Expérimentation d’un langage de contexte et de propriétés pour la validation formelle de modèles logiciels

Philippe Dhaussy*, Arnaud Monégier du Sorbier**, Eric Bonnafous***

* Laboratoire LISyC-DTN, ENSIETA
dhaussy@ensieta.fr
** THALES AIR SYSTEMS
arnaud.monegier@thalesgroup.com
*** CS-SI
eric.bonnafous@c-s.fr

22 octobre 09
Sommaire

Goals

Approach

Experiments

Discussion
Motivation: Integration of formal methods in the engineering processes

Adoption of a pragmatic approach.

Currently, we focus on:

- a formalization of use cases (contexts) and requirements
- a construction of a methodology for model validation without changing in deep their practices.

- Difficulty to formalize the requirements
- Semantic gap between
  - the system to be validated
  - the formal model needed for the validation.
**Motivation : Integration of formal methods in the engineering processes**

Adoption of a pragmatic approach.

Currently, we focus on:

- a formalization of use cases (contexts) and requirements
- a construction of a methodology for model validation without changing in deep their practices.

High expressiveness, but not easily readable
Not easy to handle by the engineers in industrial projects.
**Motivation: Integration of formal methods in the engineering processes**

Adoption of a pragmatic approach.

Currently, we focus on:

- a formalization of use cases (contexts) and requirements
- a construction of a methodology for model validation without changing in deep their practices.

![Diagram showing the integration of formal methods in engineering processes.](image-url)

```
We want to identify contexts to verify properties .
```
Sommaire

Goals

Approach

Experiments

Discussion
Context expression

Contexts: to well-defined operational phases
initialization, reconfiguration, degraded modes, error scenarios, etc.

In reality, useful information:
very often implicit
disseminated in several
documents

Model to be Validated
SRS-WTOS-REQ-004

On receipt of a *MsgFieldMask* message from the COMM_WT, the WT_IOS shall set the WT_State to ‘STANDBY’ and transmit the *EvtTechnicalStateIos* message to the SYST_DP with the following parameters:
- `equipmentId` = `equipmentId` of the WT_IOS
- `roleId` = `roleId` of the WT_IOS
- `state` = STANDBY

If the requested WT_State is OPERATIONAL, the WT_IOS shall transmit the *MsgControlNetwork* message to the COMM_WT with the following parameters:
- `orderId`
- `command` = ‘READY’

End Requirement
Context expression

Method to support the formal specification of these contexts

Model: tightly synchronized with its environment.
Circumvent the combinatorial explosion

Formalization

Context (scenarios)

Model to be Validated
A strong hypothesis

The proof relevance: based on a strong hypothesis:

It is possible to specify the sets of bounded behaviors in a complete way.

Not formally justified

The essential idea:

The designer can correctly develop a software only if he knows the constraints of its use.

It is particularly true in embedded system domain.
Pragmatic approach for integration in engineering processes

Adoption of a pragmatic approach.

Currently, we focus on:

• a formalization of use cases (contexts) and requirements
• a construction of a methodology for model validation without changing in deep their practices.
Context Description Language (CDL)

Description of the environmental context: difficult task.

Context Description Language (CDL):
- to specify the context with scenarios and temporal properties
- to link each property to a limited scope of the system behavior.
**Verification principles**

Based on a composition of the context, a set of requirements and the model to be validated.
Verification principles: Composition

Before the composition, each property is transformed into an observer automaton.

Verdict

Reachability analysis

Labeled Transition System (LTS)
Observers

Observes the behavior of another entity or program

- Represent properties to be validated on a model.
- Not intrusive.
- Observer and model: IF2 language [Verimag]

Model to be validated

Model  ||  Observer

reachability Analysis

Verdict
Verification principles

Identification of several contexts.
Validation with a set of CDL models (a set of proof units)
CDL (Context Description Language)

CDL: DSL prototype, based on UML 2 [Whittle’s UCC]
Description of the behavior of actors of the environment.

Actors described by activity and sequence diagrams

Those entities are running in parallel.
CDL: hierarchically constructed in 3 levels
Context – properties linking

SYST-DP-REQ-6
During initialization procedure, the SYST_DP shall associate a generic equipment identifiers to one or several role in the system (MainSensor, OtherSensor, IFF, Actuator, ...). It shall also associate an identifier to each console. The SYST_DP shall send an evtEquipmentRole message, in preparation mode, for each connected generic equipment, to each connected console. Initialization procedure shall end successfully, when the SYST_DP has set all the generic equipment identifiers and all console identifiers and all evtEquipmentRole message have been sent.
End

SYST-DP-REQ-8
Once initialization is achieved, the SYST_DP shall send to each console an evtCurrentMission with curMission set to IDLE, to set current mission to idle, followed by an evtCurrentActivity with curActivity to LOGIN and status to TRUE to activate login.
End

Industrial projects:

Requirements:

• not associated to the entire lifecycle of software
• only to specific steps in its lifecycle.
**Context – properties linking**

**SYST-DP-REQ-6**

During initialization procedure, the SYST_DP shall associate a generic equipment identifiers to one or several role in the system (MainSensor, OtherSensor, IFF, Actuator, ...). It shall also associate an identifier to each console.

The SYST_DP shall send an evtEquipmentRole message, in preparation mode, for each connected generic equipment, to each connected console. Initialization procedure shall end successfully, when the SYST_DP has set all the generic equipment identifiers and all console identifiers and all evtEquipmentRole message have been sent.

End

**SYST-DP-REQ-2**

Once initialization is achieved, the SYST_DP shall send to each console an evtCurrentMission with curMission set to IDLE, to set current mission to idle, followed by an evtCurrentActivity with curActivity to LOGIN and status to TRUE to activate login.

End

**Context**

- properties
- specific context
The originality of CDL: ability to link each property to a context diagram (Level 1 or 2)
The benefits are:

- to explicit the conditions under which a given property is checked.

- Easier property specification
Requirement formalization

SYST-DP-REQ-6
During initialization procedure, the SYST_DP shall associate a generic equipment identifiers to one or several role in the system (MainSensor, OtherSensor, IFF, Actuator, ...). It shall also associate an identifier to each console.
The SYST_DP shall send an evtEquipmentRole message, in preparation mode, for each connected generic equipment, to each connected console.
Initialization procedure shall end successfully, when the SYST_DP has set all the generic equipment identifiers and all console identifiers and all evtEquipmentRole message have been sent.
End

Extracted from industrial documentation.

Textual requirement

→ ambiguous and complex.
Requirement decomposition

After a discussion with industrial partners, we had to rewrite it:

→ to decompose in a set of requirements.
→ to formalize them.

SYST-DP-REQ-6-1
During initialization procedure, the SYST_DP shall associate an identifier to NC console (IHM), before dMax_cons time units.

SYST-DP-REQ-6-2
After initialization, in preparation mode, the SYST_DP shall send an evtEquipmentRole for each connected generic equipment, to each connected console, before dMax_dev time units.

SYST-DP-REQ-6-3
Each device returns a statusRole message to SYST_DP before dMax_ack time units.

SYST-DP-REQ-6-4
The SYST_DP shall send an notifyRole message for each connected generic device, to each connected console. Initialization procedure shall end successfully, when the SYST_DP has set all the generic device identifiers and all console identifiers and all notifyRole messages have been sent.
Property formalization with definition patterns

**Property SYST-DP-REQ-6-1;**

```
AN
  exactly one occurrence of chgt_state_SYST
end

eventually leads-to [ 0 .. dmax_cons [ ALL combined
  exactly one occurrence of send_1_cons
  exactly one occurrence of send_2_cons
end

chgt_state_SYST may never occur
one of send_1_cons cannot occur before the first one of chgt_state_SYST
one of send_2_cons cannot occur before the first one of chgt_state_SYST
repeatability : true
```

Pattern-based approach
[ Dwyer, Cheng]
+ extensions

**SYST-DP-REQ-6-1**
_During initialization procedure, the SYST_DP shall associate an identifier to NC console (IHM), before dMax_cons time units._

*Response, Precedence, Absence, Existence*
Property definition pattern: detectable events

Detectable events:
transmissions or receptions, actions, model state changes.

Property SYST-DP-REQ-6-1;

AN
   exactly one occurrence of chgt_state_SYST
end

eventually leads-to [ 0 .. dmax_cons [ 

ALL combined
   exactly one occurrence of send_1_cons
   exactly one occurrence of send_2_cons
end

chgt_state_SYST may never occur
one of send_1_cons cannot occur before the first one of chgt_state_SYST
one of send_2_cons cannot occur before the first one of chgt_state_SYST
repeatability: true
Possibility of handling **sets of events**, ordered or not ordered.

Property SYST-DP-REQ-6-1;

- **AN** exactly one occurrence of chgt_state_SYST end
- eventually leads-to \([ 0 .. dmax_cons ]\)

- **ALL combined**
  - exactly one occurrence of send_1_cons
  - exactly one occurrence of send_2_cons
- chgt_state_SYST may never occur
- one of send_1_cons cannot occur before the first one of chgt_state_SYST
- one of send_2_cons cannot occur before the first one of chgt_state_SYST

**Repeatability**: true

**AN**: an event

**ALL**: all the events
Property definition pattern: options

Enrichment with Options using annotations [Smith]. To produce distinct variations on a property pattern.

Property SYST-DP-REQ-6-1;

AN

exactly one occurrence of chgt_state_SYST
end

eventually leads-to [ 0 .. dmax_cons [ 

ALL combined

exactly one occurrence of send_1_cons
exactly one occurrence of send_2_cons
end

chgt_state_SYST may never occur
one of send_1_cons cannot occur before the first one of chgt_state_SYST
one of send_2_cons cannot occur before the first one of chgt_state_SYST

repeatability: true

Pre-arity, Post-arity, Immediacy, Precedence, Nullity, Repeatability
Transformation into an observer automaton

Each property is transformed into an observer automaton (reject node)
Safety and bounded liveness properties.

Pty SYST-DP-REQ-6-1

Accessibility analysis:
Checking on the global LTS if a reject state is reached
Sommaire

Goals

Approach

Experiments

Discussion
Toolset : Observer-Based Prover
(Eclipse Plugin, EPL licence, TopCased project)

Proof Unit

Model to be Validated (UML, SysML, AADL, SDL)

automated

automated or manual

OBP takes as input the model to be validated and each CDL model.
Connection to an existing academic simulat (with IF2 language or (soon) FIACRE)

Formal program (IF2 or FIACRE)

• Context path
• Observer automata
• Model under study

LTS

IFx (Verimag)

TINA (Laas)

Tools (simulator, Model-checker)
Generation of context paths

From CDL diagrams:
Generation of a set of context path automata.

Each path:
one environment run
one possible interaction between model and context
Observer-Based Prover

Each path is transformed in IF2 code and composed with a set of observers and the model.

One context path (IF2)

Observer automata

Labelled Transition System

To be validated
Generation, simulation, accessibility analysis

CDL Model

OBP_CDL

Link editor

OBP_VERIF

Model to be Validated

Proof return data

a context path

Several observer automata

Academic Simulator (IFx)

Labelled Transition System

OBP_VERIF

CDL Model

Proof return data
Methodology : process

<table>
<thead>
<tr>
<th>a) Context Description</th>
<th>b) Property Specification</th>
<th>c) Proof Unit Construction</th>
<th>d) Model under study</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actors identification</td>
<td>Pattern Selection</td>
<td>Link Properties to the Context</td>
<td></td>
</tr>
<tr>
<td>Basic Interactions Capture</td>
<td>Adding options and guards to handle events</td>
<td>Proof Unit</td>
<td></td>
</tr>
<tr>
<td>High level Behavior Construction</td>
<td>New Events Identified</td>
<td>Model Checker</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Component Model

Component behavior automation

Property == true
Some industrial experiments results

Several industrial software components of embedded systems.

For each component:

- requirement documents
  
  Use cases,

  Requirements (natural language)

- component executable model (UML or SDL)

- translated in IF2 models (manually or semi-automatically)
AFN (Aircraft Facilities Notification) is an ATC Data-Link application

AFN User Model: SDL Model
3 processes
13400 lines (96 states, 107 transitions)
Some industrial experiments results

In industrial documents, requirements:

- At different abstraction levels
  - extraction of requirements corresponding to the model abstraction level.
- Rewritten into a set of several properties
  - Decomposition
  - Pattern-based rewriting consequently to discussion with industrial partners.
## Some industrial experiments results

<table>
<thead>
<tr>
<th>Category</th>
<th>CS1 (Partner1)</th>
<th>CS2 (Partner1)</th>
<th>CS3 (Partner1)</th>
<th>CS4 (Partner1)</th>
<th>CS5 (Partner2)</th>
<th>CS6 (Partner2)</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Provable properties</strong></td>
<td>38/49 (78%)</td>
<td>73/94 (78%)</td>
<td>72/136 (53%)</td>
<td>49/85 (58%)</td>
<td>155/188 (82%)</td>
<td>41/151 (27%)</td>
<td>428/703 (61%)</td>
</tr>
<tr>
<td><strong>Non-computable properties</strong></td>
<td>0/49 (0%)</td>
<td>2/94 (2%)</td>
<td>24/136 (18%)</td>
<td>2/85 (2%)</td>
<td>18/188 (10%)</td>
<td>48/151 (32%)</td>
<td>94/703 (13%)</td>
</tr>
<tr>
<td><strong>Non-provable properties</strong></td>
<td>11/49 (22%)</td>
<td>19/94 (20%)</td>
<td>40/136 (29%)</td>
<td>34/85 (40%)</td>
<td>15/188 (8%)</td>
<td>62/151 (41%)</td>
<td>181/703 (26%)</td>
</tr>
</tbody>
</table>

### Number of properties translated from requirements.

Three categories of properties.

(For confidential reasons, company and system names are not mentioned in this presentation)
Some industrial experiments results

<table>
<thead>
<tr>
<th></th>
<th>CS1 (Partner1)</th>
<th>CS2 (Partner1)</th>
<th>CS3 (Partner1)</th>
<th>CS4 (Partner1)</th>
<th>CS5 (Partner2)</th>
<th>CS6 (Partner2)</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Provable properties</td>
<td>38/49 (78%)</td>
<td>73/94 (78%)</td>
<td>72/136 (53%)</td>
<td>49/85 (58%)</td>
<td>155/188 (82%)</td>
<td>41/151 (27%)</td>
<td>428/703 (61%)</td>
</tr>
<tr>
<td>Non-computable properties</td>
<td>0/49 (0%)</td>
<td>2/94 (2%)</td>
<td>24/136 (18%)</td>
<td>2/85 (2%)</td>
<td>18/188 (10%)</td>
<td>48/151 (32%)</td>
<td>94/703 (13%)</td>
</tr>
<tr>
<td>Non-provable properties</td>
<td>11/49 (22%)</td>
<td>19/94 (20%)</td>
<td>40/136 (29%)</td>
<td>34/85 (40%)</td>
<td>15/188 (8%)</td>
<td>62/151 (41%)</td>
<td>181/703 (26%)</td>
</tr>
</tbody>
</table>

Captured with our patterns, translated into observers
Some industrial experiments results

<table>
<thead>
<tr>
<th></th>
<th>CS1 (Partner1)</th>
<th>CS2 (Partner1)</th>
<th>CS3 (Partner1)</th>
<th>CS4 (Partner1)</th>
<th>CS5 (Partner2)</th>
<th>CS6 (Partner2)</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Provable properties</td>
<td>38/49 (78%)</td>
<td>73/94 (78%)</td>
<td>72/136 (53%)</td>
<td>49/85 (58%)</td>
<td>155/188 (82%)</td>
<td>41/151 (27%)</td>
<td>428/703 (61%)</td>
</tr>
<tr>
<td>Non-computable properties</td>
<td>0/49 (0%)</td>
<td>2/94 (2%)</td>
<td>24/136 (18%)</td>
<td>2/85 (2%)</td>
<td>18/188 (10%)</td>
<td>48/151 (32%)</td>
<td>94/703 (13%)</td>
</tr>
<tr>
<td>Non-provable properties</td>
<td>11/49 (22%)</td>
<td>19/94 (20%)</td>
<td>40/136 (29%)</td>
<td>34/85 (40%)</td>
<td>15/188 (8%)</td>
<td>62/151 (41%)</td>
<td>181/703 (26%)</td>
</tr>
</tbody>
</table>

Captured with our patterns but not translated into an observer (unbounded liveness)
Some industrial experiments results

<table>
<thead>
<tr>
<th></th>
<th>CS1 (Partner1)</th>
<th>CS2 (Partner1)</th>
<th>CS3 (Partner1)</th>
<th>CS4 (Partner1)</th>
<th>CS5 (Partner2)</th>
<th>CS6 (Partner2)</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Provable properties</td>
<td>38/49 (78%)</td>
<td>73/94 (78%)</td>
<td>72/136 (53%)</td>
<td>49/85 (58%)</td>
<td>155/188 (82%)</td>
<td>41/151 (27%)</td>
<td>428/703 (61%)</td>
</tr>
<tr>
<td>Non-computable properties</td>
<td>0/49 (0%)</td>
<td>2/94 (2%)</td>
<td>24/136 (18%)</td>
<td>2/85 (2%)</td>
<td>18/188 (10%)</td>
<td>48/151 (32%)</td>
<td>94/703 (13%)</td>
</tr>
<tr>
<td>Non-provable properties</td>
<td>11/49 (22%)</td>
<td>19/94 (20%)</td>
<td>40/136 (29%)</td>
<td>34/85 (40%)</td>
<td>15/188 (8%)</td>
<td>62/151 (41%)</td>
<td>181/703 (26%)</td>
</tr>
</tbody>
</table>

Not captured with our patterns.
(example : undetectable events for the observer)
Some industrial experiments results

<table>
<thead>
<tr>
<th></th>
<th>CS1 (Partner1)</th>
<th>CS2 (Partner1)</th>
<th>CS3 (Partner1)</th>
<th>CS4 (Partner1)</th>
<th>CS5 (Partner2)</th>
<th>CS6 (Partner2)</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Provable properties</td>
<td>38/49 (78%)</td>
<td>73/94 (78%)</td>
<td>72/136 (53%)</td>
<td>49/85 (58%)</td>
<td>155/188 (82%)</td>
<td>41/151 (27%)</td>
<td>428/703 (61%)</td>
</tr>
<tr>
<td>Non-computable properties</td>
<td>0/49 (0%)</td>
<td>2/94 (2%)</td>
<td>24/136 (18%)</td>
<td>2/85 (2%)</td>
<td>18/188 (10%)</td>
<td>48/151 (32%)</td>
<td>94/703 (13%)</td>
</tr>
<tr>
<td>Non-provable properties</td>
<td>11/49 (22%)</td>
<td>19/94 (20%)</td>
<td>40/136 (29%)</td>
<td>34/85 (40%)</td>
<td>15/188 (8%)</td>
<td>62/151 (41%)</td>
<td>181/703 (26%)</td>
</tr>
</tbody>
</table>

For the CS5 component:
the percentage (82%) of provable properties: very high.
Most of properties: written with a good property pattern matching.
Some industrial experiments results

<table>
<thead>
<tr>
<th></th>
<th>CS1 (Partner1)</th>
<th>CS2 (Partner1)</th>
<th>CS3 (Partner1)</th>
<th>CS4 (Partner1)</th>
<th>CS5 (Partner2)</th>
<th>CS6 (Partner2)</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Provable properties</td>
<td>38/49 (78%)</td>
<td>73/94 (78%)</td>
<td>72/136 (53%)</td>
<td>49/85 (58%)</td>
<td>155/188 (82%)</td>
<td><strong>41/151 (27%)</strong></td>
<td>428/703 (61%)</td>
</tr>
<tr>
<td>Non-computable</td>
<td>0/49 (0%)</td>
<td>2/94 (2%)</td>
<td>24/136 (18%)</td>
<td>2/85 (2%)</td>
<td>18/188 (10%)</td>
<td>48/151 (32%)</td>
<td>94/703 (13%)</td>
</tr>
<tr>
<td>properties</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Non-provable</td>
<td>11/49 (22%)</td>
<td>19/94 (20%)</td>
<td>40/136 (29%)</td>
<td>34/85 (40%)</td>
<td>15/188 (8%)</td>
<td>62/151 (41%)</td>
<td>181/703 (26%)</td>
</tr>
<tr>
<td>properties</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**For the CS6 component:**

the percentage (27%) is very low. 
difficult to re-write the properties from specification documentation. 
much time to interpret properties with our partners to formalize them with our patterns.
Some industrial experiments results

About forty properties have been formally verified
2 errors: detected in CS5 and CS6
Sommaire

Goals

Approach

Experiments

Discussion
Approach benefits

- Requirements often partially described.
- CDL: formalization of contexts and properties.
- Contribution to overcome the combinatorial explosion.
- Motivation of the partners for a more formal approach to express their requirements.
- Better appropriation of formal verification process by partners.
- Help to structure and formalize their specification.
**Difficulties**

- **Scenarios**: lack of complete and coherent description
  - Many discussions with experts required
  - Long discussions for understanding and capture in a model

- **Requirements**: difficulty to formalize them into formal properties.
  - Different abstraction levels.
  - Several interpretations.
  - Some refer to an applicable configuration, operational phase or history without defining it.

- **Complexity**: CDL programming / path set complexity.
SYST_DP-REQ-4
On receipt of a $MsgFieldMask$ message from the COMM_WT, the WT_IOS shall set the WT_State to ‘STANDBY’ and transmit the $EvtTechnicalStateIos$ message to the SYST_DP with the following parameters:
equipmentId = equipmentId of the WT_IOS
roleId = roleId of the WT_IOS
state = STANDBY
If the requested WT_State is OPERATIONAL, the WT_IOS shall transmit the $MsgControlNetwork$ message to the COMM_WT with the following parameters:
orderId
command = ‘READY’
End Requirement

For software engineers : easy CDL modeling
System engineers : don’t want formalize at the CDL level (too low abstraction level)
Currently, we are working on definition of specific patterns at higher abstraction level.
Needed transformations to generate CDL properties from these specific patterns.
TR6 process

Conception des diagrammes CDL :
A partir des exigences du système, de la spécification de l’environnement
CDL Model Generation

Use cases, requirements

Modeling patterns

Context and requirements (user formalism)

Transformation patterns

Manual (know how)

Automatic

Proof results

Currently: OBP-CDL experimented with Airbus, THALES, CNES, CS-SI, DGA, …

Model-checking
Merci pour votre attention