Sur la pratique des méthodes formelles par de non praticiens : Autopsie d'un robot

*Eric JENN, IRT Saint Exupéry et Thales Avionics*

*Journée FMF*

*LAAS, 10 octobre 2017*
Collaborative work in INGEQUIP project at IRT

Pierre-Alain Bourdil
Arnaud Dieumegard
Ning Ge
Faiez Zalila

with academic support from LAAS, ONERA, IRIT, ISAE

with financial backing from CGI, ANR, and industrial members:
The patient

Pathologies

Data
Time
HMIs
SW impl.
Numerical

Remedies

Formal verif. of conf. data
Formal verif. of timed systems
Formal verification of HMIs
Correctness by construction
Formal code verification

The patient

Pathologies

Data
Time
HMIs
SW impl.
Numerical

Remedies

Formal verif. of conf. data
Formal verif. of timed systems
Formal verification of HMIs
Correctness by construction
Formal code verification
Virtual platforms
Design space exploration

Formal development methods
A robot

An environment

A supervision station
TwIRTee

EMBEDDED SYSTEM
TwIRTee

Ethernet CAM -> WIFI -> Ethernet

ethernet

CAN bus

MISSION

POWER

TREK1000 (ground)
TwIRTee

![Diagram of TwIRTee system with Ethernet, CAN bus, and Wi-Fi connections between devices.]

- Ethernet CAM
- WiFi
- ethernet
- CAN bus
- POWER

MISSION

CHANNEL LEFT

RIGHT

TREK1000 (ground)
TwIRTee

MISSION

MON: ZYNQ

COM: PPC

MON

COM

CAN bus

Ethernet CAM

WiFi

TREK1000 (ground)

POWER
TwIRTee

**MISSION**
- MON: ZYNQ
- COM: PPC
- OS: Trampoline
- OS: LINUX

**POWER**
- Encoder
- Motor
- H-bridge
- Buck analog part
- Compas
- I2C
- discrete
- analog
- discrete

**CAN bus**
- Ethernet CAM
- WiFi
- ethernet

**ETH**
- Ethernet
- WiFi

**OS**: LINUX

**TREK1000**
- (on-board)
- (ground)
Clock synchronization: Verification of timed systems

- Ethernet CAM
- WiFi
- ethernet
- CAN bus
- ZYNQ
- RAM
- ARP
- IBPM
- MM
- RCS
- RM
- CS
- OS: Trampoline
- PPC
- LOC
- GM
- MIM
- PM
- CS
- MON
- COM
- OS: LINUX
- MISSION
- POWER
- left and right wheel
- Encoder
- Motor
- H-bridge
- I2C
- Buck analog part
- Compass
- Power control
- Motor control
- MW: CAN driver
- OS: Trampoline
- TREK1000 (on-board)
- TREK1000 (ground)
- RAM
- PM
- RCS
- OS: LINUX
Clock synchronization: Verification of timed systems

Mission:
- ZYNQ
- PPC
- MON
- COM
- CAN bus
- Ethernet CAM
- WiFi
- Ethernet

Power:
- Encoder
- Motor
- H-bridge
- Compass
- Buck analog part
- I2C
- Left and right wheel

OS: Trampoline

RAM
- (on-board)
- PM
- RCS
- HMI
- RAM
- PM
- RCS
- HMI
- OS : LINUX

Power control
- MW: CAN driver
- OS: Trampoline
- CS
- PPC

RMS
- RAM
- ARP
- IBPM
- MM
- RCS
- RM
- CS
- OS : LINUX

Address:
- 09/10/2017
- © IRT AESE "Saint Exupéry" - All rights reserved Confidential and proprietary document
Verification of timed systems: Fiacre and Tina

- Ethernet CAM
- WiFi
- ethernet
- CAN bus

**MISSION**
- ZYNQ
- OS: LINUX
- ARP
- RAM
- PM
- RCS
- CS
- Motor
- Encoder
- Compas
- left and right wheel

**POWER**
- Motor
- H-bridge
- Buck analog part
- Compas
- I2C

- OS: Trampoline
- Trekmobil
- CAN bus
- Trekmobil (ground)
- Trekmobil (on-board)
- RAM
- PM
- RCS
- OS: LINUX
Verification of timed systems: Fiacre and Tina

Clocks Synchro

- Node 1 and 2 physical clocks
- Node 1 to 5 virtual (synchronized) clocks
- Node 3 virtual clock fault effect
- Node 3, 4 and 5 physical clocks
**Verification of timed systems: Fiacre and Tina**

### Clock synchronization protocol

#### Formalization (abstractions, hypothesis, …)

2.2.1 Hypothesis

- **CLOCK1**: the drift rate of a correct clock is $\rho = 10^{-6}$

2.2.2 Formalization

<table>
<thead>
<tr>
<th>Property</th>
<th>Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CLOCK1</strong>: the drift rate of the clock is $\rho = 10^{-6}$</td>
<td>Each temporal constraint of the form $[a, b]$ which relative to a duration measure by the physical local correct clock of a node is replaced with $[a(1 - \rho), b(1 - \rho)]$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Requirements</th>
<th>Model elements</th>
<th>Rationale</th>
</tr>
</thead>
<tbody>
<tr>
<td>Each temporal constraint of the form $[a, b]$ which relative to a duration measure by the physical local correct clock of a node is replaced with $[a(1 - \rho), b(1 - \rho)]$</td>
<td>Each temporal constraint of the form $[a, b]$ which model a duration relative to a timer is replaced with $[a(1 - \rho), b(1 - \rho)]$</td>
<td>Design choice =&gt; Duration measure local to a clock are done using timer. These are relative to local clock</td>
</tr>
</tbody>
</table>
Verification of timed systems: Fiacre and Tina

The model (structure)
Verification of timed systems: Fiacre and Tina

The model (behaviour)

```c
// Timer model a timer
process Timer | triggered: none | nid: Nodes, sarm: bool, &cancel: bool,
&fc: nFC, &failedClocks: FailedClocks
| is
states idle, armed, failing, triggered
from idle
on arm;
wait [0,0];
cancel := false;
to armed
from armed
select
on arm and not cancel;
select
on fc < FC and not failedClocks[nid];
// clock failing
wait [0,0.006]; failedClocks[nid] := true; fc := fc+1; to failing
[] on failedClocks[nid];
wait [0, ...]
[] on not failedClocks[nid];
wait [0.005,0.005]
end
[] on cancel; wait[0,0]; arm:=false; cancel:=false; to idle
end;
to triggered
from failing
select
on arm and not cancel; wait [0, ...]
// nevertheless, cancel is still possible
[] on cancel; wait[0,0]; arm:=false; cancel:=false; to idle
end;
to triggered
from triggered
select
cancel; wait[0,0]
// nevertheless, cancel is still possible
[] on cancel; wait[0,0]; arm:=false; cancel:=false; to idle
end;
to triggered
from triggered
select
[] on failedClocks[nid];
// clock failing
wait [0, ...]
```

It is always true that starting from a configuration where all nodes are in state “stop” with values “starts” greater or equal to 1, the system reaches a configuration where all nodes not in state “failSilent” are in state “synchroSuccess.”
Verification of timed systems: Fiacre and Tina

From a crude HMI...

... to a fancy Eclipse simulation environment

... with MSCs...

... and causal diagrams.
Verification of timed systems: Fiacre and Tina

- Some specific properties have been verified: are they worth the effort?
- How to gain confidence on the model?
- How to debug the model?

- From "simulation" to "advanced simulation"
  - Guided-simulation (property-driven simulation)
Redundancy Mngt: Verification of synchronous SW:
Verification of synchronous SW: Redundancy Mngt

CRM_R3. There shall never be two confirmed MASTERS in different channels.

CRM_P3.

∗(masterChannel = RIGHT & masterChannel = LEFT);

CRM_R5. The time window shall end at the same time on all channels.

CRM_P5.

ALL \[0, \text{NB_UNITS} - 1\] (env\text{ip}n[0])

\& SOME \[0, \text{NB_UNITS} - 1\] (∼inSW[i])

→ ALL j: \[0, \text{NB_UNITS} - 1\] (∼inSW[j]);
Verification of synchronous SW: Redundancy Mngt

- **P1**: The skew $\varepsilon$ of all the local clocks on all units is bounded
- **P2**: The network queuing delay $q$ and the network transmission delay $\mu$ are bounded
- **P3**: The task completion time $\alpha$ on all units is bounded
- **P4**: The global cycle time $T$ is such that

\[ T > \alpha_{\text{max}} + q_{\text{max}} + \mu_{\text{max}} \]
Mission management: Verification of configuration data

- **Ethernet CAM**
- **WiFi**
- **ethernet**

**MISSION**
- **ZYNQ**
- **RMS**
- **RAM**
- **ARP**
- **IBPM**
- **RCS**
- **RM**
- **CS**
- **OS: LINUX**

**POWERS**
- **Encoder**
- **Motor**
- **H-bridge**
- **left and right wheel**

**MON**
- **LOC**
- **GM**
- **MM**
- **PM**
- **CS**
- **OS: Trampoline**

**COM**
- **RAM**
- **PM**
- **RCS**
- **OS: LINUX**

**CAN bus**

**POWER**
- **Compas**
- **Buck analog part**

**MW: CAN driver**
- **Power control**
- **Motor control**
- **OS: Trampoline**

**TREK1000** (on-board)

**TREK1000** (ground)
Formal verification of configuration data

No path from a runway source node to this node
No path from this node to a runway sink node.
Formal verification of configuration data

[Carto-P001] No intersection of edges
∀ A, B, C, D ∈ Wps ∧ e1, e2 ∈ Edges ∧ A, B ∈ e1 ∧ C, D ∈ e2 ∧ (A ≠ C) ∧ (A ≠ D) ∧ (B ≠ C) ∧ (B ≠ D).
intersectant(e1, e2) = true ⇐ (ccw(A, B, D) ≠ ccw(B, C, D)) ∧ (ccw(A, B, C) ≠ ccw(A, B, D)),
where
ccw(A, B, C) = (C.y − A.y)(B.x − A.x) − (B.y − A.y)(C.x − A.x)

[Carto-P004] Connected runway nodes
∀ r1, r2 ∈ RUN
(r1 ≠ r2 → reachable(r1, r2) ∧ reachable(r2, r1)),
where reachable(a, b) stands for the node can reach the node b.

[Carto-P005] Aligned runway nodes
∀ r1, r2, r3 ∈ RUNWAY, r1 ≠ r2 ∧ r2 ≠ r3 → aligned(r1, r2, r3, δ),
where RUNWAY is the set of runway nodes, and δ is the tolerance alignment value.
Formal verification of configuration data

• A way to get acquainted to the formalism and tool...

• Consider dedicated tools!
Supervision: Formal verification of HMIs
Supervision: Formal verification of HMI

Alert 10
some text
Alert 2
Alert 3
some text
Alert 7
Alert 1
Some text

OS: LINUX
RCS_HMI
PM_HMI
RAM_HMI
Supervision: Formal verification of HMI

Predefined slots

Alert 10
some text

Alert 2

Alert 3
some text

Alert 7

Alert 1
Some text

OS: LINUX

RAM_{HMI}

PM_{HMI}

RCS_{HMI}
Supervision: Formal verification of HMI

Predefined slots
Formal verification of HMIs

All alerts
Alert 1
Alert 2
Alert 3
Alert 4
Alert 5
Alert 6
Alert 7
Alert 8
Alert 9
Alert 10

Priority-sorted alerts
Alert 10
Alert 2
Alert 3
Alert 7
Alert 10
Alert 4
Alert 5
Alert 6
Alert 8
Alert 9

Sorting network
LUSTRE

Target alert selection
LUSTRE

Target slot selection
LUSTRE

Routing
LUSTRE

Initial target alert

target alert

move=up/down/none

Target alert move

Target slot move

Alert 10
Alert 2
Alert 3
Alert 7
Alert 10
Alert 4
Alert 5
Alert 6
Alert 8
Alert 9

Alert 10
Alert 2
Alert 7
Alert 1
Alert 3
Alert 10
Alert 4
Alert 5
Alert 6
Alert 8
Alert 9

Alert 1
Alert 2
Alert 7
Alert 1
Alert 3
Alert 10
Alert 4
Alert 5
Alert 6
Alert 8
Alert 9
Formal verification of HMI

- **VISIBILITY**
  - **REQ-16**: Only active alerts shall be displayed.
  - **PROP-16**:
    \[
    \text{ALL } s: \text{slots} \left( s.\id = -1 \rightarrow \text{ALL } i: [0, 7] (a[i].\id = s.\id \rightarrow a[i].\act) \right)
    \]

- **TASK RELATED**
  - **REQ-8-1**: If the user clicks the [target alert up] button and the priority of previous selected alert is greater than the priorities of any current alerts, then the index of the returned alert shall be 0.
  - **PROP-8-1**:
    \[
    (\text{aselmov = up} \& \text{ALL } i: [0, 7] (p > a[i].p)) \rightarrow \text{idx} = 0
    \]

- **RELIABILITY**
  - **REQ-17**: The target alert shall always be displayed in the target slot.
  - **PROP-17**:
    \[
    \text{ALL } s: \text{Slots, a: Alerts} \left( (s.\x = \text{target}_x \& s.\y = \text{target}_y \& a.\idx = \text{target}\_index) \rightarrow s.\id = a.\id \right)
    \]
Formal verification of HMIs

Informal specification

Formal specification

Formal design

Formal verification of implementation
Formal verification of HMIs

• A very simple HMI designed using formal models and methods...

• HMIs deserve formal specification and verification
• HMIs are (probably) in the area where model-checking techniques are very efficient
Anticollision function: Correction by construction
Anticollision function: Correction *by construction*
Anticollision function: Correction by construction
Anticollision function: Correction by construction
Anticollision function: Correction by construction
Anticollision function: Correction by construction

- **Category 1.a:** writing rules aimed at avoiding modelling errors
  - **Rule 7:** Discriminate necessary and sufficient modelling elements
    - **Statement:**
      (e) Make a clear distinction between the model elements that are actual parts of the specification and/or design from those introduced to support the proof process. A naming convention can be applied to discriminate those modelling elements.
    - **Rationale:** The formal specification shall be kept as simple and short as possible. However, some modelling elements may be useful to simplify the proof. It is up to the designer to identify which one is to be kept according to the verification effort.
    - **Examples:**

- **Category 1.b:** writing rules aimed at facilitating the proof process

- **Category 1.c:** writing rules aimed at facilitating the review process

- **Category 2.a:** review rules aimed at revealing modelling errors

---

GETTING CONFIDENCE IN EVENT-B MODELS
 Guidelines

This document includes some guidelines to improve confidence in Event-B models.

A. Ursula de ROYANDE and Cor. JOAN

---
Anticollision function: Correction by construction

• A complete (simple) function formally developed top to bottom using a combination of Robin and S3

• No floating point operations

• Automation?
Image-based position monitoring: Numerical issues
Image-based position monitoring: Numerical issues

- Camera
- Image
- Segment extraction
- Segments
- Compliance image / location
- Envelope
- Computation
- Pose OK
- Pose KO
- Map
- Compass
- Odometry
- In-door positioning
- Hybridation
- Error ellipsoid
- Image / location
- Envelope computation
- ZYNQ (MON)
- PPC (COM)
Image-based position monitoring: Numerical issues

- Matrix inversion (Kalman)

```c
// Calcul du gain de Kalman
// On rappelle que S est semi-définis positive.
// K = P*H'*pinv(H);
double Kd[][1];
T_mat K = {1,1, _MAT_ Kd};
double SID[][1];
T_mat SI = {1,1, _MAT_ SI};
double H03[][1];
T_mat HT = {1,1, _MAT_ HT};
double H04[][1];
T_mat TMF4 = {3,1, _MAT_ TMF4};

mat_transp(H03, HT);
mat_inv(SI, SID);
mat_mul(H04, SID, TMF4);
mat_mul(Kd, TMF4, HT);
```
Image-based position monitoring: Numerical issues

- Matrix inversion (Kalman)
- Eigenvectors (error ellipsoid)
Image-based position monitoring: Numerical issues

• Matrix inversion (Kalman)

• Eigenvectors (error ellipsoid)

• Geometrical computations (enveloppe computation)
Image-based position monitoring: Numerical issues

• Excerpt from the BOOST library

**there are some other rules that valid**

Besides the concepts, which are checks on compile-time, there are some other rules that valid polygons must fulfill. This follows the opengeospatial rules (see link)

- Polygons are simple geometric objects (See also wiki but how polygons).
- If the polygons underlying ring_type is defined as clockwise, the exterior ring must have the clockwise orientation, and any interior ring must be counter clockwise. If the ring_type is defined counter clockwise, it is vice versa.
- If the polygons underlying ring_type is defined as closed, all rings must be closed: the first point must be spatially equal to the last point.

**There should be no self intersections**

- There should be no cut lines, spikes or punctures.
- The interior rings should be located within the exterior ring. Interior rings may not be located within each other.
Image-based position monitoring: Numerical issues

CGAL

\[ Speed = 1 \]

BOOST

\[ Speed = 50 \]

Strange, strange...
Formal verification of configuration data

• Found a place where $0.1 + 0.1 \neq 0.2$... and where it actually matters...

• Find the best representation for hardware implementation
To conclude: dealing with (so) many methods and tools...
To conclude: dealing with (so) many methods and tools...

- Structured argument
- Evidences
- Derivation tree
- Rules
- Leaves are "facts"
- Tools
- Safety case
- Verification arguments
- Translation
- Datalog program
- ETB
- Execution
- Orchestration

Datalog program
ETB

Safety case
Translation
Verification arguments
Translation
Datalog program
Execution
Orchestration

GSN

Derivation tree
Rules
Leaves are "facts".
Merci de votre attention