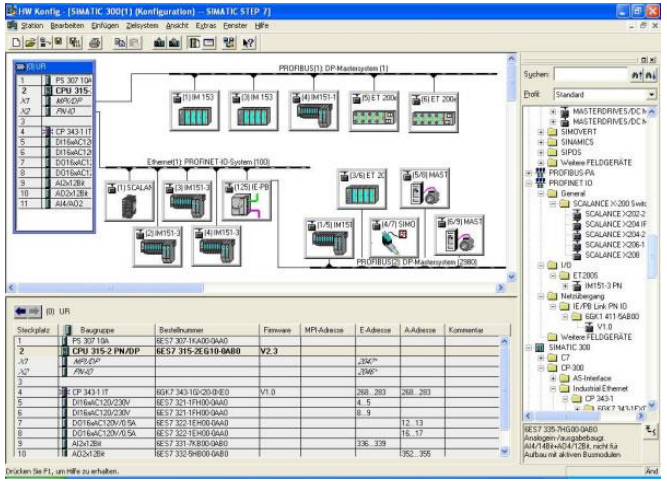
- One language (out of 5) for programming PLCs (IEC61131-3)
- Very Similar to electric circuits

```
1. ----[ ]---------|--[ ]--|------( )--
          X        |   Y   |        S
                   |       |
                   |--[ ]--|
                       Z
```

The above realises the function: S = X AND (Y OR Z)

≡ Mandatory IDE

- Step7 IDE required to program Siemens-S7 to keep the SIL3 certificate
- Ladder code generator is not connected to Step7
- Ladder programs have to be manually entered into Step7 with the GUI
- Cross-read required to check identity between programs



- Solution abandonned
- Translator not published because too specific

Instruction List

≡ Programming PLCs

- One language (out of 5) for programming PLCs (IEC61131-3)
- Kind of assembly language for PLCs

≡ Mandatory IDE

- Step7 IDE accept Instruction List copy/paste actions
- Simpler, safer, no further cross-verification

- Translator not published because too specific

≡ Applications

- PSD controller Paris Line 1 (2015), Caracas L2 (2016)
- Bridge Crane Lyon (2013)

```
IMPLEMENTATION  T003_set_i

REFINES  T_empty_mch

LOCAL_OPERATIONS
    bb <-- ope_01(nn) =
    PRE
        bb : BOOL &
        nn : NAT
    THEN
        skip
    END

OPERATIONS
    bb <-- ope_01 (nn) =
    BEGIN
        IF nn=3
        THEN
            bb := FALSE
        END
    END

END
```

```
VAR_INPUT
      nn : INT;
END_VAR

VAR_OUTPUT
      bb : BOOL;
END_VAR

VAR_TEMP
END_VAR

BEGIN

NETWORK
TITLE = RESEAU_1
      U(;
      L       #nn;
      L       3;
      ==I     ;
      );
      NOT     ;
      SAVE    ;
      BEB     ;

NETWORK
TITLE = RESEAU_2
      U       "F_GLOBDB".VKE0;
      =       #bb;

END_FUNCTION
```

Instruction List

Bridge cranes (SNCF Lyon)

- Safe movements over catenary lines

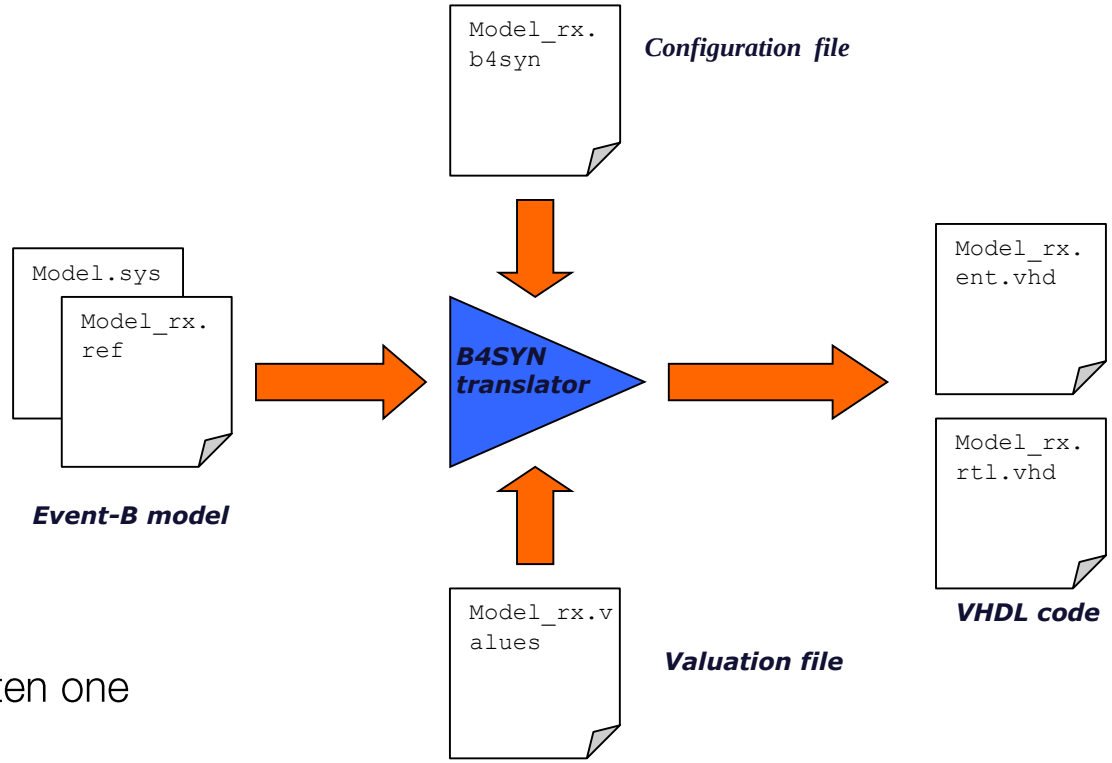
Ground PLC

Crane PLC

VHDL

≡ Objectives

- Code generator producing VHDL from proven Event-B models
- Seamless integration to Atelier B



≡ Development and fine-tuning

- Co-developed with STMicroelectronics Rousset (Smartcard division)
- Event-B model describes synchronous and asynchronous behaviour
- Code generator produces VHDL with a different structure from hand written one
- Applied to smartcard EAL6+ certified product (Memory Protection Unit)
  - Similar size (number of gates), less components, better debug vectors
  - Publication on this subject: *A proved "correct by construction" realistic digital circuit* [1]

≡ Exploitation (B_4SYN)

- Free extension to Atelier B 4.4 (ETA Q4 2016)
- Level crossing controller being developed for French Railways (FPGA, SIL4

B4SYN
VHDL CODE GENERATOR

[1] http://www.methode-b.com/2009/10/09/workshop-riab/

VHDL

≡ Event-B model

Constants

Variables

Events

≡ Parameters

Constants

Variables
- **reset**
- **clock**
- **inputs**
- **combinatorial outputs**
- **synchronous outputs**

Events
- **reset**
- **phi (input acquisition)**
- **psi (combinatorial propagation)**
- **update registers and synchronous outputs**

≡ Event-B model

Constants
- **bit level operation**

Variables
- reset
- clock
- inputs
- combinatorial outputs
- synchronous outputs

Events
- reset
- phi (input acquisition)
- psi (combinatorial propagation)
- update registers and synchronous outputs

- ☐ Prove the non-divergence of new events (VARIANT clause)
- ☐ Prove coverage
- ☐ Prove exclusivity

B4IDE v0.8.7b

Fichier | Compiler | Outils

File Edit Search Time Markers View Help

From: 0 sec To: 145 ns | Marker: -- | Cursor: 65700 ps

SST

- top
  - s0_testbench
    - uut
    - input

Signals

b1_in
b2_in
b3_t
b2_t
b1_t
c1
reset
phi1
b1
b2

Filter:

Append | Insert | Replace

Signals

Time
c1
input[2:0]
grd_phi
grd_synchro
c1_in
b1_in
b2_in
b3_t
b2_t
b1_t
reset
phi1
b1
b2

Waves

10 ns 20 ns 30 ns 40 ns 50 ns 60 ns

U
000 010 001 011 111

Signature composant

S0

PHI1
RESET
B1
B2
C1

LLVM

≡ Objectives

- Code generator producing LLVM Internal Representation (IR) from proven B models
- LLVM IR to be translated in any language supported by LLVM (JS, Java, C, Ada, etc.)
- Seamless integration to Atelier B

≡ Development and fine-tuning

- Co-developed with UFRN (Brazil)
- B model describes software functions (no interrupt, no parallel execution)
- Code generator supports C_4B B language subset
- Proof of concept demonstrated on several existing SIL4 software

≡ Exploitation (B_2LLVM)

- Free extension to Atelier B 4.4 (ETA Q4 2016)

LLVM-based code generation for B, D. Deharbe, R. Bonichon, T. Lecomte, SBMF 2015

LLVM

```
result <-- pow ( numberA , numberB ) =
PRE
    numberA : INT & numberB : INT & numberB : 0 .. MAXINT
THEN
    result := numberA ** numberB
END
```

B Specification

```
result <-- pow ( numberA , numberB ) =
BEGIN
  IF numberB >= 0 THEN
      IF numberB = 0 THEN
          result := 1
      ELSIF numberB = 1 THEN
          result := numberA
      ELSE
          VAR ii , exp IN
              ii := 1 ;
              exp := numberA ;
              WHILE ii < numberB  DO
                  exp := exp * numberA ;
                  ii := ii + 1
              INVARIANT
                  ii : INT & ii <= numberB & exp <= numberA ** numberB
              VARIANT
                  numberB - ii
              END ;
              result := exp
          END
      END
  END
END ;
```

B Implementation

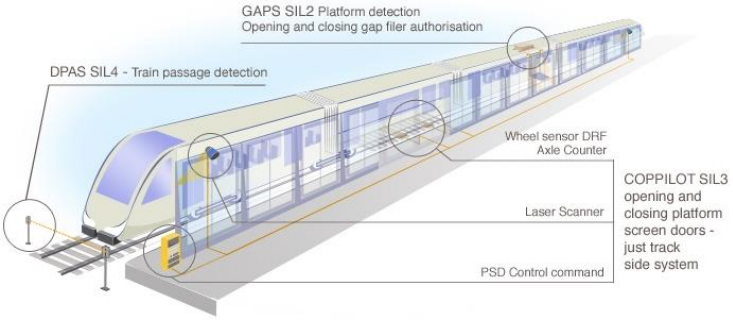
```
define void @Calculator$pow(i32 %numberA, i32 %numberB, i32* %result) {
entry:
  %ii = alloca i32
  %exp = alloca i32
  %0 = icmp sge i32 %numberB, 0
  br i1 %0, label %label0, label %exit
label0:
  %1 = icmp eq i32 %numberB, 0
  br i1 %1, label %label1, label %label2
label1:
  store i32 1, i32* %result
  br label %exit
label2:
  %2 = icmp eq i32 %numberB, 1
  br i1 %2, label %label3, label %label4
label3:
  store i32 %numberA, i32* %result
  br label %exit
label4:
  store i32 1, i32* %ii
  store i32 %numberA, i32* %exp
  br label %label6
label6:
  %3 = load i32* %ii
  %4 = icmp slt i32 %3, %numberB
  br i1 %4, label %label7, label %label5
label7:
  %5 = load i32* %exp
  %6 = mul i32 %5, %numberA
  store i32 %6, i32* %exp
  %7 = load i32* %ii
  %8 = add i32 %7, 1
  store i32 %8, i32* %ii
  br label %label6
label5:
  %9 = load i32* %exp
  store i32 %9, i32* %result
  br label %exit
exit:
  ret void
}
```

LLVM IR code

Binary

≡ C32

- Generate Hex file from B0 implementation
- Application: PSD Sao Paulo L15 (2015), Stockholm Citybanan 2016



Modèle

```
IMPLEMENTATION
  machine_i
REFINES
  machine
OPERATIONS
  operation =
  BEGIN
    IF TRUE=TRUE THEN
      skip
    ELSIF TRUE=TRUE THEN
      skip
    ELSE
      skip
    END
  END
END
```

Mapping

```
= 0x00018000    machine__initialisation
= 0x00018028    operation
```

ASM Mips32

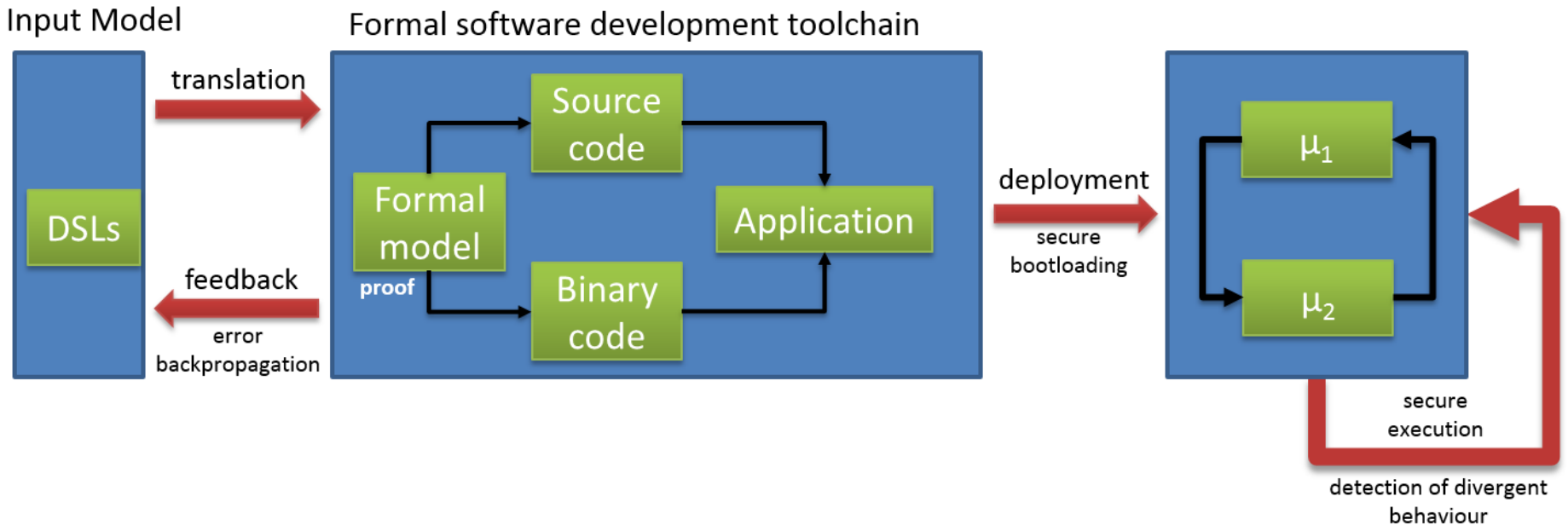
```
machine__initialisation:
    addiu   $sp,$sp,-24
    sw  $ra,20($sp)
    sw  $fp,16($sp)
    movz    $fp,$sp,$zero
    movz    $sp,$fp,$zero
    lw  $fp,16($sp)
    lw  $ra,20($sp)
    addiu   $sp,$sp,24
    jr  $ra
    nop
operation:
    addiu   $sp,$sp,-24
    sw  $ra,20($sp)
    sw  $fp,16($sp)
    movz    $fp,$sp,$zero
.if_begin_1:
    lui $t2,%hi(1)
    ori $t2,$t2,%lo(1)
    lui $t3,%hi(1)
    ori $t3,$t3,%lo(1)
    xor $t1,$t2,$t3
    sltiu   $t1,$t1,1
    beq $t1,$zero,%gp_rel_sub_4_div_4(.if_else_1)
    nop
.if_then_1:
    b   %gp_rel_sub_4_div_4(.if_end_1)
    nop
.if_else_1:
.if_begin_2:
    lui $t2,%hi(1)
    ori $t2,$t2,%lo(1)
    lui $t3,%hi(1)
    ori $t3,$t3,%lo(1)
    xor $t1,$t2,$t3
    sltiu   $t1,$t1,1
    beq $t1,$zero,%gp_rel_sub_4_div_4(.if_else_2)
    nop
.if_then_2:
    b   %gp_rel_sub_4_div_4(.if_end_2)
    nop
.if_else_2:
.if_end_2:
.if_end_1:
    movz    $sp,$fp,$zero
    lw  $fp,16($sp)
    lw  $ra,20($sp)
    addiu   $sp,$sp,24
    jr  $ra
    nop
```

Hex

```
:0200000040001f9
:04800000e8ffbd27b1
:0480040001400bfaff6
:0480080001000beaff7
:04800c000af0a003d3
:0480100000ae8c003b7
:0480140001000be8f0b
:0480180001400bf8f02
:04801c001800bd2764
:0480200008800e00371
:0480240000000000058
:04802800e8ffbd2789
:04802c001400bfafce
:0480300001000beafcf
:0480340000af0a003ab
:0480380000000a3cfe
:04803c0001004a35c0
:0480400000000b3cf5
:0480440001006b3597
:0480480026484b017a
:04804c000100292dd9
:0480500003002011f8
:0480540000000000028
:0480580000b00001009
:04805c000000000020
:0480600000000a3cd6
:0480640001004a3598
:0480680000000b3ccd
:04806c0001006b356f
:0480700026484b0152
:0480740000100292db1
:0480780003002011d0
:04807c0000000000000
:0480800001000010eb
:04808400000000000f8
:0480880000ae8c0033f
:04808c001000be8f93
:0480900001400bf8f8a
:0480940001800bd27ec
:0480980008800e003f9
:04809c0000000000e0
:00000001ff
```

Future

Low Cost High Integrity Platform

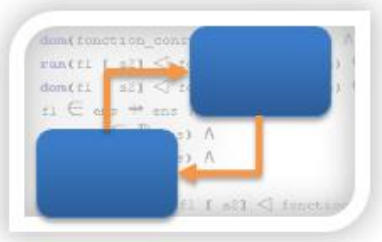


≡ Automatic generation of proven code

- Formal part hidden behind the curtain
- Support for bounded algorithm complexity
- Domain Specific Languages: grafcet, relays drawings, etc (IEC61131-3) to keep development perturbation low

- To ease SIL4 certificate achievement:
  - Developer focused on functional aspects
  - Even non high grade engineers could participate

Funded by bpifrance

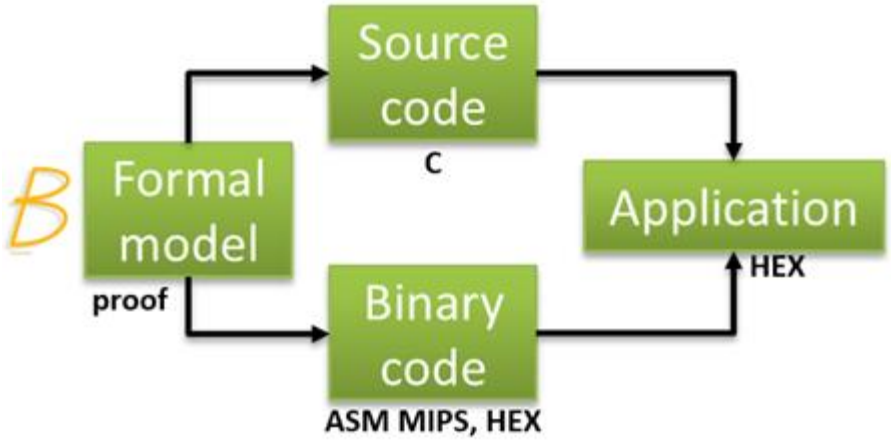


Low Cost High Integrity Platform

≡ 1 B model (application)

- Data
  - Integer arithmetic
  - Boolean equations
  - State machine
  - Variables typing and properties
- Cyclic software, no interrupt modifying state variables

≡ 1 C code generator + commercial compiler => Hex



≡ 1 ASM MIPS / Hex code generator

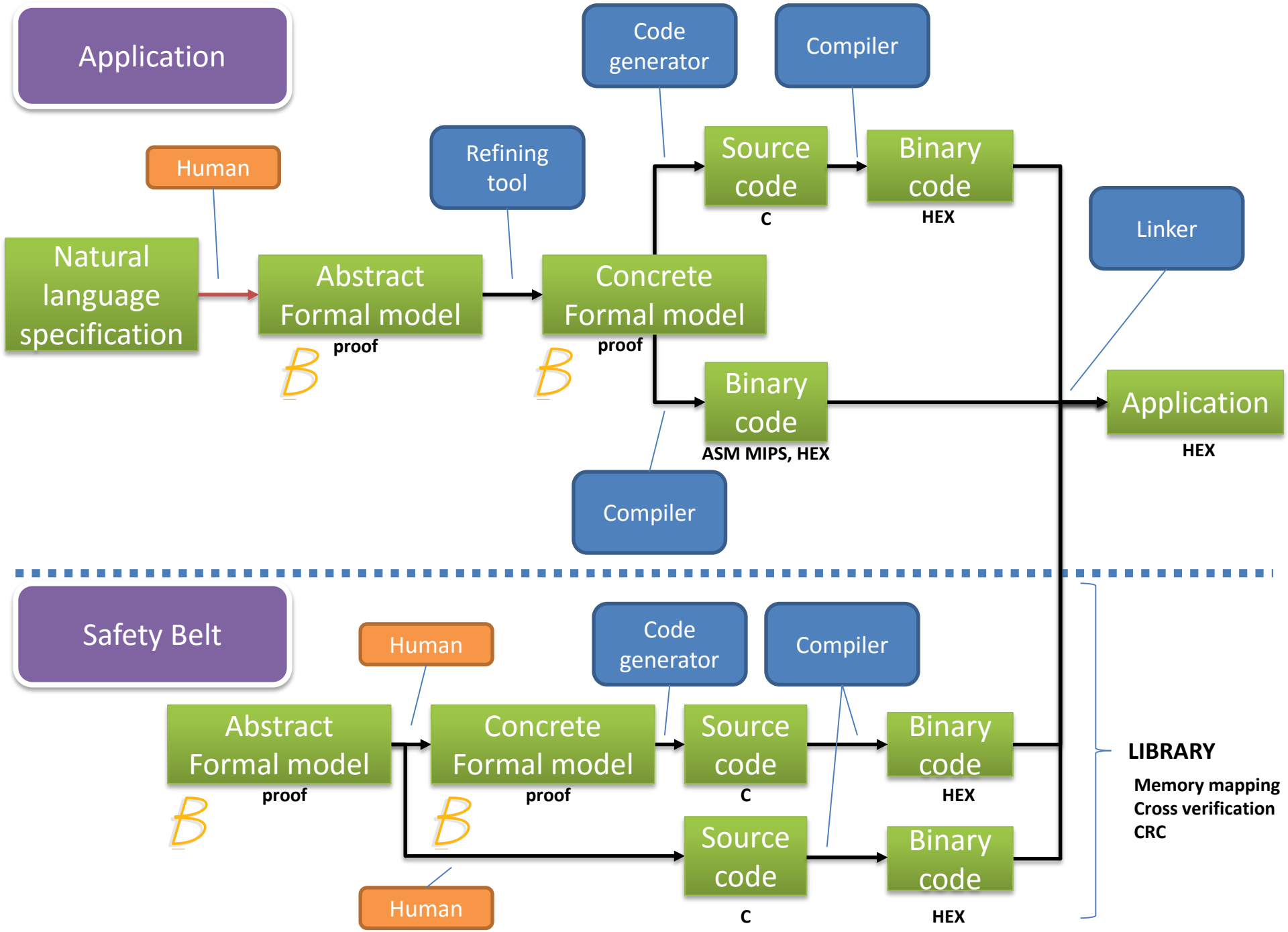
≡ Application proved vs requirements

- Implementation doesn't contradict specification
- A minima, no programming error:
  - overflow,
  - div 0,
  - tables

≡ Low level software ensuring safety developped in B

- Once for all
- Safety features out of reach of the developers
- No requirement to certify the tools

Funded by bpifrance


Application
Code generator
Compiler
Human
Refining tool
Source code
C
Binary code
HEX
Linker
Natural language specification
Abstract Formal model
proof
Concrete Formal model
proof
Binary code
ASM MIPS, HEX
Application
HEX
Compiler
Safety Belt
Human
Code generator
Compiler
Abstract Formal model
proof
Concrete Formal model
proof
Source code
C
Binary code
HEX
Source code
C
Binary code
HEX
Human
LIBRARY
Memory mapping
Cross verification
CRC

Low Cost High Integrity Platform

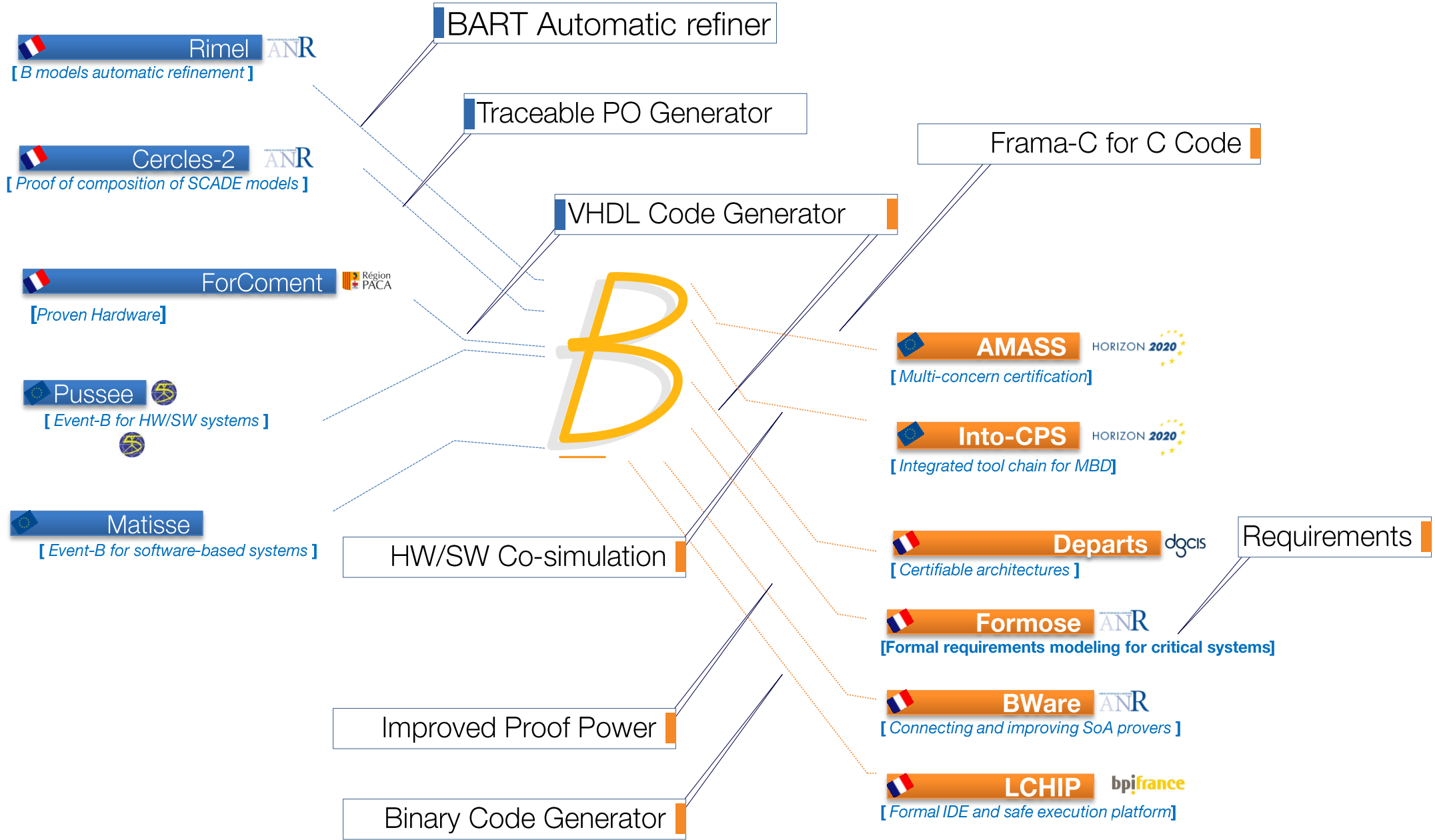
≡ Starter-kit @ Q3 2017

- Mother board (I/Os, maintenance processor, network sockets)
- Daughter card, embedding the double-processor
- IDE v1, based on Atelier B

≡ IDE v2 @ Q2 2018

≡ IDE v3 @ Q2 2019

R&D Projects


Rimel
ANR
[B models automatic refinement]
Cercles-2
ANR
[Proof of composition of SCADE models]
ForComent
Région PACA
[Proven Hardware]
Pussee
[Event-B for HW/SW systems]
Matisse
[Event-B for software-based systems]
BART Automatic refiner
Traceable PO Generator
VHDL Code Generator
HW/SW Co-simulation
Improved Proof Power
Binary Code Generator
Frama-C for C Code
AMASS
HORIZON 2020
[Multi-concern certification]
Into-CPS
HORIZON 2020
[Integrated tool chain for MBD]
Departs
dgcis
[Certifiable architectures]
Requirements
Formose
ANR
[Formal requirements modeling for critical systems]
BWare
ANR
[Connecting and improving SoA provers]
LCHIP
bpifrance
[Formal IDE and safe execution platform]

Conclusion & Perspectives

- Atelier B still alive

  +10B passengers transported by « B inside » metros - ***Lille 2015, Paris L4 2018***
  +100M passengers going through doors operated by B software
  Sao Paulo L15 2016, Stockholm Citybanan 2017

- Is not only a tool for piloting metros
  - Opening doors safely — *Formal Methods in Safety-Critical Railway Systems,* SBMF 2007
  - Validating parameters — *Formal Data Validation in the Railways*, SSS 2016
  - Contributing to the certification of microcircuits — *A « Correct by Construction » Realistic Digital Circuit*, RIAB/FM 2009

- New usage
  - Integrated to PLC — *Double Processor and Formal Proof for SIL4 Automation,* λμ 2016
  - Applications to (nuclear) energy automation — *Formal Virtual Modelling and Data Verification for Supervision Systems,* FM 2015

- Forthcoming evolutions
  - Linked with industrial needs
  - Funded by industrial & collaborative R&D projects

formal methods are only a tiny part of the whole picture

- Let's meet for ABZ 2026 !

Thank you for your attention

Thierry Lecomte
ABZ 2016
Linz

CLEARSY
SYSTEM ENGINEERING