A Model-Based Methodology to Formalize Railway Systems

Prepared by: Melissa Issad (Ecole Centrale Paris/ Siemens)
Co-authors: Leila Kloul (Versailles University), Antoine Rauzy (Ecole Centrale Paris)
Outline

- Motivations
- SCOLA: a Scenario Oriented Language
- Modeling CBTC systems using SCOLA
- Modeling existing CBTC system specifications using SCOLA
- Conclusion and future work
Motivations

1. It is all about complexity!

“This century is the century of complexity, and complexity and its associated technologies and theories of artificial life, agent-based models, self-organization and the science of networks will revolutionize the way science is done”

Stephen Hawking, 2000
Motivations

Complex railway systems

Different Railway transportations

Different functions and applications
Motivations

2. V-Cycle for product development

V-Cycle in EN 50126
Motivations

IN PRACTICE

Needs analysis
Almost a year

System specification
More than a year

System design

System integration

System validation

Software development
Up to 5 years

Operational qualification

Motivations

Up to 5 years

More than a year

Almost a year
3. Limits of the system modeling

- Two main approaches for system modeling:
  - **Language centric**
    - Use of all the items provided by the language to model the system
    - Result: redundant or irrelevant information
  - **System centric**
    - Modify the modeling language to fit the system
    - Result: Not generic methodologies

- Modeling language with no semantics behind!

Example: UML, SysML, …etc
Motivations

4. Formal Modeling

Formal model

- System specification
- Safety analysis
- Software Engineering
Motivations

- Formalize the informal
- Unify the system description
- Link with external tools

- Obtain a graphical representation
5. Scenarios

- Set of multiple actions
- Triggered by events
- Divided into steps
- Representation of the system behavior
- Linked to the system requirements
- Allocation of components to actions
Instead of looking at systems options, we must identify systems concepts. Build a formal modeling language based on the concepts and that fits the behavior of the system.

**Where?**
- System architecture:
  - Functional view
  - Structural view
  - Behavioral view
  - Functional scenarios

**How?**
- Identifying the abstract concepts of the system and their relationships

**When?**
- At the very first steps of the system design

SCOLA, a Scenario Oriented Language
A **system** consists of:
- A set of **components** which execute **functions**
- A system can be seen at different **abstraction levels**

---

1. **Metamodel of a system in SCOLA**
2. What is a function?

A **function** can be a set of functions characterized by:

- an **ID**
- executed by **one or two components**.
- **the three different types** possible and receives and send data.

![Diagram](image)

- **Identifier**
  - 0..1 Id
- **In/out data**
- **Abstraction level**
- **Function**
  - 1
- **Component**
  - 1.2
- **Type**
  - Simple
  - Transfer
  - Test
3. What is a component?

A **component** can be a set of **multiple sub-components**, characterized by:
- an **ID**
- its ability to execute **functions**
- **interfaces** to be linked to other components

A component **receives** and **sends** information. Functions express the relationship between in and outs.
4. The operators of the language

- **Precedence:** $f_1 \rightarrow f_2$
- **Parallelism:** $f_1 \parallel f_2$
- **Choice:** $f_1 \lor f_2$
- **Cooperation:** from $C_1$ to $C_2$ by $C$
- **Assignment:** $L_n \downarrow L_{n+1}$

**Graphical Representation**

[Diagram showing the operators with arrows and nodes representing $f_1$, $f_2$, $C_1$, $C_2$, and $f$]
1. The non-equipped train is detected based on the occupation of track circuits.

2. Track circuits occupancy information is transmitted to wayside CBTC.

3. Wayside CBTC updates a track circuits occupancy map and computes the target point for train A. This target point must not be overpassed in order to guarantee anti-collision.

4. Wayside radio continuously transmits to train A its target point (∅).

5. Train A adapts its speed according to the protection distance to be maintained (∅).
1. Graphical Representation with SCOLA

Consider the Arrival At Station Scenario

- $f_{0,1}$: The wayside selects the stopping point
- $f_{0,2}$: The wayside sends the stopping point to the train
- $f_{0,3}$: The train triggers the braking system
- $f_{0,4}$: The train informs the wayside of the doors opening
- $f_{0,5}$: The wayside opens the platform doors
- $f_{0,6}$: The wayside informs the train of the platform doors opening
- $f_{0,7}$: The train opens the doors
- $f_{0,8}$: The train informs the passengers of the next stop station
- $f_{0,9}$: The wayside triggers a timer at the train stop
- $f_{0,10}$: The train triggers the propulsion system
Consider a function of the ‘Arrival At Station’ Scenario:

- $f_{0,3}$: The train triggers the braking system
  - $f_{1,1}$: The train detects that it is at the stopping point
  - $f_{1,2}$: The train informs the driver that it is at the stopping point
  - $f_{1,3}$: The train triggers the braking system
  - $f_{1,4}$: The train sends the braking information to the driver
Graphical representation of the function $f_{0,3}$
Modeling CBTC systems using SCOLA

Graphical representation of the Arrival at Station scenario

‘Arrival at Station’ scenario representation
Modeling CBTC systems using SCOLA

2. Textual representation of scenario in SCOLA

```
System CBTC {
    Architecture system {
        Component Train {
            Basic-Component OBCU ;
        }
        Component Wayside {
            Basic-Component WCU ;
        }
    }
    Block TGMT {
        system.Train train1, train2
        system.Wayside wayside
        system.Train.OBCU obcu1, obcu2
        system.Wayside.WCU wcu
    }
}
```
Modeling CBTC systems using SCOLA

Scenario system with TGMT {
    Scenario F01 = "The trains calculate their positions" {
        Action F11 = "Train1 determines its position" By TGMT.obcu1 ;
        Action F12 = "Train2 determines its position" By TGMT.obcu2 ;
        Script F11 || F12 ;
    }
    Scenario F02 = "The trains send their position to the wayside" {
        Transfer F11 = "Train1 sends its position to the wayside" From TGMT.obcu1 to TGMT.wcu ;
        Transfer F12 = "Train2 sends its position to the wayside" From TGMT.obcu2 to TGMT.wcu ;
        Script F11 || F12 ;
    }
    Action F03 = "The wayside determines the space limit between the two trains" by TGMT.wayside ;
    Scenario F04 = "The wayside transmits the safe distance limit to the trains" {
        Transfer F11 = "The wayside transmits to train1" from TGMT.wcu to TGMT.obcu1 ;
        Transfer F12 = "The wayside transmits to train1" From TGMT.wcu to TGMT.obcu2 ;
        Script F11 || F12 ;
    }
    If test = "Limit Reached" {Action F05 = "Emergency brake of the train2" By TGMT.obcu2 ;}
    else {Action F06 = "the train determine their speed limit plan" By TGMT.obcu1 ;
        Action F07 = "the train determine their speed limit plan" By TGMT.obcu2 ;}
    Script F01 -> (F02 -> (F03 -> test -> (test.F05 or (test.F06 -> test.F07))))) ;
}
}
Modeling existing CBTC system specifications using SCOLA

1. Do we need to be experts of the system to understand the specification?

2. What are the components of the system?

3. What is the relationship between the scenarios and the system architecture?

4. Do we need all the scenario details for each step of the system engineering?
Depending on what we want to do with the system (safety analysis, system validation & verification, software development), we might (not) need some irrelevant information.

Our solution:

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
<th>Level 0</th>
<th>Level1</th>
<th>Functional Architecture</th>
<th>Train</th>
<th>Wayside</th>
<th>Safety related</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step1</td>
<td>The wayside selects the stopping point</td>
<td>The wayside selects the stopping point</td>
<td>The ATS selects a stopping point at the platform via ATS_Select_Stopping_Point</td>
<td>F7</td>
<td>OSUB, Platform, HMI, Train Control, PIS/TMS</td>
<td>WCU_ATP: X, WCU_TTS: X, ATS: X</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The WCU_ATP now uses the selected stopping point in the WCDs to the on-board subsystem</td>
<td>F7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The WCU_ATP reflects the information on the selected stopping point to the ATS</td>
<td>F7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Step2</td>
<td>The train controls the braking action</td>
<td>The train controls the braking action</td>
<td>The on-board subsystem controls the braking action by the command TCL_ATC_Motoring/Braking and a corresponding value</td>
<td>F7</td>
<td></td>
<td></td>
<td>No</td>
</tr>
</tbody>
</table>

Modeling existing CBTC system specifications using SCOLA.
Conclusion

- A novel scenario based modeling formalism
- Two representations: Textual and graphical
- Relies on a formal semantics
- Provides multiple levels of abstraction
- Re-usable components
- Provides a help to the next steps of the process
- Generic enough to be used for all the complex systems
- A stepping stone for the dysfonctional scenarios modeling
On-going work

- Implementation of SCOLA
- Introduction of the exchanged data into the language
  - Create inputs/outputs for each function
  - Differentiate between safety data and non-safety one
Conclusion

On-going work

• Evaluation of SCOLA in the safety analysis
  ◦ Evaluate the matching concepts between system specifications and safety analysis
  ◦ Methodology to introduce the language in the existing approaches for safety analysis (on-going)
  ◦ Build an inductive and probabilistic approach to generate dysfunctional scenarios starting from the functional scenarios
  ◦ Create a benchmark for system specifications and safety analysis
References