Incremental model-based design methodology to develop CPS with Sysml/OCL/Reo

Perla Tannoury[1[1−2]], Samir Chouali[2[1−3−5]], and Ahmed Hammad[3[1−4−6]]

[1] Univ. Bourgogne Franche-Comté,FEMTO-ST Institute/CNRS, Besançon
[2] perla.tannoury@femto-st.fr
[3] schouali@femto-st.fr
[4] ahammad@femto-st.fr
[5] https://www.femto-st.fr/fr/personnel-femto/schouali
[6] https://www.femto-st.fr/en/femto-people/ahammad

1 Use case study

This use case study were inspired by an example that was used and detailed in depth in the previous work of one of the author [3]; the Communications-Based Train Control (CBTC) system example. The CBTC system is a signaling and control system for light rail in urban cities, such as tramways, as well as heavy rail, such as the metro and the APM (Automated People Mover) systems [2], such as the Airport metro. These systems provide railway operators exact control over train movement based on position data provided by high-precision on-board equipment. A continuous bi-directional track-train data transmission, on the other hand, can supply far more control and status information to the train. Today, most CBTC systems use radio transmission for this communication. These systems enable for more trains to travel on the same line at faster frequencies and speeds (with or without drivers), resulting in increased capacity, efficiency, and operational flexibility and safety, as well as lower operating costs. Fig.1 illustrates the CBTC system, and in the following parts we will focus on the CBTC SysML structure, Reo circuits, and how to combine both together. The CBTC system is made up of a number of components that are located both inside and outside of trains, that constantly interact to calculate and exchange parameters required to ensure the safe circulation of trains. Before diving into the SysML of the system and drawing its Reo circuits, we will start by introducing the system informally by showing the chaining of components and their interactions through five main events:

1. Sending coverage requests to MCU
   - On each train of T1 and T2, there is an OBD that will communicate with MCU component and ask if they are visible.
   - This process is done by sending the message covReq to Coverage sub-component.
2. T1 and T2 are covered
   MCU replies to T1 and T2 by sending covered or uncovered messages.
3. Requesting VMAZs from MCU
   - T1 and T2 will ask separately MCU for their Vital Movement Authority Zone(VMAZ), by sending the message vmazReq.
   - A VMAZ is an area where the train can safely circulate.
4. VMAZs of T1 and T2 from MCU
   - T1 and T2 are circulating safely on the railway due to MCU sub-components (Coverage and ProdVmaz).

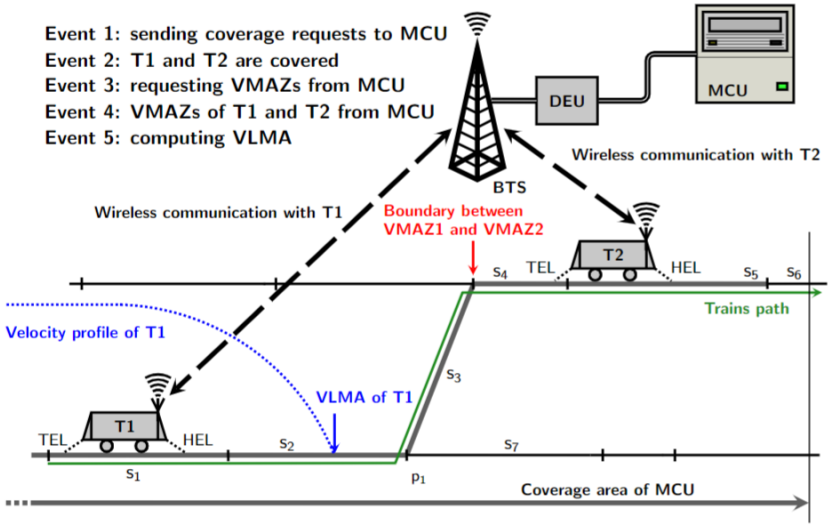


Fig. 1: Simplified trains protection functions [3].

– MCU ensures that the trains will never overlap, thus avoid collisions.

5. Computing VLMA
   Least but not last, the train computes the Vital Limit of Movement Authority(VLMA) by fixing a safety margin within the limit of its VMAZ. Based on VLMA the OBD component will either enable its sub-component CtrVelocity, or EmgcyBrake.

1.1 CBTC SysML diagrams

Requirement Diagram: Almost all design processes begin with some needs specified by requirements. Fig.2 specifies the system needs in a requirement diagram. In this diagram, the initial requirement R1 must ensure the visibility of the train throughout its journey without any problems, and it is refined by the requirement R1.1. In R1.1, MCU and OBD must interact with each other without deadlocks. Furthermore, R1.1 is decomposed into R1.11 where MCU must respond to OBD request within 5 unit times and it is satisfied by the block "OBD". Requirement R1.12 refines R1.1 and ensures that MCU must provide OBD requested services within 5 unit times and is satisfied by the block "MCU".

Extended Block Definition Diagram: Fig.3 shows an example of ExtBDD with seven blocks. It is the first level of modeling of the CBTC system. The block named "CBTC" represents the system as a whole. It is decomposed into two sub blocks (OBD and MCU) and is linked to them by the composition relationship. The component "OBD" is divided into two sub-components which are "CtrVelocity" and "EmgcyBreak". "MCU" is decomposed into two sub-components that are "ProdVMAZ" and "Coverage".

Reo Internal Block Diagram: Before diving into Reo IBD, we will compare the classic internal block diagram (IBD) to Reo IBD. The IBD is used as a detailed point of view of a specific component using its sub-components. The interaction between the sub-components is modeled by connectors between their

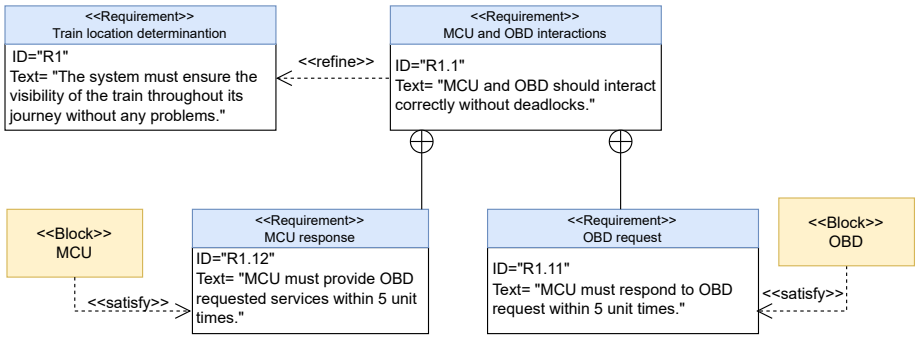


Fig. 2: Requirement diagram for CBTC system.

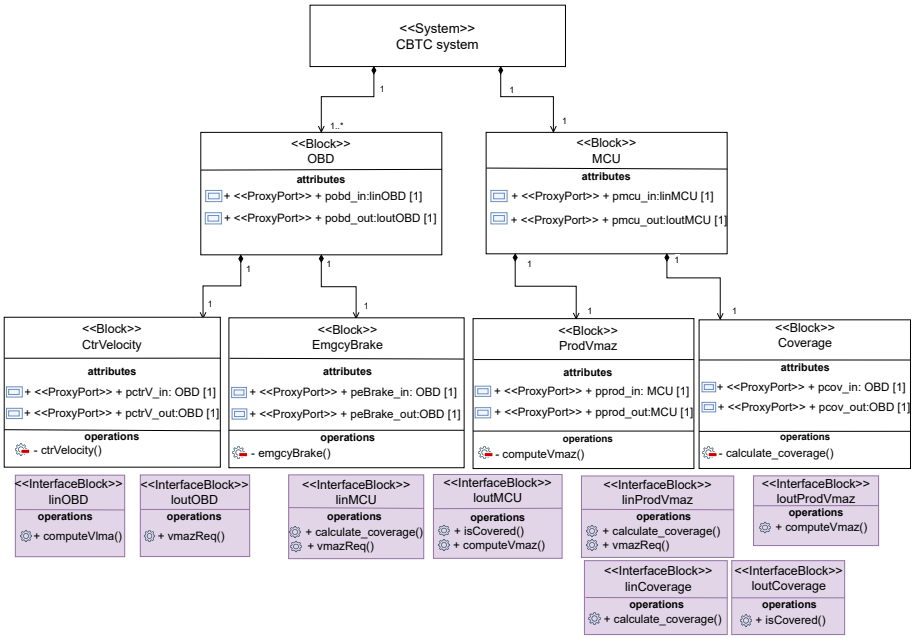
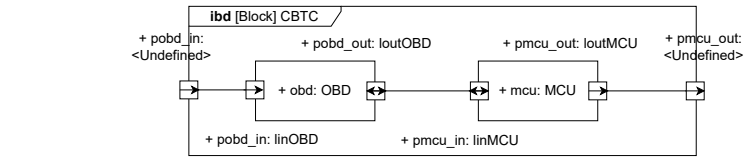
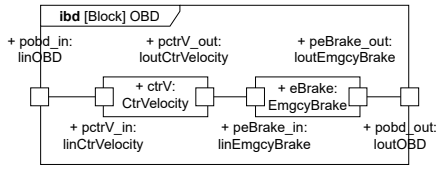


Fig. 3: CBTC Extended Block Definition Diagram.

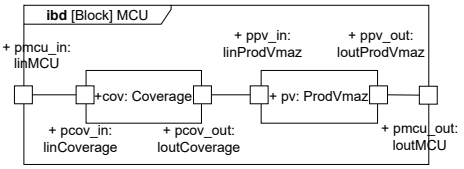
ports. The following two figures Fig.4b and Fig.4c illustrate the internal composition of the On Boarding Device (OBD) and the Movement Computing Unit (MCU). In Fig.4b, the OBD has two responses when it receives its VMAZ, either to slow down using CtrVelocity or to use the emergency brake (EmgcyBrake) in the event of a sudden accident or an emergency. In Fig.4c, the MCU waits for a signal from the OBD and then signals the train coverage. In addition the MCU also provides the OBD with a Vital Movement Authority Zone (ProdVmaz) where the train can circulate safely on the railway. Fig.4 is an illustration of how OBD and MCU work together internally. However, no matter how much we



(a) CBTC internal block diagram

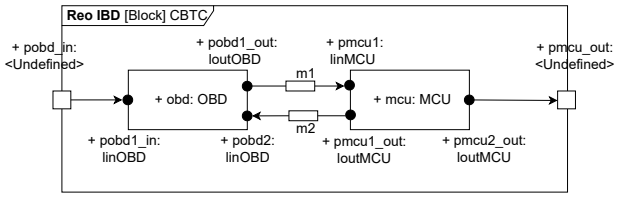


(b) OBD internal block diagram

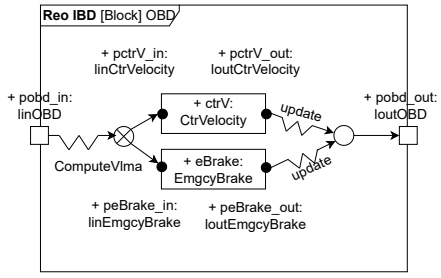


(c) MCU internal block diagram

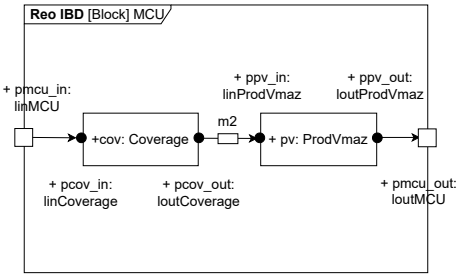
Fig. 4: CBTC, OBD and MCU internal block diagram



(a) Reo IBD of CBTC



(b) Reo IBD of OBD



(c) Reo IBD of MCU

Fig. 5: The railway architecture in a CBTC system using the new meta-model "Reo IBD".

detail the IBD of a component, we will never know the type of messages that are exchanged between the sub-components and how to specify protocols directly that will describe the interaction among components in an explicit and concrete way. Our main contribution is to build a new meta-model "Reo IBD" that combines IBD and Reo in order to: 1) Ensure compatibility of system components and validation of system properties. 2) Specify the connections between components in any abstract level so that the modeling can be more comprehensive and better matched with its real configuration. 3) Characterize message flow and

Table 1: OCL on Reo IBD of OBD and MCU.

ReoIBD of	Description	OCL
OBD	Reo IBD of OBD represents the sub-components of its associated OBD component	`context Composition inv: src in OBD::allInstances() implies tgt in ReoIBD-OBD::allInstances()`
MCU	Reo IBD of MCU represents the sub-components of its associated MCU component	`context Composition inv: src in MCU::allInstances() implies tgt in ReoIBD-MCU::allInstances()`

their properties explicitly. 4) Provide a friendly and detailed interface. 5) Allow resuming the thousands of document lines using a set of convivial and graphical models. 6) Create a high level language that offers to designers sufficient design flexibility. 7) Reduce time and cost.

1.2 Contribution

Reo IBD Model Fig.5 is a graphical representation of the railway architecture of the CBTC system using "Reo IBD". Reo IBD is a mix of Reo and IBD where the IBD connectors are translated into Reo circuits. Five main components are used to model the CBTC system (MCU, OBD, xrouter, CtrVelocity, and EmgcyBrake). In Fig.5a, two FIFO channels (m1, m2) are used to illustrate the asynchronous communication OBD and MCU. Fig.5b and Fig.5c present the internal behaviour of OBD and MCU. Three filters channels (ComputeVlma and update) are used to apply functions on the data flowing at their ports. Therefore, the filter channel "ComputeVlma" returns the speed value, responding to the message sent by ProdVmaz, and the filter channel "Update" updates the data of the message sent by Coverage. To model the routing replication of data to CtrVelocity or EmgcyBrake, the xrouter component (white circle with a cross) is used. Once data enters the xrouter component, it is sent either to CtrVelocity or to EmergcyBrake but never to both. Merger, replicator, and sync components are used to model synchronization properties. Table 1 highlights the OCL on Reo IBD, it highlights the consistency between the abstract model in Fig.5a and the concrete models in Fig.5b and Fig.5c. For further information on the behavior of Reo components, please refer to [1].

"Reo IBD" tries to create a better and simpler understanding of functions of the system by explicitly expressing their components' behavior. It uses a graphic representation and is often easier to understand and comprehend than a purely textual description. The IBD represents the properties of a system block and the information flow between the elements; while Reo allows to formally specify how, when, and upon which conditions data may flow from the input to the output ports of a system block. In other words, Reo orchestrates the compatibility, correctness, and reliability of the components within the system. Thus, "Reo IBD" is a win-win solution to reduce cost and gain time while providing a wholesome model for a cyber-physical system.

References

1. Arbab, F.: Reo: a channel-based coordination model for component composition. Mathematical Structures in Computer Science **14**(3), 329–366 (2004). https://doi.org/10.1017/S0960129504004153

2. Kunz, G., Perondi, E., Machado, J.: Modeling and simulating the controller behavior of an automated people mover using iec 61850 communication requirements. In: 2011 9th IEEE International Conference on Industrial Informatics. pp. 603–608. IEEE (2011)
3. Lion, B., Chouali, S., Arbab, F.: Compiling protocols to promela and verifying their ltl properties. In: MODELS Workshops. pp. 31–39 (2018)