Formal Techniques For Safer Signalling Systems

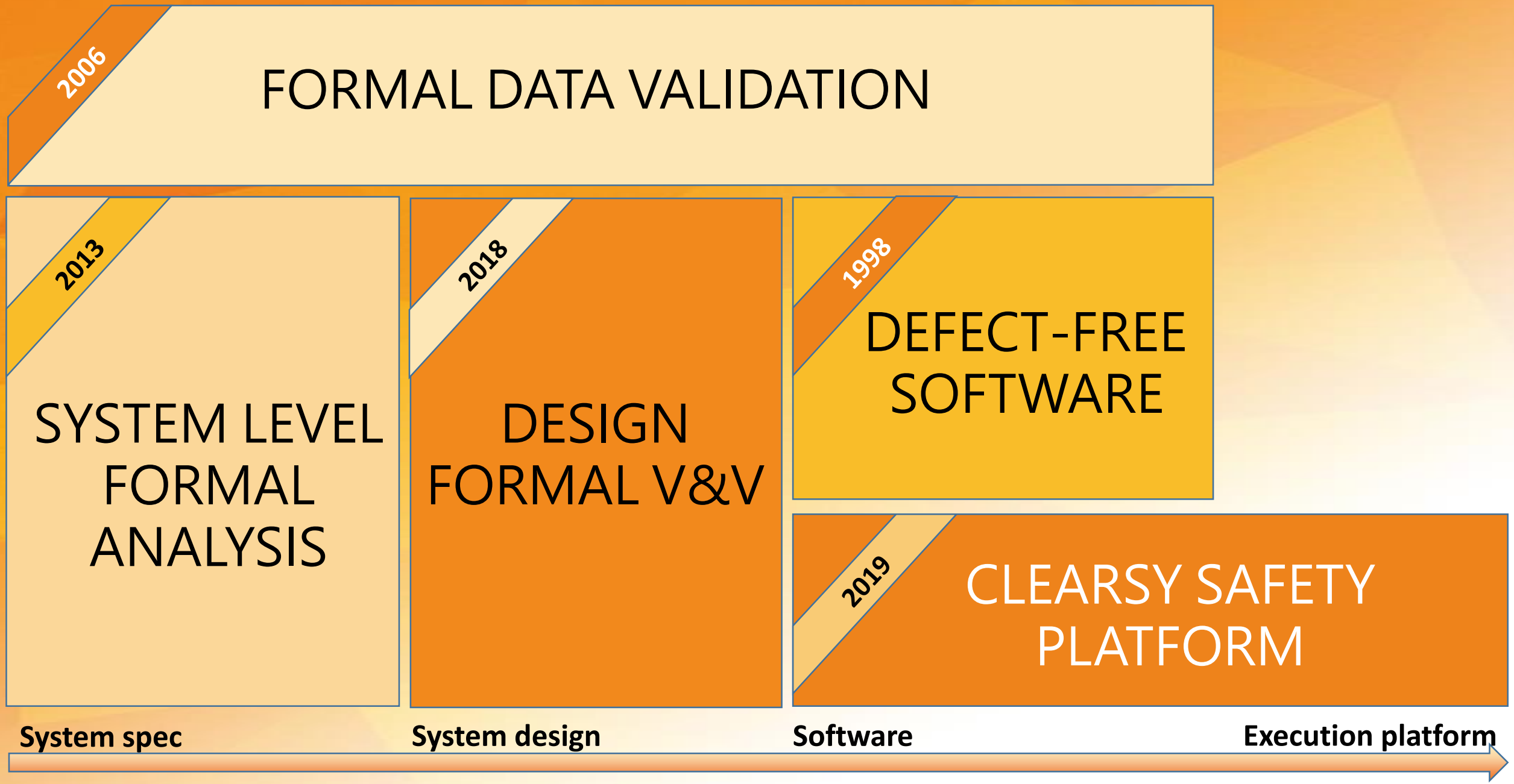
DisCoRail 2019 - International Workshop on Distributed Computing in Future Railway Systems

Lyngby, Denmark, June 17th 2019

Thierry Lecomte
R&D Director, CLEARSY

thierry.lecomte@clearsy.com

CLEARSY
SYSTEMS ENGINEERING


2006
FORMAL DATA VALIDATION
2013
SYSTEM LEVEL FORMAL ANALYSIS
2018
DESIGN FORMAL V&V
1998
DEFECT-FREE SOFTWARE
2019
CLEARSY SAFETY PLATFORM
System spec
System design
Software
Execution platform

1998

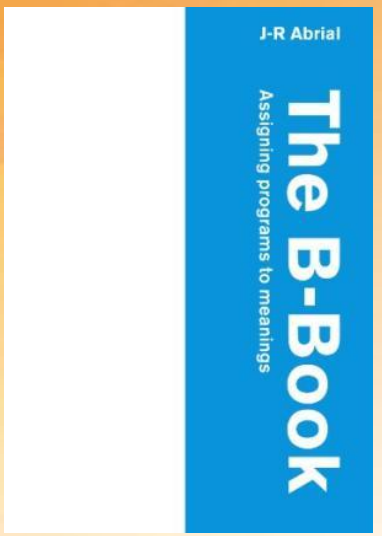
DEFECT-FREE SOFTWARE

B METHOD

http://www.methode-b.com/

Assigning programs to meanings

Programs proved to comply with specification



FORMAL SOFTWARE DEVELOPMENT

« Only inactive sequences can be added to the active sequences execution queue. »

```
activation_sequence = /* Activation d'une séquence non active */
PRE ¬(sequences = sequences_actives) THEN
    ANY sequ WHERE
        sequ ∈ sequences - sequences_actives
    THEN
        sequences_actives := sequences_actives ∪ {sequ}
    END
END;
```

```
activation_sequence = /* Activation d'une séquence non active */
VAR sequ IN
    sequ <-- indexSequenceInactive;
    activeSequence(sequ)
END;
```

```
void M0__activation_sequence(void)
{
    CTX__SEQUENCES sequ;

    sequence_manager__indexSequenceInactive(&sequ);
    sequence_manager__activeSequence(sequ);
}
```

```
0x01F970  FFFF 8B4C 2440 89C5 8D7D 0C8B 4110 89CE
0x01F980  83C6 0C8D 1485 0000 0000 8D42 0883 F807
0x01F990  7617 F7C7 0400 0000 740F 8B41 0C8D 7D10
0x01F9A0  83C6 0489 450C 8D42 04FC 89C1 C1E9 02F3
```

Natural language requirement

Behaviour + properties

B Specification

Behaviour + properties

B Implementation

C generated code

Binary code

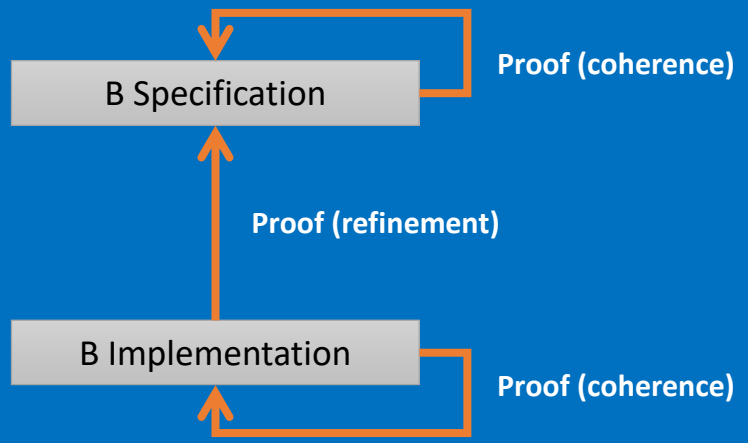
FORMAL SOFTWARE DEVELOPMENT

« Only inactive sequences can be added to the active sequences execution queue. »

Natural language requirement

```
activation_sequence = /* Activation d'une séquence non active */
PRE ¬(sequences = sequences_actives) THEN
    ANY sequ WHERE
        sequ ∈ sequences - sequences_actives
    THEN
        sequences_actives := sequences_actives ∪ {sequ}
    END
END;
```

```
activation_sequence = /* Activation d'une séquence non active */
VAR sequ IN
    sequ <-- indexSequenceInactive;
    activeSequence(sequ)
END;
```



```
void M0__activation_sequence(void)
{
    CTX__SEQUENCES sequ;

    sequence_manager__indexSequenceInactive(&sequ);
    sequence_manager__activeSequence(sequ);
}
```

C generated code

Cyclic software single-thread

```
0x01F970  FFFF 8B4C 2440 89C5 8D7D 0C8B 4110 89CE
0x01F980  83C6 0C8D 1485 0000 0000 8D42 0883 F807
0x01F990  7617 F7C7 0400 0000 740F 8B41 0C8D 7D10
0x01F9A0  83C6 0489 450C 8D42 04FC 89C1 C1E9 02F3
```

Binary code

1998

DEFECT-FREE SOFTWARE

Météor: A Successful Application of B in a Large Project
International Symposium on Formal Methods - FM'99
P. Behm, P. Benoit, A. Faivre, J.-M. Meynadier

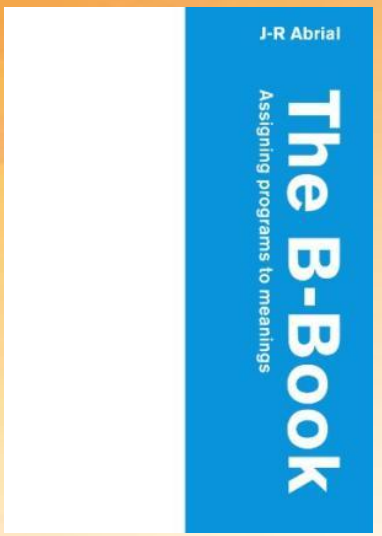
Using B as a High Level Programming Language in an Industrial Project: Roissy VAL
International Conference of B and Z Users - ZB 2005
F. Badeau, A. Amelot

Formal method and coded processor
ERTS 2006
D. Dollé

B METHOD

http://www.methode-b.com/

Assigning programs to meanings
Programs proved to comply with specification



ATELIER B

http://www.atelierb.eu/

Driverless metro Paris L14 Meteor
30% automatic metro worldwide
Maintained & developed by CLEARSY

FREE TOOL

1998

DEFECT-FREE SOFTWARE

Automatic Driving Metro subsystems

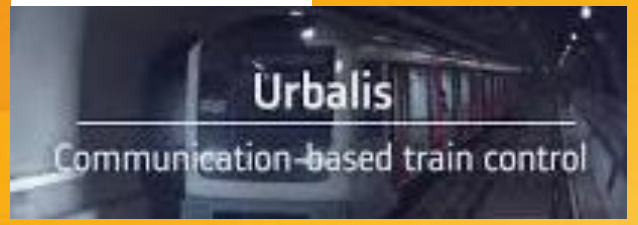
Localization: graph-based algorithms

Energy control: integer arithmetic (braking curve)

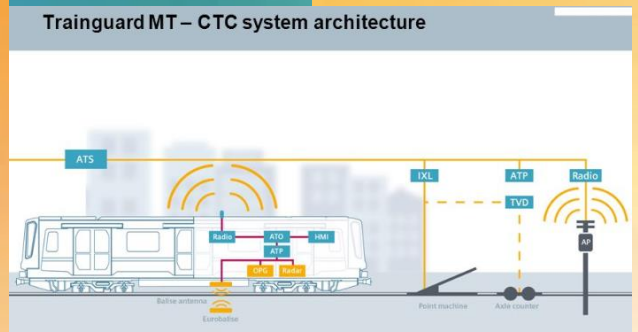
Emergency braking: Boolean predicates

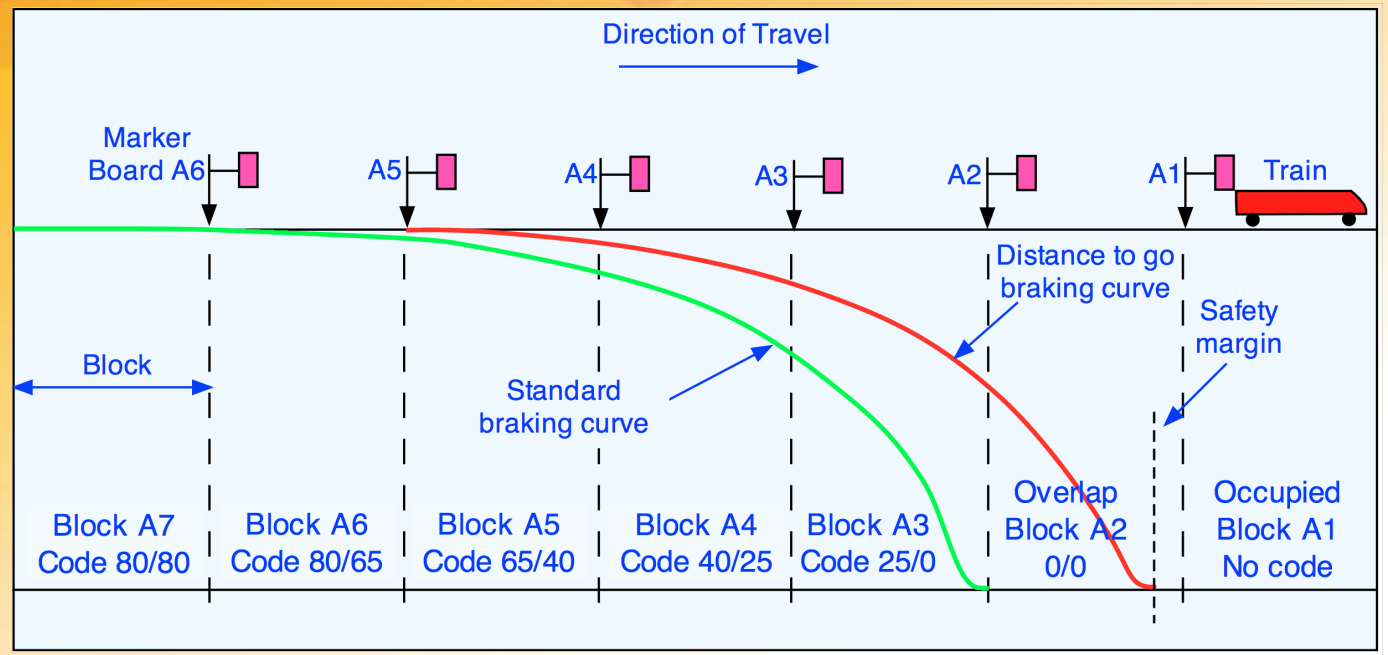
Trackside software (Interlocking)

ALSTOM



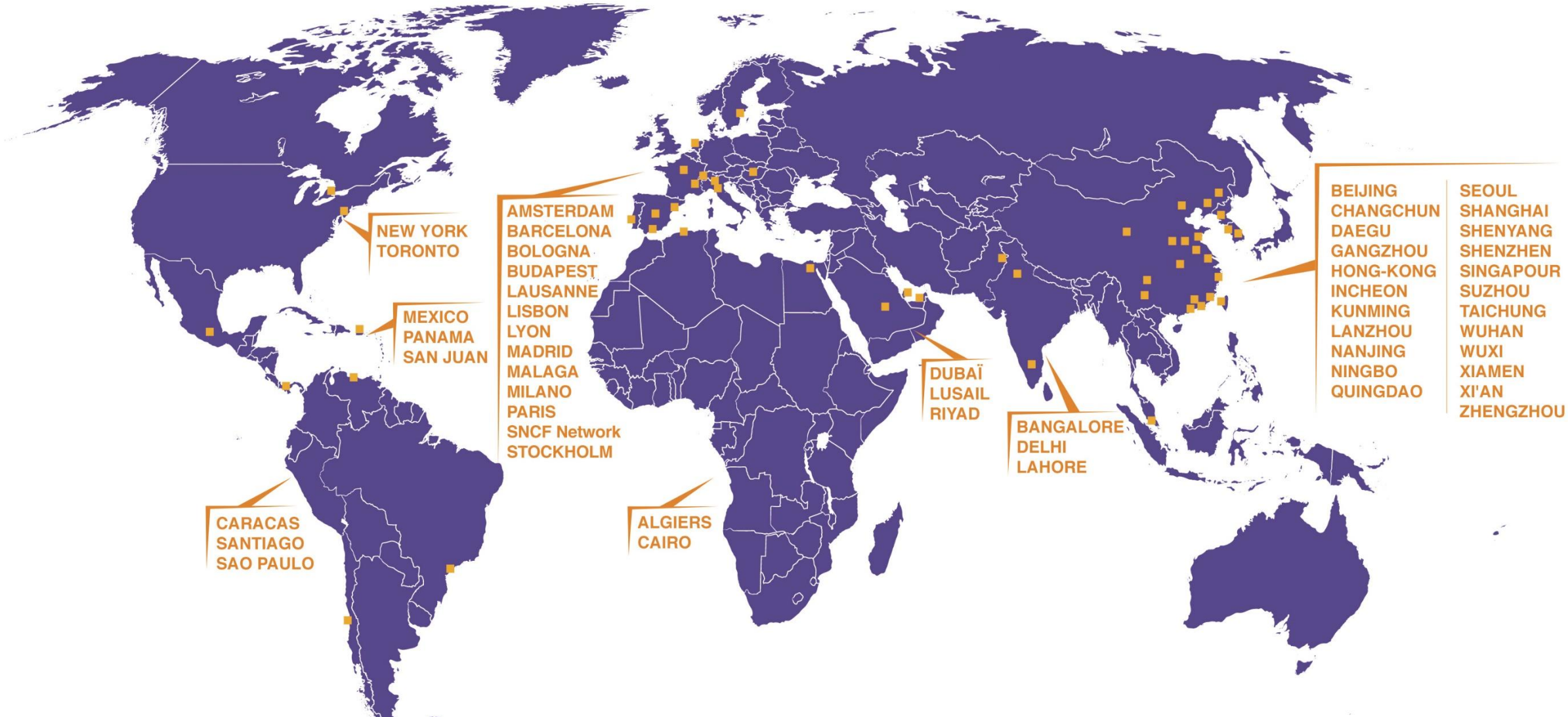
SIEMENS





ATELIER B

METROS AND TRAINS EQUIPPED WITH **B SIL4 SOFTWARE**



DISSEMINATION: 21 videos – 7 hours

Lecture 0: Marketing video

This video explains why you should follow the MOOC on B and what its expected benefits on your career are.

Level: Basic | Video duration: 02:55

Lecture 1: Course Introduction

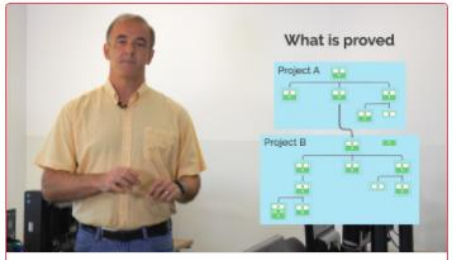
This video presents the structure of the course, provides an overview of the different kinds of formal methods and specification styles, and tells us some myths on formal methods

Level: Basic | Video duration: 08:43

Lecture 2: Overview of the B method

This video briefly introduces the tool Atelier-B, the B and Event-B languages, and some industrial references. The main concepts of B are exposed.

Level: Basic | Video duration: 15:03



Lecture 3: The concepts of B

This video presents the founding notions of B: projects, libraries, modules, components, abstract machine, refinement, implementation , and proof.

Level: Basic | Video duration: 09:29

Lecture 4 : introduction to Abstract Machines

This video introduces the notion of abstract machines, based on an example that is verified, animated and for which C source code is generated.

Level: Basic | Video duration: 16:31

MOOC
massive open online course

https://mooc.imd.ufrn.br/

UFRN
UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE

IMD
INSTITUTO METRÓPOLE DIGITAL

2006

FORMAL DATA VALIDATION

2006

FORMAL DATA VALIDATION

Formal Data Validation in the Railways
Safety-critical Systems Symposium 2016
T. Lecomte, E. Mottin

Formally Checking Large Data Sets in the Railways
ICFEM 2014
T. Lecomte, L. Burdy, M. Leuschel, F. Mejia

Large data sets automatic check

Data involved in the building / execution of safety critical systems

Properties: international standards, national regulations, manufacturer habits, customer requirements

	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

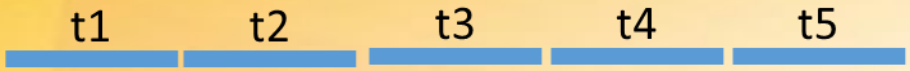
Up to 100,000+ raw data chunks

Are they
- Consistent ?
- Correct ?
- Safe ?

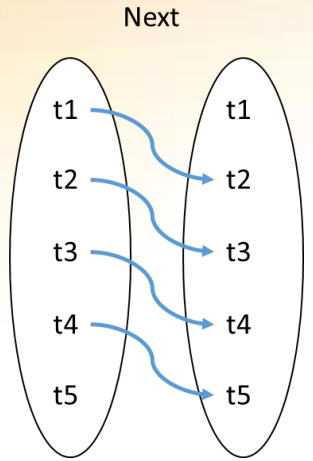
2006

FORMAL DATA VALIDATION

Modelling language based on set theory and first order predicates logic (B mathematical language)



Let the set TrackCircuit = {t1, t2, t3, t4, t5}



Let the function Next $\in$ TrackCircuit $\nrightarrow$ TrackCircuit

Example: Next(t1) = t2, Next(t2) = t3, Next(t3) = t4, Next(t4) = t5

Next = {t1 $\mapsto$ t2, t2 $\mapsto$ t3, t3 $\mapsto$ t4, t4 $\mapsto$ t5}

Let the function KpAbs : TrackCircuit $\rightarrow$ N

$\forall x.(x \in \text{TrackCircuit} \wedge x \in \text{dom(Next)} \Rightarrow \text{KpAbs (Next(x))} > \text{KpAbs(x)})$

2006

FORMAL DATA VALIDATION

Industry ready

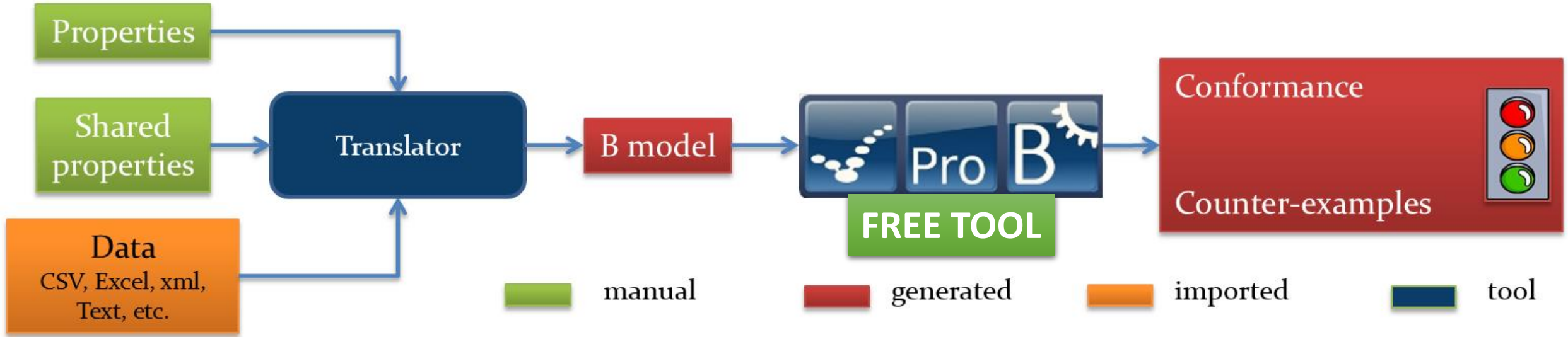
Push-button fast verification, replayable at will (hour(s), PARETO law)

ProB model checker improved over 10+ years

Most energy spent on formalizing properties

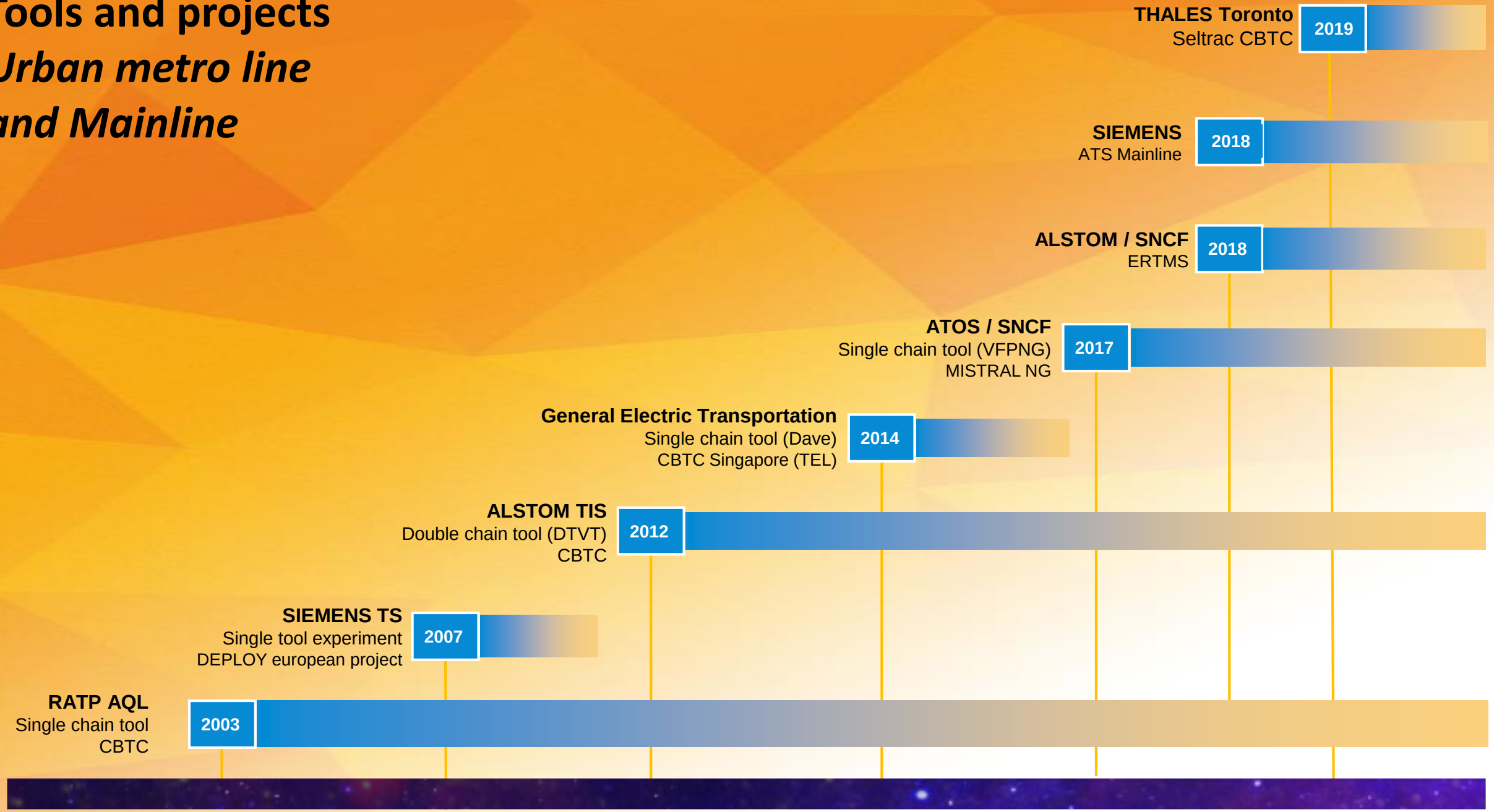
~85% reuse from project to another

Diverse tool chain if required (SIL4 – T2)



Tools and projects

Urban metro line and Mainline



2006

FORMAL DATA VALIDATION

Well-oiled process

20+ CBTC validated

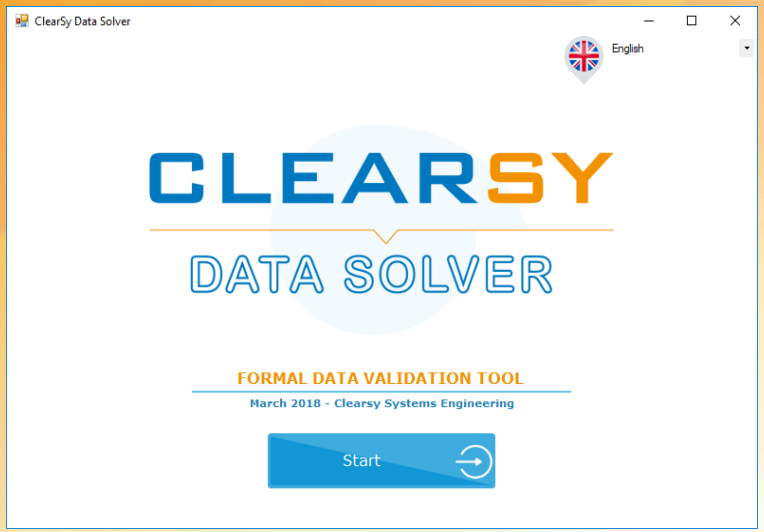
Extension to light rail and mainline signalling

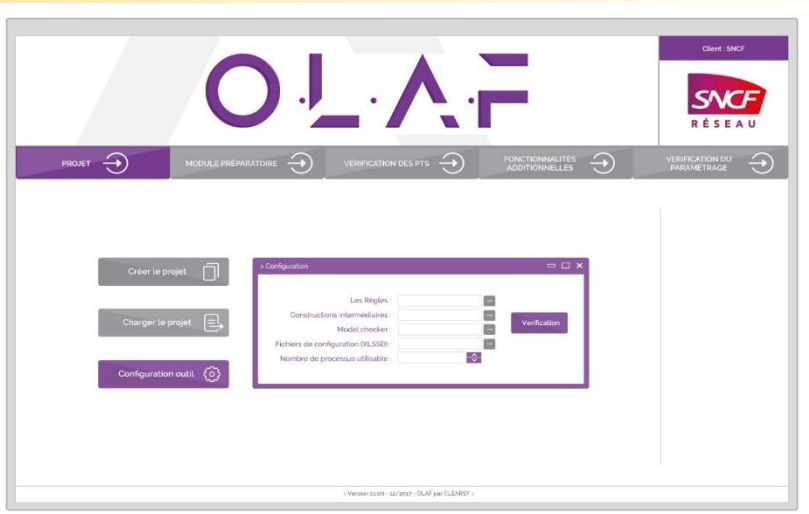
5000+ properties formally modelled

~85% reuse for a product line

Adaptation:

- **Interfaces with input files**
- **Modelling language**
- **Generation of reports**
- **Integration with in-house process**





2013

SYSTEM LEVEL FORMAL ANALYSIS

2013

SYSTEM LEVEL FORMAL ANALYSIS

Using formal proof and B method at system level for industrial projects
RSSR 2016
D. Sabatier

Safety analysis of a CBTC system: a rigorous approach with Event-B
RSSR 2017
M. Comptier, D. Déharbe, L. Mussat, P. Thibaut, D. Sabatier

Formal Proofs for the NYCT Line 7 Modernization Project
ABZ 2012
D. Sabatier, L. Burdy, J. Guéry

Proof for everyone

Develop a reasoning:

- with natural language
- based on assumptions
- anyone could understand the proof
- NEVER doubt that properties are logically deduced from these assumptions

Evidence that the system is safe:

- Not only because good reputation of the system's supplier
- Not only because independent experts say it is safe
- Using a reasoning that can be re-checked at any time

Evidence using logics validated by a formal tool (Event-B proof)

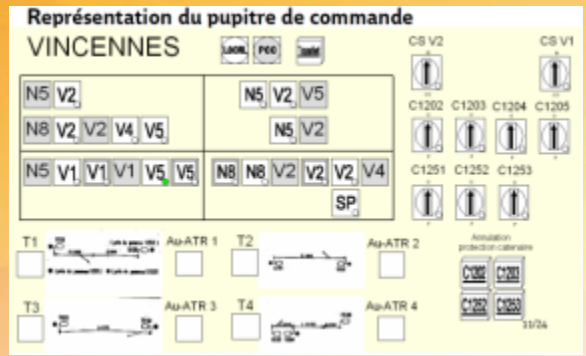
2013

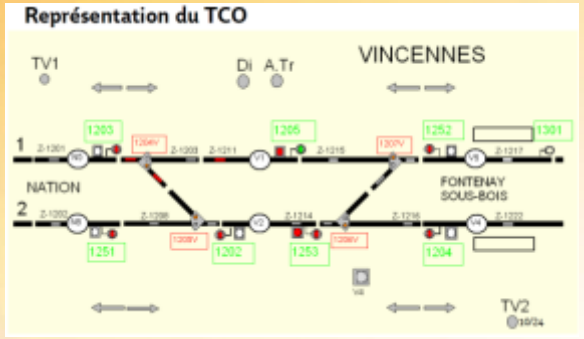
SYSTEM LEVEL FORMAL ANALYSIS

Not a “model of everything” like …

Atelier de Modélisation de l’Environnement de la Ligne A

- simulates all systems: signaling, tracks, energy control and rolling stock
- contains 2M+ variables
- requires several computers to run the whole simulation of the RER A

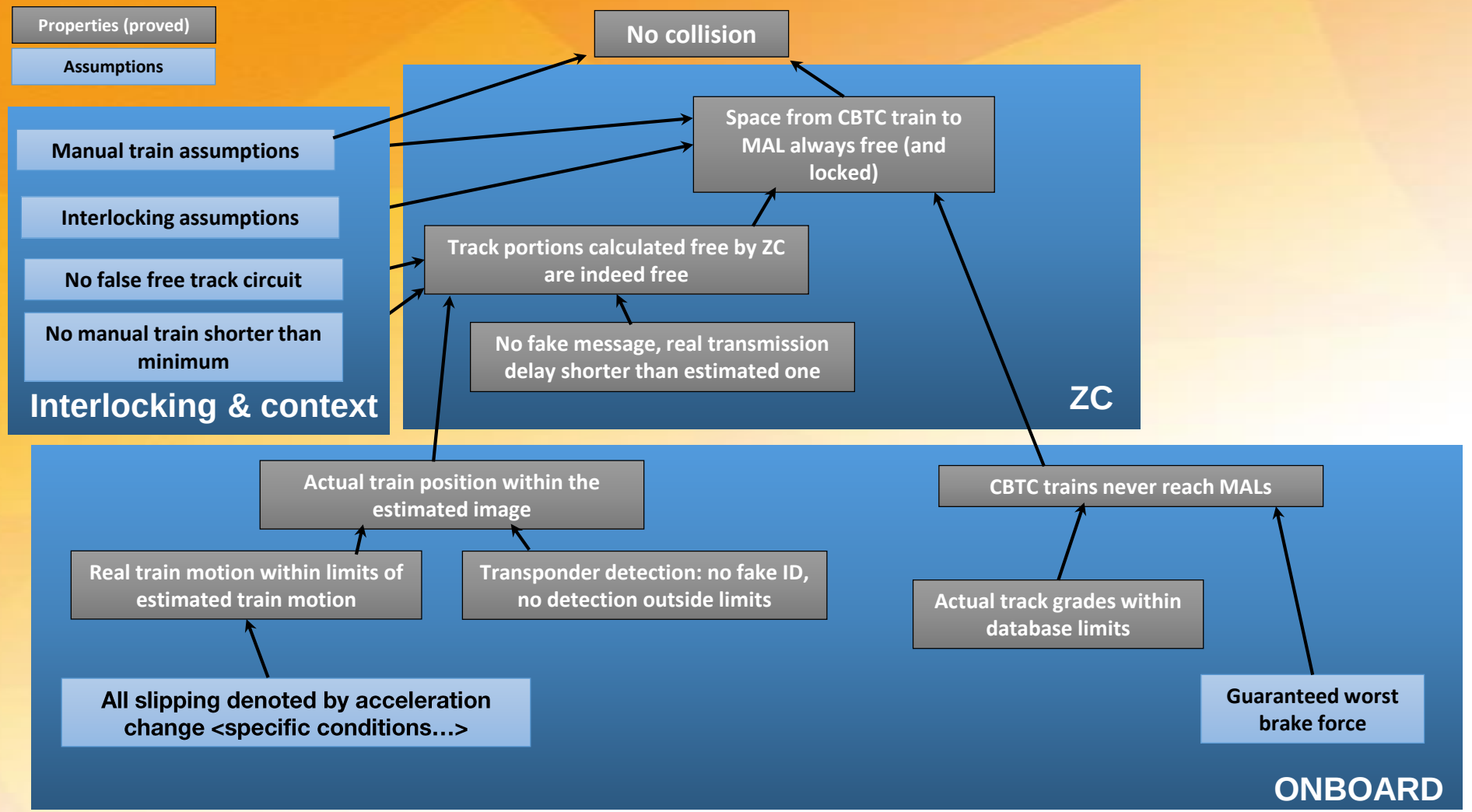




2013

SYSTEM LEVEL FORMAL ANALYSIS

Property hierarchy (example)



2013

SYSTEM LEVEL FORMAL ANALYSIS

NYCT Flushing (several men.years)

understanding how and why US signaling rules work and why CBTC would integrate safely :

- **Write in a document a precursor of the proof of a property:**
  - **Using only well defined assumptions**
  - **Using only logics**
  - **Explain how the property can be deduced**
  - **Requires the help of the domain experts**
  - **60% of the total workload**
- **Convert the previous work into Event-B-models such that the proof of these models are equivalent to the previous reasoning**
  - **standalone Event-B work**
  - **40% of the total workload**

More invasive than an ISA

NYCT Culver

Model reuse
50% workload saved

RATP OCTYS CBTC

OCTYS to be deployed in all non automatic metro lines in Paris
Safety issue detected in the specification

2018

DESIGN FORMAL V&V

2018

DESIGN
FORMAL V&V

Linking spec with implementation details

Same approach:

- Write in a document a precursor of the proof of a property:
  - Using also B models / source code
- Convert the previous work into Event-B-models such that the proof of these models are equivalent to the previous reasoning
  - standalone Event-B work

- Alternative to “traditional” design and safety analysis methods based on scenario

Property-Based Modelling and Validation of a CBTC Zone Controller in Event-B
RSSR 2019
M. Comptier, M. Leuschel, F. Mejia, J.M. Perez

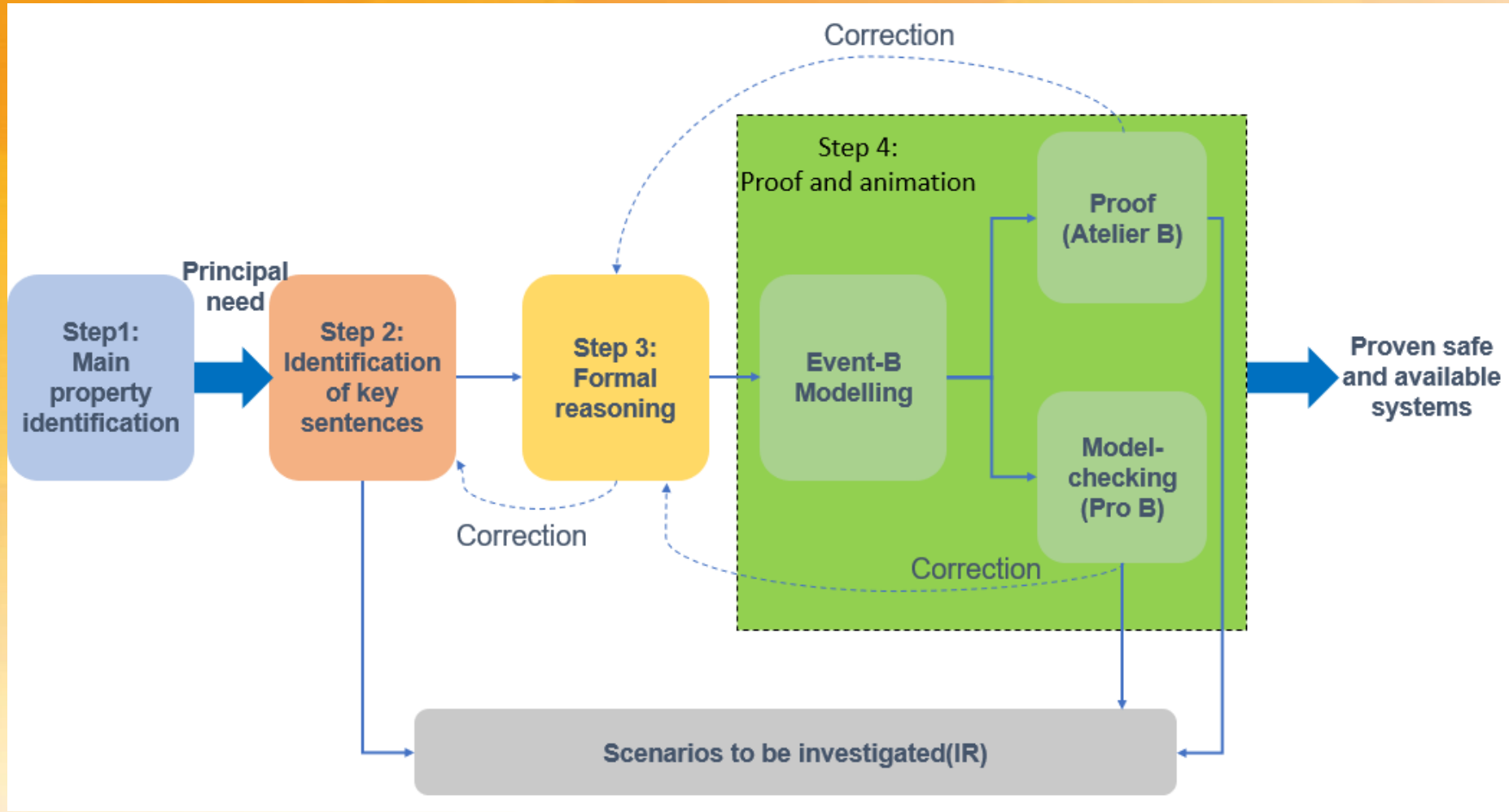
2018

DESIGN
FORMAL V&V

Mixing modelling for proof and for animation

Prove model

Verify that the model is able to play scenarios



2018

DESIGN
FORMAL V&V

Linking spec with implementation details

Main results:

- Retrieve and/or explain clearly the fundamental design principles
- Exhibit and explain formally the assumptions mandatory to demonstrate the conformity of the design wrt the system needs
- Carrying the proof with Atelier B guarantees that the demonstration is mathematically correct
- Retrieve and formalize the historical reasoning of the designers and keep track of their justification
- Identify complexity that is not necessary to maintain the properties
- Functional improvements and performance gains
- Possibly detect corner cases where the properties are not fulfilled
- The approach is pragmatic with a rapid delivery of concrete results

2019

CLEARSY SAFETY PLATFORM

2019

CLEARSY SAFETY PLATFORM

Safety for everyone

Developing a SIL3/4 system is difficult:

- Requires rare and overbooked « Leonardo Da Vinci » engineers
  - Need for understanding / mastering all details (hardware, software, environment, sensors, actuators, etc.)
  - Far from « it compiles, hence it works »
- The safety case has to demonstrate the safety

Double cœur et preuve formelle pour automatismes SIL4
LambdaMu 2016
T. Lecomte

Formal Methods in Safety-Critical Railway Systems
SBMF 2007
T. Lecomte, T. Servat, G. Pouzancre

Applying a Formal Method in Industry: A 25-Year Trajectory
SBMF 2017
T. Lecomte, D. Déharbe, E. Prun, E. Mottin

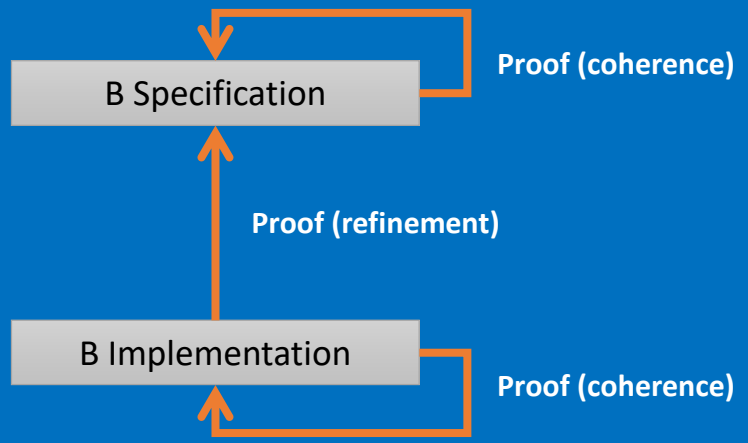
B only

« Only inactive sequences can be added to the active sequences execution queue. »

Natural language requirement

```
activation_sequence = /* Activation d'une séquence non active */
PRE ¬(sequences = sequences_actives) THEN
    ANY sequ WHERE
        sequ ∈ sequences - sequences_actives
    THEN
        sequences_actives := sequences_actives ∪ {sequ}
    END
END;
```

```
activation_sequence = /* Activation d'une séquence non active */
VAR sequ IN
    sequ <-- indexSequenceInactive;
    activeSequence(sequ)
END;
```



```
void M0__activation_sequence(void)
{
    CTX__SEQUENCES sequ;

    sequence_manager__indexSequenceInactive(&sequ);
    sequence_manager__activeSequence(sequ);
}
```

C generated code

```
0x01F970  FFFF 8B4C 2440 89C5 8D7D 0C8B 4110 89CE
0x01F980  83C6 0C8D 1485 0000 0000 8D42 0883 F807
0x01F990  7617 F7C7 0400 0000 740F 8B41 0C8D 7D10
0x01F9A0  83C6 0489 450C 8D42 04FC 89C1 C1E9 02F3
```

Binary code

B + SAFE PLATFORM

« Only inactive sequences can be added to the active sequences execution queue. »

Natural language requirement

```
activation_sequence = /* Activation d'une séquence non active */
PRE ¬(sequences = sequences_actives) THEN
    ANY sequ WHERE
        sequ ∈ sequences - sequences_actives
    THEN
        sequences_actives := sequences_actives ∪ {sequ}
    END
END;
```

```
activation_sequence = /* Activation d'une séquence non active */
VAR sequ IN
    sequ <-- indexSequenceInactive;
    activeSequence(sequ)
END;
```

```
void M0__activation_sequence(void)
{
    CTX__SEQUENCES sequ;

    sequence_manager__indexSequenceInactive(&sequ);
    sequence_manager__activeSequence(sequ);
}
```

```
0x01F970  FFFF 8B4C 2440 89C5 8D7D 0C8B 4110 89CE
0x01F980  83C6 0C8D 1485 0000 0000 8D42 0883 F807
0x01F990  7617 F7C7 0400 0000 740F 8B41 0C8D 7D10
0x01F9A0  83C6 0489 450C 8D42 04FC 89C1 C1E9 02F3
```

B Specification

Proof (coherence)

Proof (refinement)

B Implementation

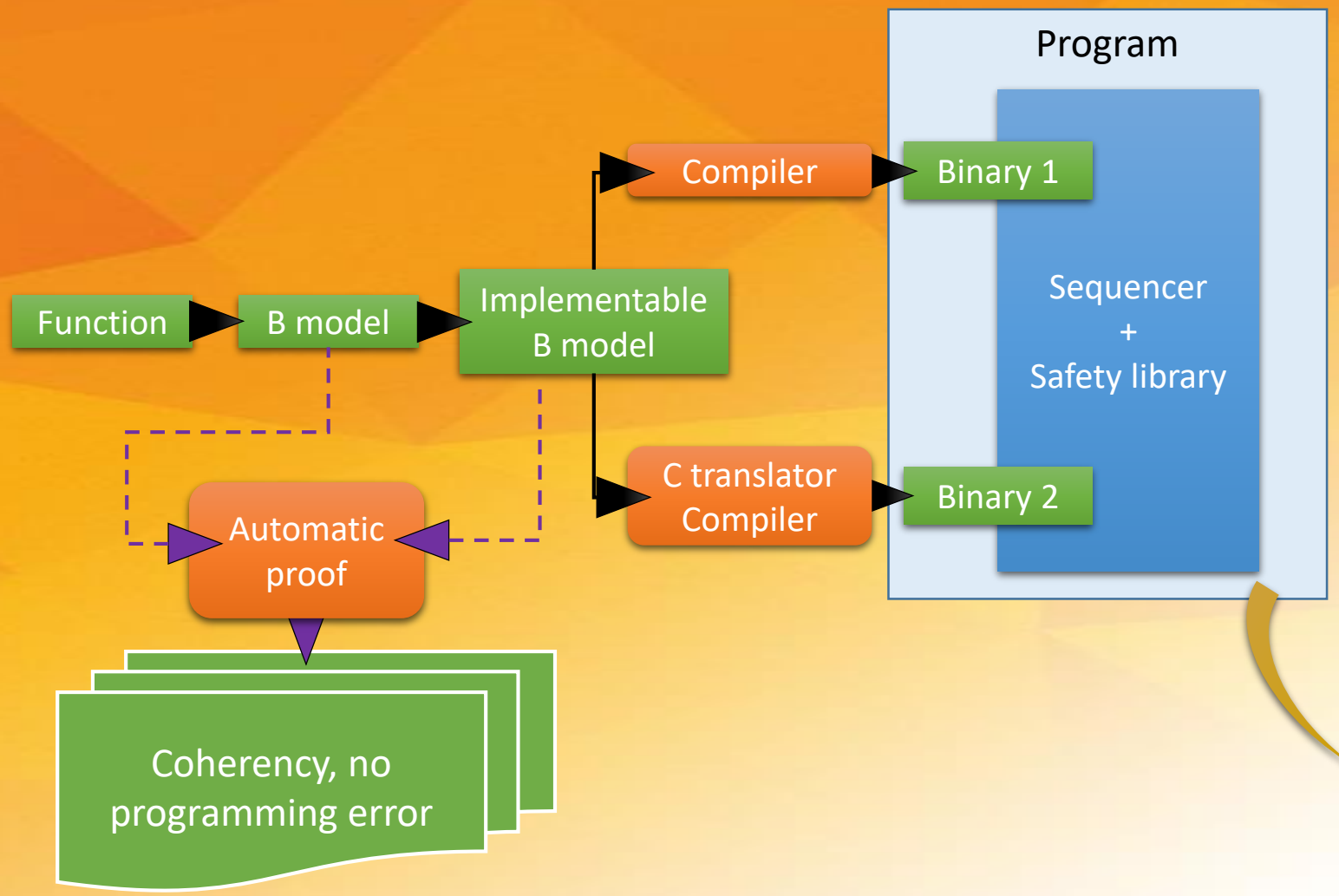
Proof (coherence)

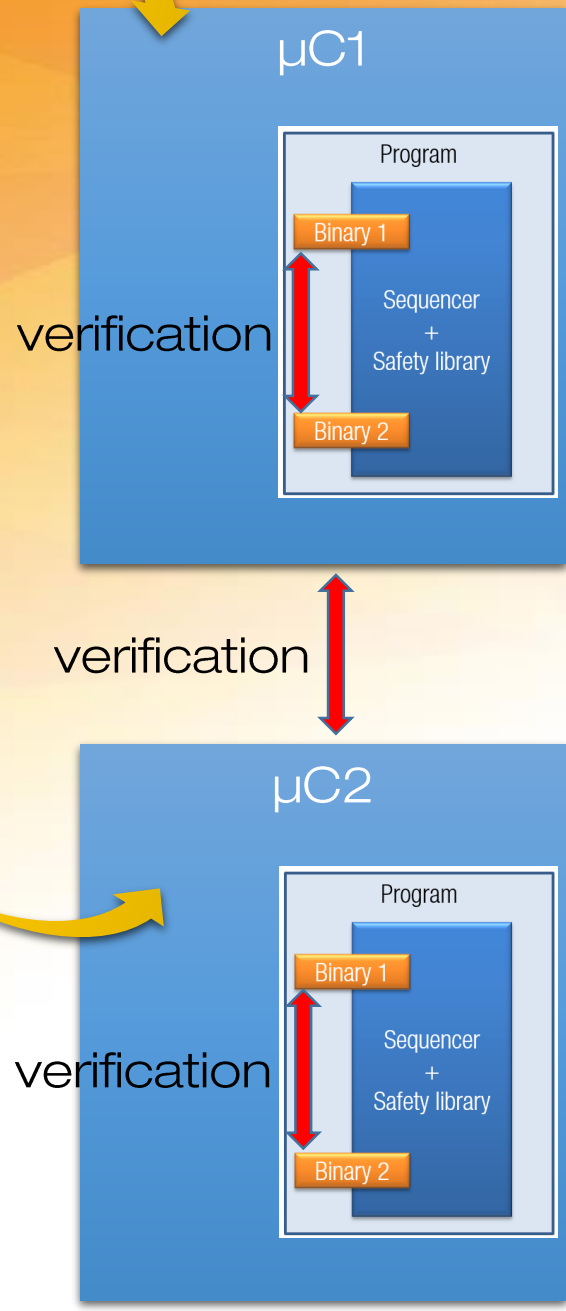
C generated code

Binary code

+

ARCHITECTURE & PROCESS





Outputs deactivated in case of misbehaviour

CSSP takes care of all safety verifications

If a divergent behaviour is detected

- one of the 4 software instances behaves differently
- one UC behaves differently

or if a structural error is detected

- bad CRC on memory
- UC unable to execute an operation properly
- etc.

then the detecting UC reboots

Verification intra-MCU
13 000 per second

Verification inter-MCU
48 per second

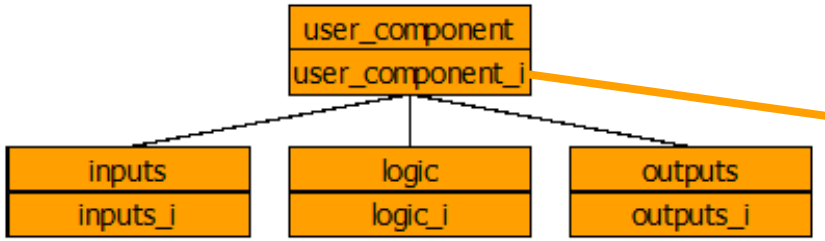
measured on Clock example

Verification deferred
over several cycles

CLEARSY
SYSTEMS ENGINEERING

B MODEL GENERATED

Syntax:
pp <-- ff(vv)
represents a call to operation ff(vv)
that returns the value pp




```
user_app =
BEGIN
    divergence_test_var :=  0;
    read_inputs;
    user_logic;
    write_outputs
END;
```

call

call

call

```
read_inputs  =
BEGIN
    I0 <-- read_global_input(0);
    I1 <-- read_global_input(1);
    I2 <-- read_global_input(2)
END;

po <-- get_I0 =
BEGIN
    po <-- read_global_input(0)
END;

po <-- get_I1 =
BEGIN
    po <-- read_global_input(1)
END;

po <-- get_I2 =
BEGIN
    po <-- read_global_input(2)
END
```

used by

```
user_logic = skip;

po <-- get_O0 =
BEGIN
    po := O0
END;

po <-- get_O1 =
BEGIN
    po := O1
END
```

used by

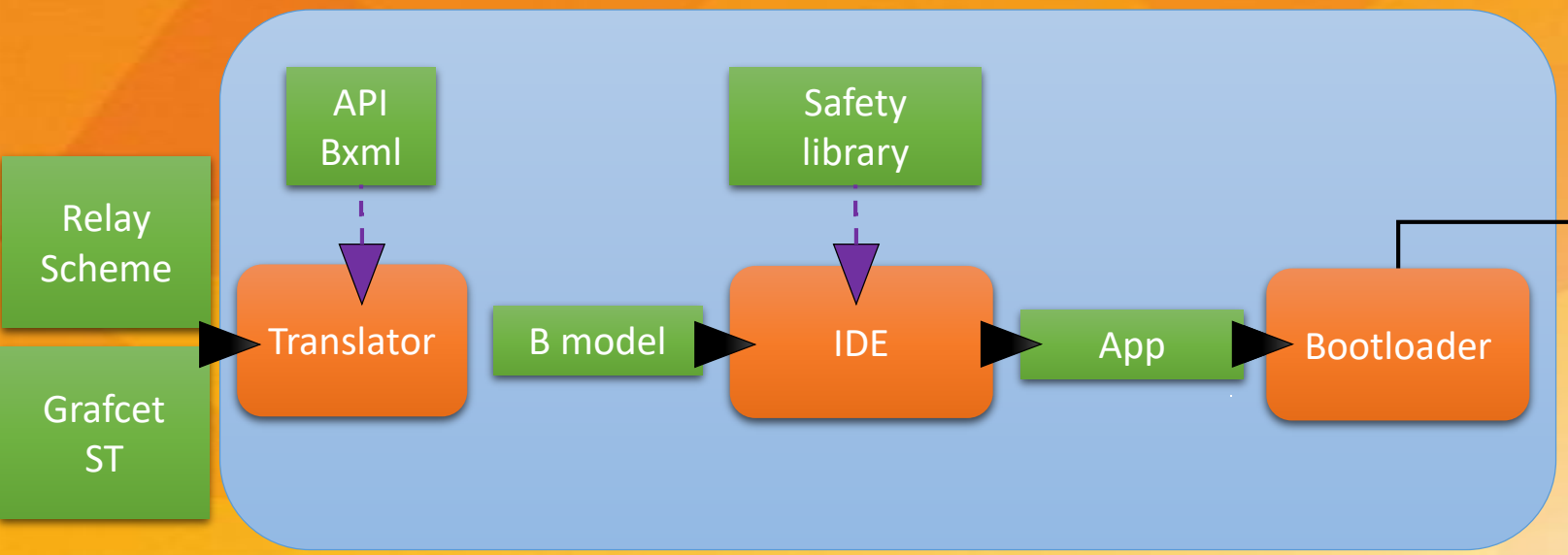
used by

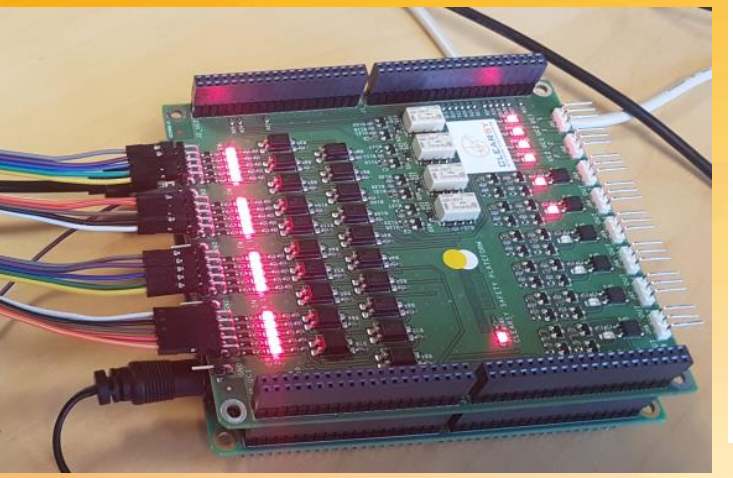
```
write_outputs =
VAR
    lsb
IN
    lsb:(lsb : uint8_t);

    lsb <-- get_O0;
    write_global_output(0, lsb);

    lsb <-- get_O1;
    write_global_output(1, lsb)
END
```

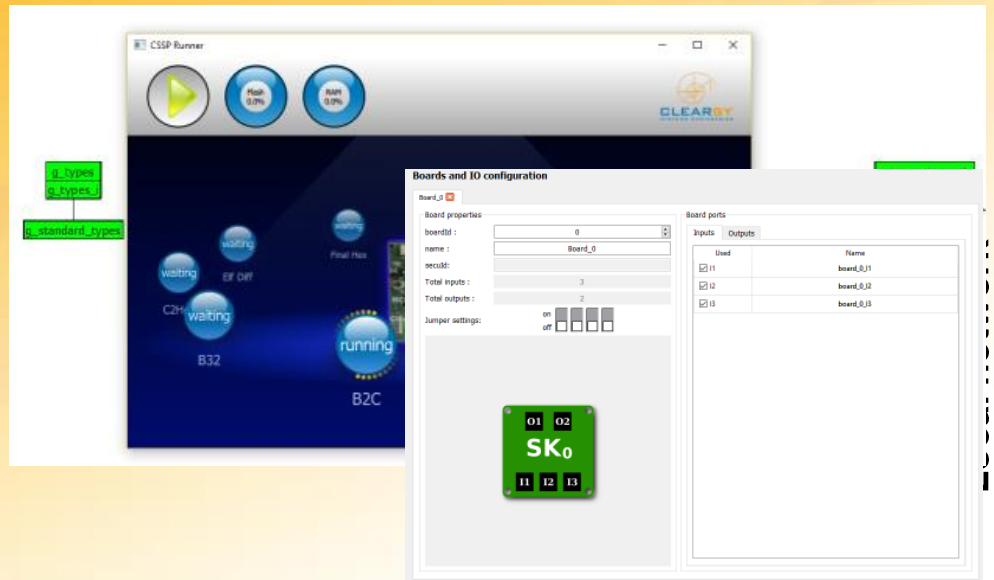
Programming CSSP is mainly implementing *user_logic* OPERATION

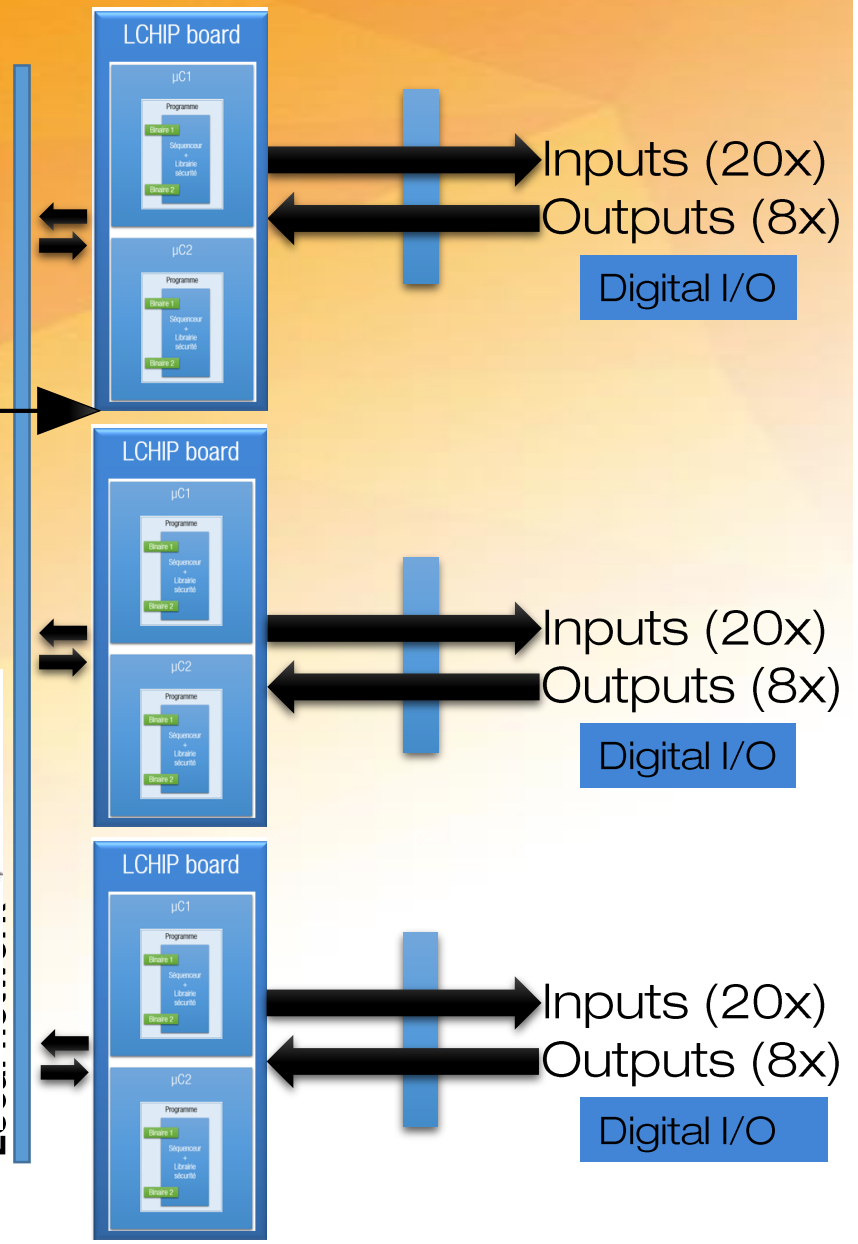
- Reading inputs values with *get_IO*, *get_I1* and *get_I2*
- Modifying the variables *O0* and *O1*

FORMAL IDE


API
Bxml
Safety
library
Relay
Scheme
Grafcet
ST
Translator
B model
IDE
App
Bootloader

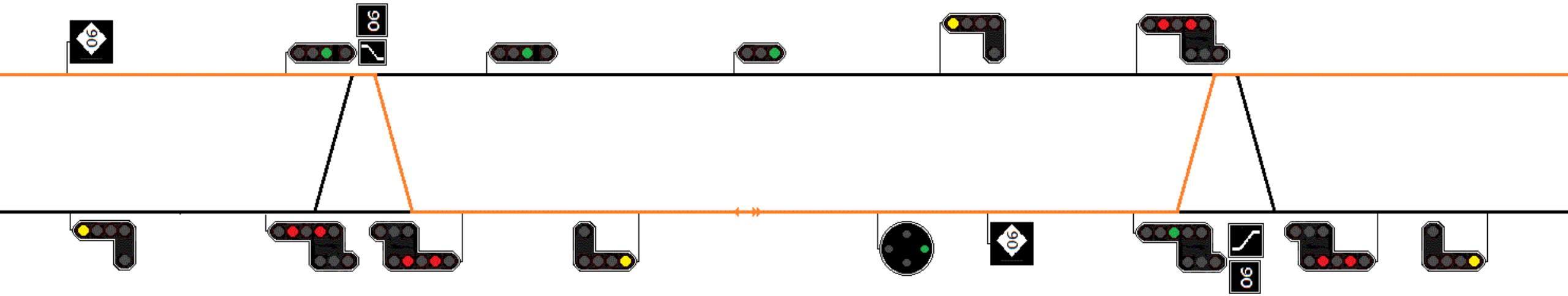


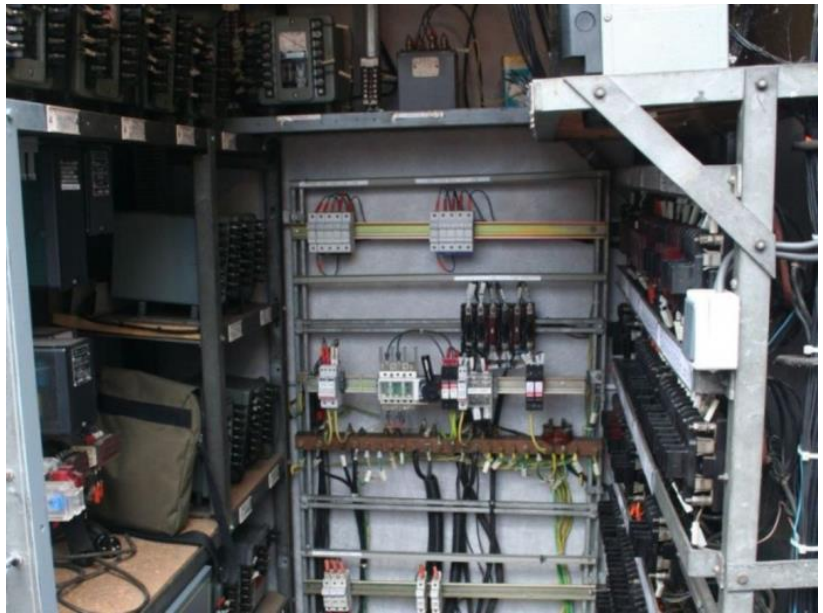

CSSP Runner
Boards and IO configuration
Board properties
Board ports
Inputs
Outputs
g_types
g_standard_types
waiting
running
B32
B2C
SK0


LCHIP board
µC1
µC2
Programme
Inputs (20x)
Outputs (8x)
Digital I/O
LCHIP board
µC1
µC2
Programme
Inputs (20x)
Outputs (8x)
Digital I/O
LCHIP board
µC1
µC2
Programme
Inputs (20x)
Outputs (8x)
Digital I/O

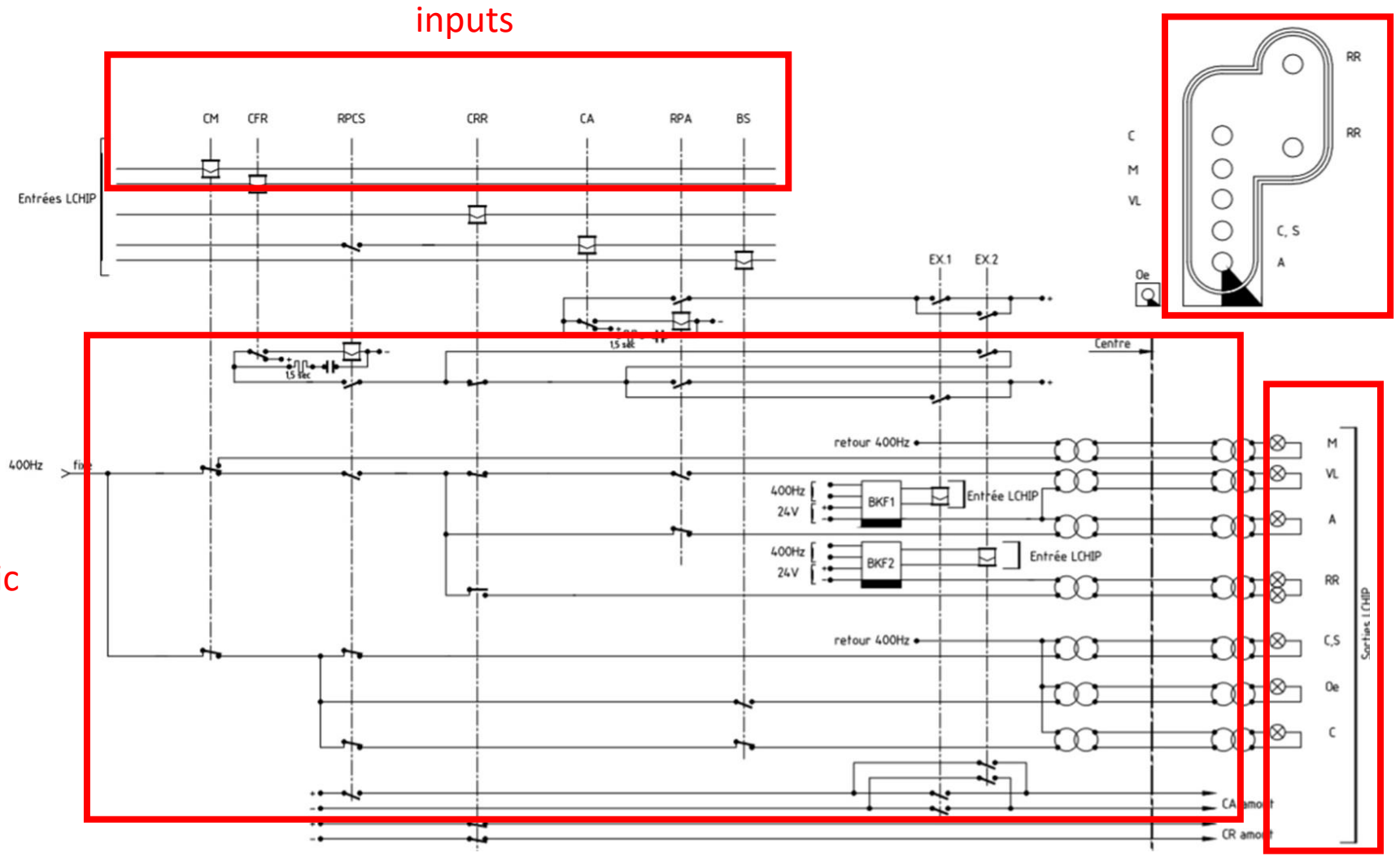
CSSP IN ACTION

Replace relay-based installation for temporary works



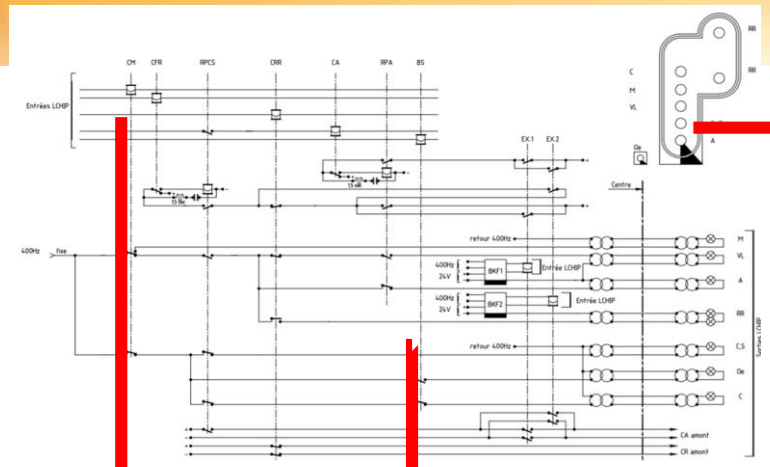


CSSP IN ACTION

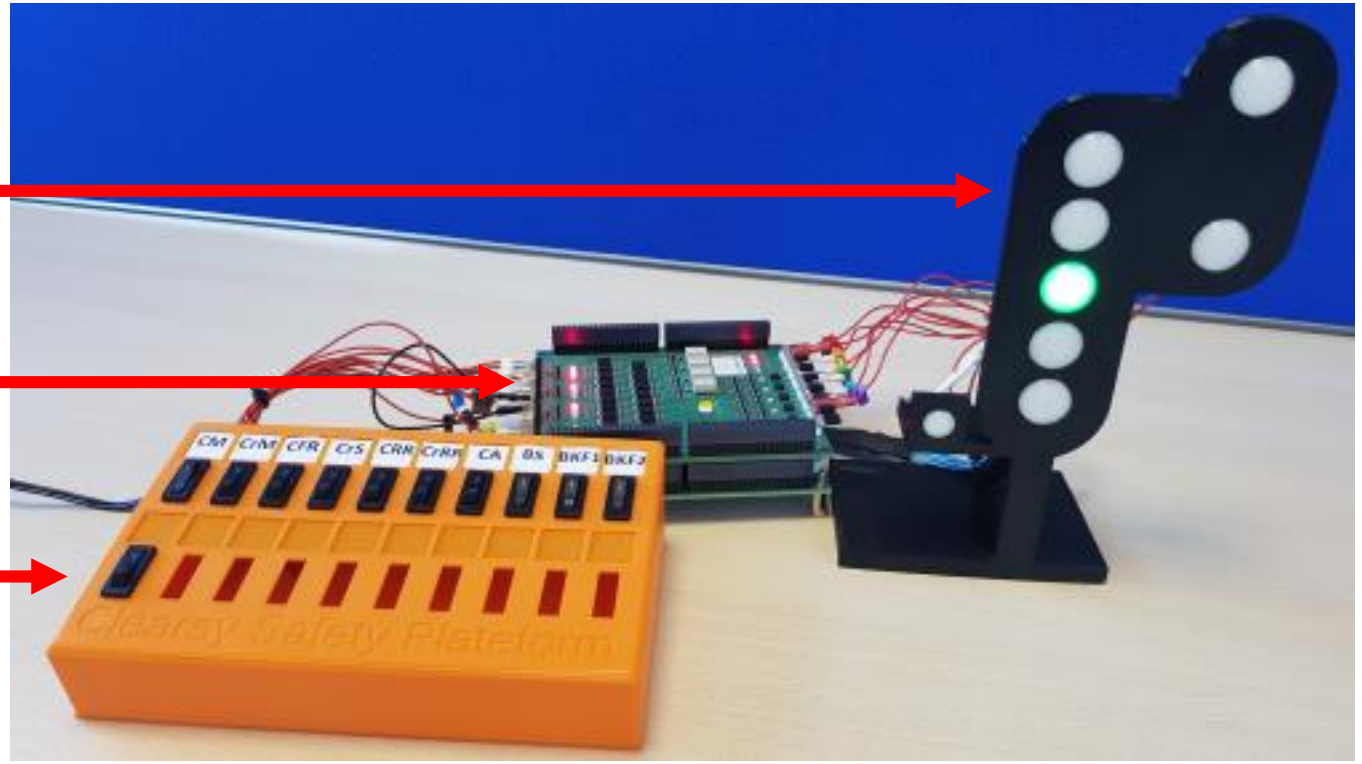


signal

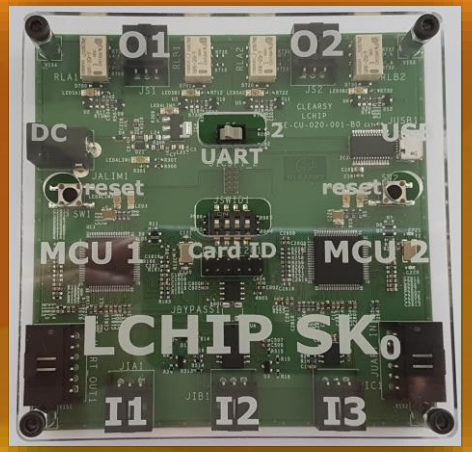
CSSP IN ACTION



- (System-level) model proved to avoid collision
- POC fully developed in 2 weeks



FORMAL METHODS | SAFETY



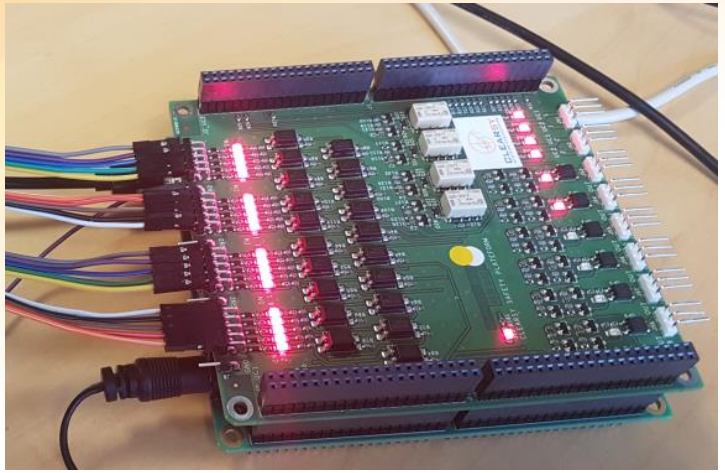
Starter Kit SK0

Programmable with B
Connection with DSL to avoid expert transaction
Detection of divergent behaviour
100 MIPS available
SIL4 ready
Building blocks already certified:

- São Paulo L15, 2017
- Stockholm Citybanan, 2017
- NYCT 2019, ongoing

Board SK0 available for education Q2 2019
Pedagogical kit containing:

- a user manual
- models and programs
- examples including other computers


Starter Kit SK1

http://www.clearsy.com/clearsy-safety-platform/

Formal Techniques For Safer Signalling Systems

REX & PERSPECTIVES

FORMAL METHODS NOT PANACEA

FORMAL REQUIRES PRECISION

" THE DEVIL IS IN DETAILS"

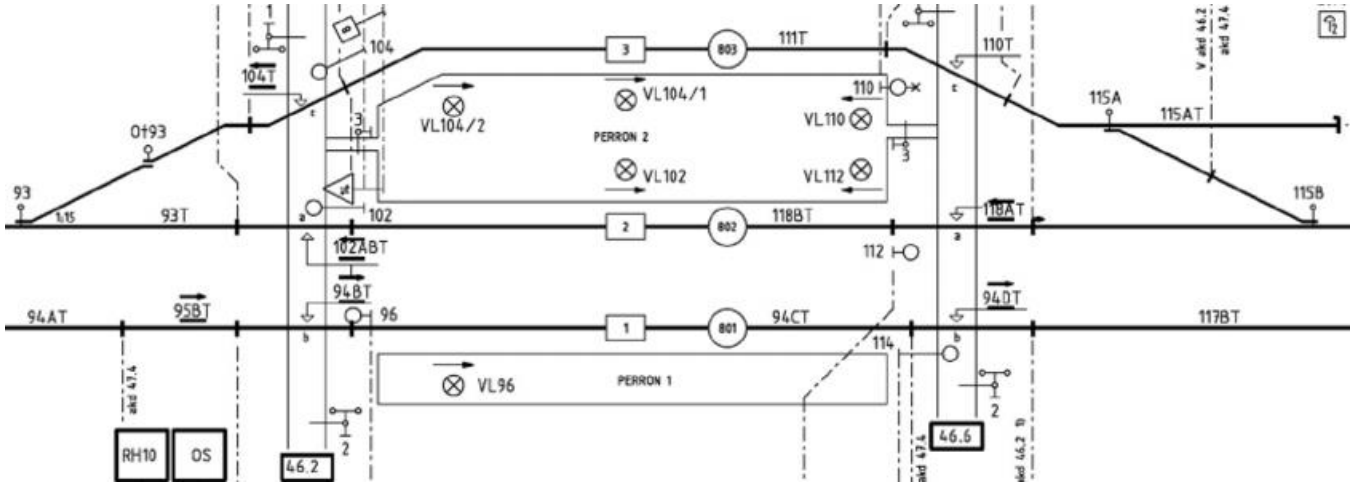
FORMAL SOFTWARE DEVELOPMENT

« There is overEnergy iff I can find a track section starting at X2M, complying with the dynamic chaining of blocks, on which I can

- either find a restriction belonging to a block such as the energy on that restriction, computed by summing deltas of energy of all restrictions located between X2MRes and this restriction, is greater than the energy associated to this restriction,

- or find 2 restrictions belonging to the EOA block, one being before the track section under consideration, the other after the track section, such as the energy associated to the EOA by using these restrictions is positive. »

[Extract from Automatic Train Protection specification]



FORMAL SOFTWARE DEVELOPMENT

p_over := bool (# (over_track) . ((over_track : seq (t_block * t_direction) & over_track /= {} & first (over_track) = p_X2MBlock |> p_X2MDir & ! ii . (ii : 1 .. size (over_track) - 1 => (over_track) (ii) : dom (sidb_nextBlock)) & ! ii . (ii : 1 .. size (over_track) => sidb_nextBlock ((over_track) (ii)) = (over_track) (ii + 1))) &(# (over_res) . ((over_res : sidb_restrictionApplicable & (# ii . (ii : dom (over_track) & ((prj2 (t_block , t_direction) (over_track (ii))) = c_up => over_res : ran (sgd_blockUpRestrictionSeq ((prj1 (t_block , t_direction) (over_track (ii)))))) & ((prj2 (t_block , t_direction) (over_track (ii))) = c_down => over_res : ran(sgd_blockDownRestrictionSeq ((prj1 (t_block , t_direction) (over_track (ii)))))) & (ii = 1 => not (over_res <= p_X2MRes)) & p_X2MSSWorst + p_X2MDSS + (SIGMA(jj) . (jj : 1 .. ii | SIGMA (pre_res) . (pre_res : t_restriction & ((prj2 (t_block , t_direction) (over_track (jj))) = c_up => pre_res : ran (sgd_blockUpRestrictionSeq ((prj1 (t_block, t_direction) (over_track (jj)))))) & ((prj2 (t_block , t_direction) (over_track (jj))) = c_down => pre_res : ran (sgd_blockDownRestrictionSeq ((prj1 (t_block , t_direction) (over_track (jj)))))) & (jj = 1 => not (pre_res <= p_X2MRes)) & (jj = ii => not (pre_res >= over_res)) | sgd_restrictionDeltaSqSpeed (pre_res)))) > sgd_restrictionSquareSpeed (over_res) & (over_res : sgd_restrictionFront => p_X2MResDist + ((SIGMA (ti) . (ti : 1 .. ii | sgd_blockLength ((prj1 (t_block , t_direction)((over_track) (ti))))))) ({ c_down |>sgd_blockLength (p_X2MBlock) sgd_restrictionAbs (p_X2MRes) , c_up |>sgd_restrictionAbs (p_X2MRes) } (p_X2MDir)) ({ c_down |>sgd_restrictionAbs (over_res) , c_up |>sgd_blockLength ((prj1 (t_block , t_direction) ((over_track) (ii)))) sgd_restrictionAbs (over_res) } ((prj2 (t_block ,t_direction) ((over_track) (ii))))) + sgd_restrictionLength (over_res) > loc_locationUncertainty + c_trainLength))))) or (# (eoa_res , res_after_eoa , ii) . (eoa_res : t_restriction & res_after_eoa : t_restriction & ii : dom (over_track) & p_EOABlock = (prj1 (t_block , t_direction)(over_track (ii))) & (ii = 1 => p_X2MRes <= eoa_res) & ((prj2 (t_block , t_direction) (over_track (ii))) = c_up => eoa_res : ran (sgd_blockUpRestrictionSeq (p_EOABlock)) & res_after_eoa : ran (sgd_blockUpRestrictionSeq (p_EOABlock)) & sgd_restrictionAbs (eoa_res) <= p_EOAAbs & p_EOAAbs < sgd_restrictionAbs (res_after_eoa) & ! ri . (ri : ran (sgd_blockUpRestrictionSeq (p_EOABlock)) => ri <= eoa_res or res_after_eoa <= ri)) & ((prj2 (t_block , t_direction) (over_track (ii))) = c_down => eoa_res : ran (sgd_blockDownRestrictionSeq (p_EOABlock)) & res_after_eoa : ran (sgd_blockDownRestrictionSeq (p_EOABlock)) & sgd_restrictionAbs (eoa_res) >= p_EOAAbs & p_EOAAbs > sgd_restrictionAbs (res_after_eoa) & ! ri . (ri : ran (sgd_blockDownRestrictionSeq (p_EOABlock)) => ri <= eoa_res or res_after_eoa <= ri)) & p_X2MSSWorst + p_X2MDSS + (SIGMA (jj) . (jj : 1 .. ii | SIGMA (pre_res) . (pre_res : t_restriction & ((prj2 (t_block , t_direction) (over_track (jj))) = c_up => pre_res : ran (sgd_blockUpRestrictionSeq ((prj1 (t_block , t_direction) (over_track (jj)))))) & ((prj2 (t_block , t_direction) (over_track (jj))) = c_down => pre_res : ran(sgd_blockDownRestrictionSeq ((prj1 (t_block , t_direction) (over_track (jj)))))) & (jj = 1 => not (pre_res <= p_X2MRes)) & (jj = ii => pre_res <= eoa_res) | sgd_restrictionDeltaSqSpeed (pre_res)))) ({ c_up |>(sgd_restrictionAccel (eoa_res) * ((sgd_restrictionAbs (res_after_eoa) p_EOAAbs) / 1024)) / 2 , c_down |>(sgd_restrictionAccel (eoa_res) * ((p_EOAAbs sgd_restrictionAbs (res_after_eoa)) / 1024)) / 2 } ((prj2 (t_block , t_direction) (over_track (ii))))) > 0)) or (# (eoa_res , ii) . (eoa_res : t_restriction & ii : dom (over_track) & (ii = 1 => not (eoa_res <= p_X2MRes)) & p_EOABlock = (prj1 (t_block , t_direction) (over_track (ii))) & ((prj2 (t_block , t_direction) (over_track (ii))) = c_up => eoa_res : ran (sgd_blockUpRestrictionSeq (p_EOABlock)) & eoa_res = last(sgd_blockUpRestrictionSeq (p_EOABlock)) & sgd_restrictionAbs (eoa_res) <= p_EOAAbs) & ((prj2 (t_block , t_direction) (over_track (ii))) = c_down => eoa_res : ran(sgd_blockDownRestrictionSeq (p_EOABlock)) & eoa_res = last (sgd_blockDownRestrictionSeq (p_EOABlock)) & sgd_restrictionAbs (eoa_res) >= p_EOAAbs) & p_X2MSSWorst + p_X2MDSS + (SIGMA (jj) . (jj : 1 .. ii | SIGMA (pre_res) . (pre_res : t_restriction & ((prj2 (t_block , t_direction) (over_track (jj))) = c_up => pre_res : ran(sgd_blockUpRestrictionSeq ((prj1 (t_block , t_direction) (over_track (jj)))))) & ((prj2 (t_block , t_direction) (over_track (jj))) = c_down => pre_res : ran(sgd_blockDownRestrictionSeq ((prj1 (t_block , t_direction) (over_track (jj)))))) & (jj = 1 => not (pre_res <= p_X2MRes)) & (jj = ii => not (pre_res >= eoa_res)) | sgd_restrictionDeltaSqSpeed (pre_res)))) + ({ c_up |> (sgd_restrictionAccel (eoa_res) * ((p_EOAAbs sgd_restrictionAbs (eoa_res)) / 1024)) / 2 , c_down |> (sgd_restrictionAccel (eoa_res) * ((sgd_restrictionAbs (eoa_res) p_EOAAbs) / 1024)) / 2 } ((prj2 (t_block , t_direction) (over_track (ii))))) > 0))

FORMAL DATA VALIDATION

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

TECHNOLOGY IS MATURE BUT REQUIRES MATURITY

- WHAT YOU GET IS WHAT YOU PROVED
- USE FORMAL METHODS WITH CARE
- TOOLING PRICE NOT A PROBLEM

HELP TO KEEP / RECOVER CONTROL

- “WHY WAS IT DESIGNED THIS WAY ?”
- “IS THIS DESIGN CORRECT / COHERENT / SAFE ?”

ROOM FOR RESEARCH

- NEED (MORE) OVERLAP BETWEEN ACTIVITIES
- HOT TOPICS: CENTRAL vs DISTRIBUTED, CYBER PHYSICAL SYSTEMS, CO-ENGINEERING (SAFETY/SECURITY), etc.

Thank you for your attention

Lyngby, Denmark, June 17th 2019

Thierry Lecomte
R&D Director, CLEARSY

thierry.lecomte@clearsy.com

CLEARSY
SYSTEMS ENGINEERING