Integrating SysML and Timed Reo to Model and Verify Cyber-Physical Systems Interactions with Timing Constraints

Perla Tannoury[1,2] and Ahmed Hammad[1,3]

[1] University of Marie and Louis Pasteur, France
FEMTO-ST Institute - UMR CNRS 6174
[2] perla.tannoury@femto-st.fr
[3] ahammad@femto-st.fr

Abstract. Modeling Cyber-Physical Systems (CPS) with timing constraints is challenging due to the complexity of their component behaviors. We propose `Timed SysReo`, a novel modeling language that extends SysML with Timed Reo to capture CPS architecture and timed interactions. Timed SysReo uses Timed Reo Internal Block Diagrams (Timed Reo IBD) and Timed SysReo Sequence Diagrams (TSRSD) to detail component interactions and message flows. Since direct formal verification of requirements in TSRSD is impractical, we automate the transformation of TSRSD into Timed Constraint Automata (TCA) using ATL rules. Requirements are expressed in Timed Scheduled-Data-Stream Logic (TSDSL), enhancing precision and rigor in CPS verification. We illustrate the efficacy of our approach through a case study involving a Smart Medical Bed (SMB).

Keywords: CPS · SysML · Timed Reo · Timed SysReo · TCA · ATL.

1 Introduction

Cyber-physical systems (CPSs) are composed of software and physical components that constantly interact with one another [15]. These systems find application across various domains such as smart cities, healthcare, and autonomous vehicles [6, 26, 22]. Efficient modeling of CPSs is crucial for understanding and verifying functional properties, particularly in critical domains. However, modeling CPS structures and interactions with timing constraints adds complexity and challenges, risking significant errors, especially in vital sectors like emergency response and healthcare.

Several languages and formalisms are used in CPS modeling [18, 7, 11]. In our study, we opted for the System Modeling Language (SysML) [21] due to its capability to model heterogeneous systems, integrating both software and hardware components. SysML enhances stakeholder understanding by modeling system architecture, behavior, and requirements. In SysML, interactions between components are modeled using Internal Block Diagram (IBD) and Sequence

Diagram (SD). However, these interactions adopt an `endogenous` coordination approach which complicates design and reduces `reusability`, increasing project costs. Moreover, while SysML proficiently models CPS, it may not fully address formal specification and verification of intricate component interactions with timing constraints.

Alternatively, the `exogenous` coordination approach, which embeds coordination logic within connectors, is gaining traction among researchers [3, 14, 17]. This method enhances reusability and simplifies design. Timed Reo, as employed by scholars in [4, 16], offers a circuit-like graphical representation and enables modeling interactions with timing constraints. It supports reusability, maintains synchrony during composition, and provides a formal representation through Timed Constraint Automata (TCA). However, Timed Reo is focused only on coordination making it difficult to model the architecture and hierarchical structure of CPS.

To our knowledge, no comprehensive study has integrated SysML and Timed Reo for CPS modelization. Previous research has focused on either SysML [13, 28] or Timed Reo [4, 16] separately. In our recent work, we introduced `SysReo` [23–25], combining SysML and Reo to enhance CPS validation. While `SysReo` effectively models intricate CPS requirements, structure, behavior, and interactions, it falls short in handling CPS with timing constraints.

In this paper, we first introduce `Timed SysReo` by extending SysML with Timed Reo, introducing the `Timed Reo Internal Block Diagram (Timed Reo IBD)` and `Timed SysReo Sequence Diagram (TSRSD)`. Then ,we propose an approach for verifying interactions and interoperability among CPS components, relying on meta-modeling [9] and transformations [8]. This involves defining meta-models for source and target models and establishing correspondences between them. To ensure accuracy and prevent errors, we develop automated ATL rules [1] for seamless TSRSD to TCA transformation. TCAs represent CPS component behavior and coordination, facilitating verification of Timed Scheduled-Data-Stream Logic (TSDSL) requirements.

Finally, we illustrate our approach using a Smart Medical Bed (SMB) case study, showcasing the benefits of Timed SysReo and formal verification for developing robust medical CPS systems.

The paper follows this structure: Section 2 provides an overview of SysML, Reo, Timed Reo, TCA, and TSDSL. Section 3 outlines the proposed modeling approach using Timed SysReo models. The transformation of TSRSD into TCA using ATL is given in Section 4. Section 5 presents the SMB system case study. Finally, Section 6 discusses related work, and Section 7 concludes the paper with directions for future research.

2 Background

This section provides a concise overview of SysML, Reo, Timed Reo, Timed Constraint Automata (TCA) and Timed Scheduled Data Stream Logic (TSDSL).

2.1 SysML

SysML [12] is a general-purpose modeling language derived from UML [19] and tailored for systems engineering. It offers nine diagrams that are presented in **Fig. 1** that allow designers to model requirements, structure, and behavior of CPS at different abstraction levels while keeping traceability between these views. In this work, we rely on four SysML diagrams to model CPS: the requirement diagram, the block definition diagram, the internal block diagram, and the sequence diagram. We illustrate the roles of the pre-mentioned diagarms using a simplified example inspired by a Communication-Based Train Control (CBTC) system.

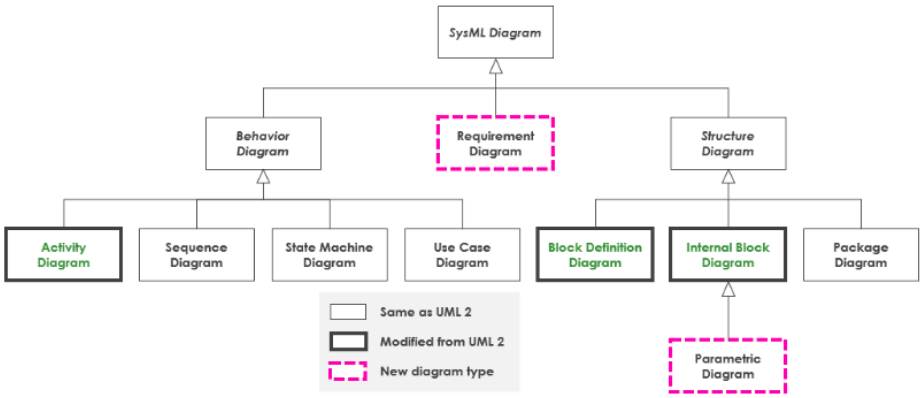


Fig. 1. The nine types of SysML diagrams [27].

A CBTC system is an automotive CPS responsible for ensuring continuous communication between train equipment and trackside control units to guarantee safe train movement. In this simplified example, the system consists of two main components: the On-Board Device (OBD), installed on the train, and the Movement Control Unit (MCU), located trackside. The OBD periodically requests coverage information from the MCU to determine whether the train remains within a safe operational zone.

Requirement Diagram (RD). The Requirement Diagram captures functional and non-functional system requirements and enables traceability between requirements and the model elements that satisfy them. It can be presented as a diagram or as a table. Each requirement is represented as a «requirement» element, and relationships such as «deriveReqt», containment, «satisfy», and «verify» can be established. **Fig. 2** shows the RD of the CBTC example. The top-level requirement $R1$ states that the train must remain continuously visible to the control infrastructure. This requirement is refined into $R1.1$, which specifies deadlock-free interaction between OBD and MCU. This is further refined into two requirements: $R1.11$, stating that the MCU must respond to OBD re-

quests, and $R1.12$, stating that the MCU must provide the requested service. These refined requirements are linked to the blocks that satisfy them (OBD and MCU), ensuring traceability between requirements and structure.

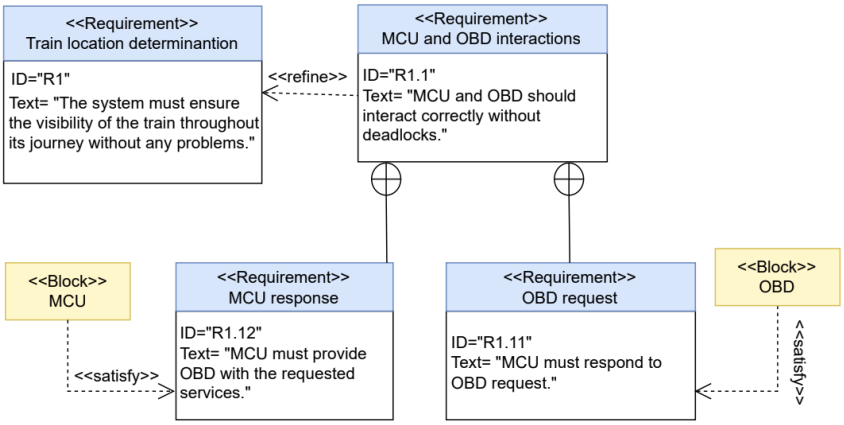


Fig. 2. Requirement Diagram (RD) of the CBTC example.

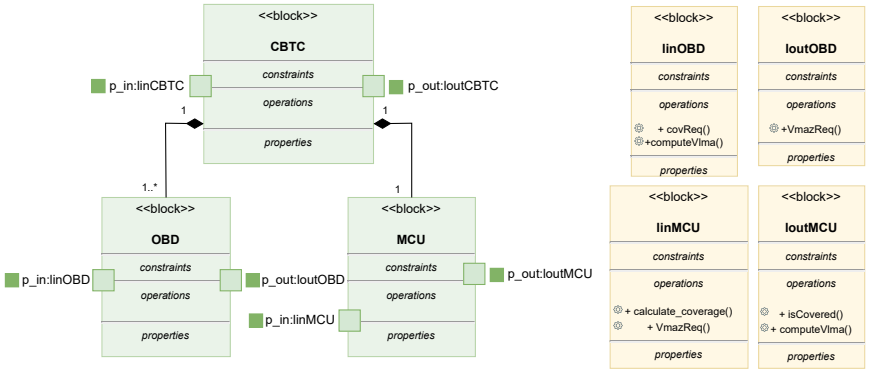


Fig. 3. Block Definition Diagram (BDD) of the CBTC example.

Block Definition Diagram (BDD). The BDD describes the system architecture in terms of blocks and their relationships. Blocks may represent software or hardware entities and can contain properties, operations, constraints, and ports. In **Fig. 3**, the main block *CBTC* is composed of two blocks: *OBD* and *MCU*. The black diamond indicates a composition relationship, meaning that OBD and MCU are integral parts of the CBTC system. Each block exposes

ports that define how it can interact with other blocks through well-defined interfaces.

Internal Block Diagram (IBD). The IBD refines a block by showing its internal parts and the connectors between them. While the BDD shows *what* the system is made of, the IBD shows *how* these parts are interconnected. In **Fig. 4**, the internal structure of the CBTC block is shown. The OBD and MCU appear as part properties inside CBTC. A connector links the output port of OBD to the input port of MCU, representing the communication path used for exchanging information.

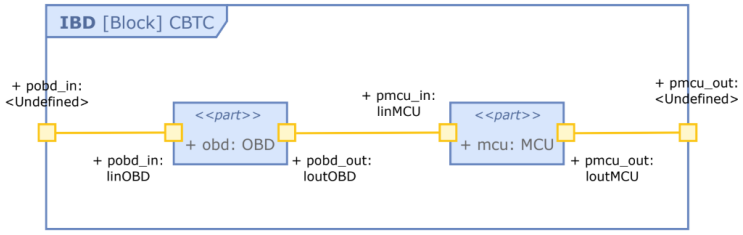


Fig. 4. Internal Block Diagram (IBD) of the CBTC example.

Sequence Diagram (SD). The SD describes system behavior during a particular scenario by showing the temporal ordering of messages exchanged between components. In **Fig. 5**, the SD illustrates a typical interaction scenario. The OBD initiates communication by sending a `covReq()` message to the MCU. Upon receiving this request, the MCU computes the coverage status and replies with an `isCovered()` message. The diagram uses lifelines to represent the components and arrows to represent exchanged messages.

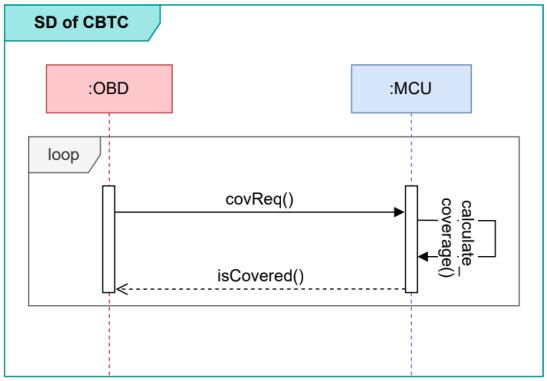


Fig. 5. Sequence Diagram (SD) of the CBTC example.

Together, these diagrams provide complementary views of the system: the RD specifies *what* the system must achieve, the BDD describes *what* the system consists of, the IBD shows *how* components are connected, and the SD explains *how* they interact during execution. However, SysML alone does not formally define the precise synchronization and timing rules of these interactions, which motivates the use of Timed Reo in the next section.

2.2 Reo and Timed Reo

Reo [3] is a channel-based coordination model in which components do not communicate directly. Instead, communication is mediated by *connectors*, which are built by composing primitive channels. A channel connects two or more *ports* and defines constraints on how data may flow between them. By composing channels, a connector specifies the allowed patterns of data transfer between component ports, such as synchronization, buffering, routing, broadcasting, or mutual exclusion. Coordination in Reo is therefore exogenous: the interaction structure is defined independently of the behavior of the connected components. An *interaction* is a single data-flow event over one or more ports that is permitted by the connector. In Timed Reo, an interaction can occur only when the required ports are ready and when both data and timing constraints imposed by the channels are satisfied.

Basic Reo channels. In **Fig.6** we illustrate the different Reo channels. The most common primitive channels include:

Fig. 6. Basic Reo channels.

Sync channel: data flows only when both ports are ready at the same time (strict synchronization).
LossySync channel: behaves like Sync, but discards the data if the receiving port is not ready.
SyncDrain channel: Lack a sink end and possess two source ends. They accept data items through one end only if there's a simultaneous data item available through the other end, with all accepted data being lost.
AsyncDrain channel: Accept data items through source ends but lose them individually, never simultaneously.
FIFO1 channel: contains a buffer that can store exactly one data item, allowing decoupled communication.
Filter channel: allows data to pass only if it satisfies a predicate.

From Reo to Timed Reo. Timed Reo [4] extends Reo by associating timing constraints with channels. This allows specifying not only how data flows, but also when it is allowed to flow. For example, a FIFO channel can be extended into a Timed FIFO where stored data expires after a given time bound.

2.3 Timed Constraint Automata (TCA)

The formal semantics of Timed Reo connectors is given in terms of Timed Constraint Automata (TCA), introduced in [4] as an extension of Constraint Automata with clocks. A TCA extends classical timed automata by labeling transitions not only with clock constraints, but also with sets of ports and data constraints, thereby capturing the joint occurrence of data-flow events and their timing. This makes it possible to model multi-port interactions and to express both synchronous and asynchronous coordination together with data-dependent behavior.

A TCA is defined as:

$$A = (L, L_0, N, \rightarrow, C, ic)$$

where L is a set of locations, L_0 the initial location, N a set of ports, C a set of clocks, ic assigns an invariant to each location, and $\rightarrow$ is the transition relation.

A transition

$$(l, P, dc, cc, R, l')$$

means that the ports $P \subseteq N$ participate in an interaction, the data constraint dc holds, the clock constraint cc is satisfied, and clocks in R are reset.

Data Constraints (DC). A data constraint is a logical predicate over the data values observed at the ports participating in the interaction. For example, $d_X = d_Y$ requires the same data value at ports X and Y.

Clock Constraints (CC). A clock constraint is a conjunction of inequalities over clocks (e.g., $t \leq 1$) restricting when a transition may occur.

Semantics of TCA (Runs). The behavior of a TCA is described by its runs. All clocks start at value 0. A run is an alternating sequence of time elapses and transitions:

$$(l_0, v_0) \xrightarrow{\delta} (l_0, v_0 + \delta) \xrightarrow{tr} (l_1, v_1) \cdots$$

Time may elapse only while the invariant of the current location holds. A transition occurs when its port set, data constraint, and clock constraint are satisfied.

TCA do not define accepted runs in the language-recognition sense. Every maximal run that respects invariants and constraints represents a valid behavior of the connector.

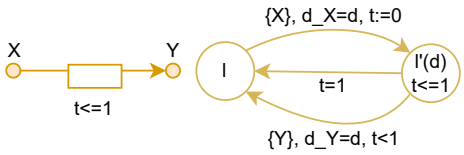


Fig. 7. Timed Reo circuit and TCA for an expiring timed FIFO channel.

2.4 Example: Expiring Timed FIFO Channel

Fig. 7 shows a Timed FIFO channel connecting input port X to output port Y through a buffer that stores exactly one data element.

When a data value d_X arrives at port X, it is stored in the buffer and clock t is reset. The invariant $t \leq 1$ specifies that the data may remain in the buffer for at most one time unit.

Two transitions are possible:

1. If port Y reads the data before $t = 1$, the interaction involves only port Y (asynchronous read), the data is delivered, and the system returns to the empty state.
2. If $t = 1$ is reached without a read, the invariant would be violated. A transition is forced that discards the data and returns the system to the empty state.

The FIFO holds only one element. Data removal is not arbitrary: it corresponds either to a read interaction at Y or to expiration enforced by the timing invariant.

2.5 Timed Scheduled Data Stream Logic (TSDSL)

Timed Scheduled Data Stream Logic (TSDSL) is a stream-based specification formalism for Timed Reo connectors [4]. It describes system behavior in terms of timed data streams observed at ports. Unlike LTL (Linear Temporal Logic) and its timed variants (such as MITL), which describe properties over a single linear sequence of states or events, TSDSL describes systems through multiple timed event sequences, one for each communication point. The focus is therefore not on how a single execution evolves, but on how different observable event streams relate to each other over time.

For a port X, a timed stream is a sequence of events:

$$\langle (d_i, t_i) \rangle_{i \geq 1}$$

where d_i is the data value and $t_i \in \mathbb{T}$ is its timestamp. A system is interpreted as a family of such streams, one per port.

TSDSL formulas constrain relations between these streams.

Example: Timed FIFO channel (capacity 1, bound 1). Let X be the input stream and Y the output stream. The FIFO behavior is specified by:

$$\Box\Big((d,t) \in X \;\Rightarrow\; \exists (d',t') \in Y\Big)$$

$$(d,t) \in X \wedge (d',t') \in Y \;\Rightarrow\; d = d'$$

$$(d,t) \in X \;\Rightarrow\; \exists (d,t') \in Y \,.\, t \leq t' \leq t+1$$

$$(d_1,t_1),(d_2,t_2) \in X \wedge t_1 < t_2 \;\Rightarrow\; t'_1 < t'_2$$

with the implicit functional matching between input and output events induced by the FIFO discipline. Each input event is matched by a unique output event carrying the same data, delayed by at most one time unit, and preserving order between events.

3 Modeling Approach

In this section, we present our model-based design approach for CPS development using Timed SysReo. We first outline the overall modeling and verification workflow. Next, we introduce the underlying meta-models used to support timed structural and behavioral specification. Then, we compare Timed SysReo to SysReo.

3.1 Approach steps

Fig. 8 illustrates the proposed methodology, which is composed of two complementary processes: a modeling phase and a validation and verification (V&V) phase.

During the modeling phase, CPS designers first capture system requirements, including both functional and timing constraints. These requirements are then progressively refined into three complementary views: (i) requirement models, (ii) structural models, and (iii) interaction and behavioral models. Specifically, structural decomposition is represented using ExtBDD diagrams, while internal architecture and timed coordination between components are captured using Timed Reo IBD. Finally, system-level behavioral coordination is specified using TSRSD, which explicitly incorporate temporal constraints into interaction scenarios.

In the V&V phase, the objective is to formally analyze timed interactions and ensure correct interoperability among CPS components. TSRSD models are systematically transformed into TCA using ATL transformation rules (see Section 4). In parallel, system requirements are formally expressed using TSDSL [4]. The resulting formal artifacts enable rigorous verification of both functional and temporal properties.

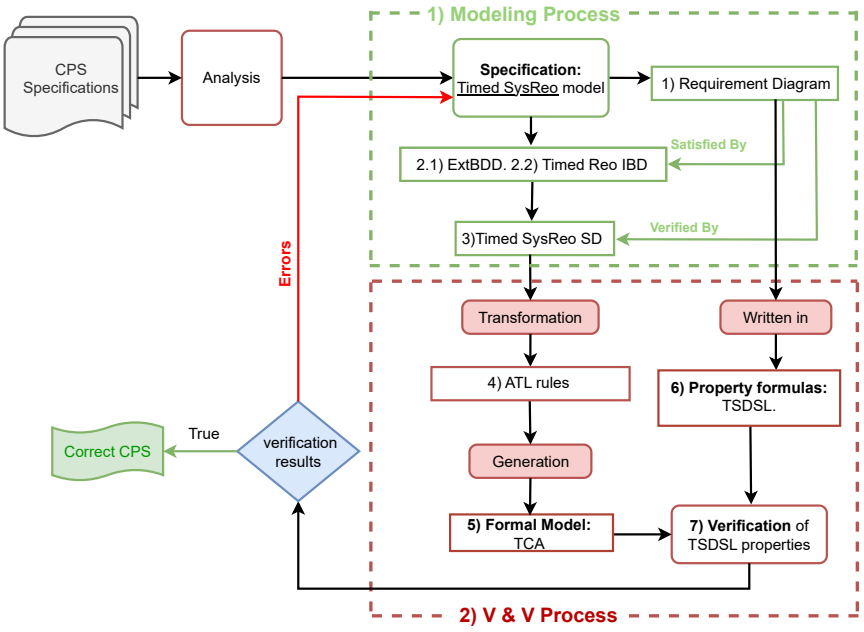


Fig. 8. Modeling and V&V processes using Timed SysReo.

If inconsistencies or violations are detected during verification, the process iterates back to the modeling phase until a consistent and correct system specification is obtained.

3.2 Timed SysReo meta-models

This subsection introduces the meta-models underlying Timed Reo IBD and TSRSD. These meta-models define the structural and behavioral abstractions used in our modeling framework.

Timed Reo IBD meta-model **Fig. 9** shows the meta-model used to describe the internal structure of CPS components and their timed interactions. In this view, a system is decomposed into interacting parts, where each part exposes its communication points through ports. Components do not communicate directly. Instead, all interaction is mediated by Timed Reo connectors, which define how data is coordinated between ports. These connectors are built from channels that determine both the communication pattern and its timing behavior. A key aspect of this model is that it explicitly supports both synchronous and asynchronous communication. Synchronous interaction is captured by channels such as *Sync*, where data transfer only occurs when all involved parties are

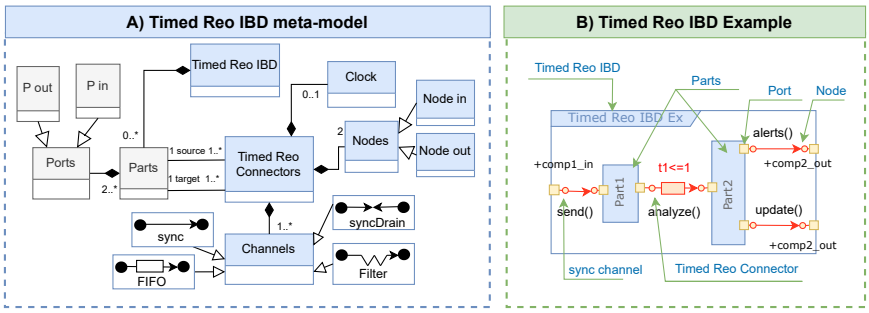


Fig. 9. Meta-model of Timed Reo IBD with an illustrative example.

ready at the same time. In contrast, asynchronous communication is represented through buffering channels, in particular the *FIFO* channel. This channel stores data temporarily, allowing the sender and receiver to operate independently in time. As a result, communication does not require both ends to be active simultaneously. This distinction is important for CPS, where components often run at different speeds or follow independent execution cycles. Timing aspects are further captured using clocks, which allow the specification of delays, timeouts, and other temporal constraints on data flow. Together, these elements make it possible to model asynchronous behavior in a precise and analyzable way at the architectural level.

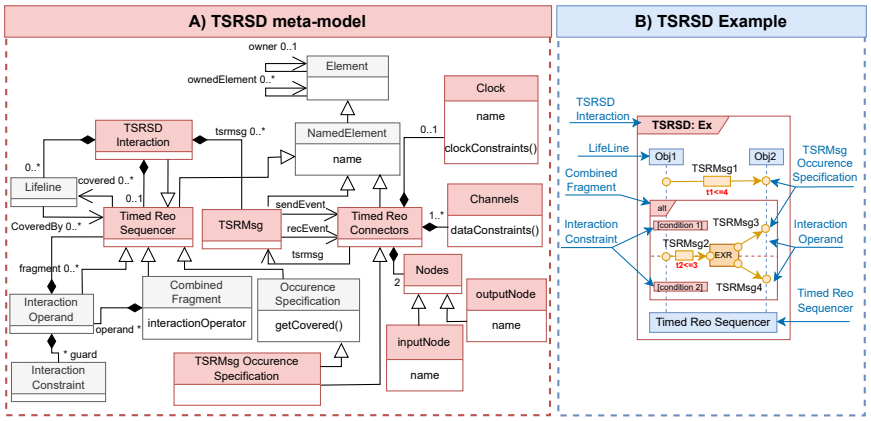


Fig. 10. Meta-model of TSRSD with an illustrative example.

TSRSD meta-model Fig. 10 presents the the TSRSD meta-model followed by an example. It focuses on the behavioral aspect of CPS interactions under timing constraints. It describes interaction scenarios as exchanges of messages between system components, each represented by a lifeline. Asynchronous communication is explicitly reflected in the way messages are modeled. A message is not treated as an instantaneous action. Instead, it is represented through separate send and receive event occurrences. This means that when a message is sent, it may be received later, depending on timing constraints and system behavior. This separation naturally captures asynchronous interaction, since communication does not require sender and receiver to be synchronized in time. The delay between sending and receiving is constrained using timing information and is consistent with the buffering behavior defined at the architectural level in Timed Reo connectors, such as FIFO channels. The overall execution of interactions is managed by the Timed Reo Sequencer, which defines the ordering of events and supports constructs such as alternatives, loops, and guarded interactions. Finally, to support formal verification, TSRSD models are transformed into Timed Constraint Automata (TCA). During this transformation, send and receive events become distinct transitions, while buffering behavior is encoded through state structure and clock constraints. This ensures that asynchronous communication is preserved in the verification model rather than being simplified away. All elements inherit from NamedElement to ensure traceability across structural and behavioral views of the system.

3.3 Comparison with SysReo

Compared to SysReo [24, 25], Timed SysReo provides a more comprehensive modeling framework for CPS by explicitly integrating temporal semantics into both structural and behavioral specifications. While SysReo focuses primarily on structural composition and interaction coordination, it does not natively support explicit representation of timing constraints such as delays, timeouts, or temporal ordering at the modeling level. As a result, timing aspects must often be handled indirectly or at the analysis stage. In contrast, Timed SysReo introduces explicit time-aware constructs in both Timed Reo IBD and TSRSD models. This enables unified modeling of structure, behavior, and timing constraints within a single framework, supporting more precise reasoning about real-time interactions and improving the fidelity of CPS specifications. This integration of timing semantics directly at the modeling level supports earlier detection of temporal inconsistencies and provides stronger guarantees during formal verification.

4 Transforming TSRSD into TCA using ATL

In this section, we introduce the Atlas Transformation Language (ATL) and explain how to convert TSRSD diagrams into Timed Constraint Automata (TCA).

4.1 ATL Overview

The *Atlas Transformation Language* (ATL) [1] is a model-to-model transformation language used in Model-Driven Engineering (MDE) to automatically generate target models from source models defined over different meta-models. ATL follows a *declarative rule-based* approach in which transformation rules describe how elements conforming to a source meta-model are matched and how corresponding elements conforming to a target meta-model are produced.

A general ATL matched rule has the following structure:

```
rule RuleName {
  from
    s : SourceMM!SourceElement (condition)
  to
    t : TargetMM!TargetElement (
        feature1 <- s.attribute1,
        feature2 <- thisModule.helperFunction(s)
    )
}
```

The `from` part specifies the type of source element to be matched and optional guard conditions. The `to` part defines how attributes and references of the target element are initialized using values, expressions, or helper functions derived from the source element.

4.2 Timed Constraint Automata Meta-Model

Based on the formal definition of Timed Constraint Automata (TCA), we propose a meta-model depicted in **Fig. 11**. The main classes are: Timed Constraint Automata, the root class containing states and transitions; State, which includes a name, type (initial or not), and an invariance condition for the maximum duration; and Transition, which specifies source and target states, node names (input/output), data constraints, and clock constraints for firing time.

4.3 ATL Transformation Rules

The transformation from TSRSD to TCA is driven by a systematic correspondence between behavioral interaction concepts and automata constructs. In particular:

1. Each TSRMsgOccurrenceSpecification on a lifeline is transformed into a State in TCA.
2. Each TSRMsg between two lifelines is transformed into a Transition.
3. Lifelines determine the set of node names associated with transitions.
4. Timed Reo connectors attached to messages determine the data and clock constraints of transitions.

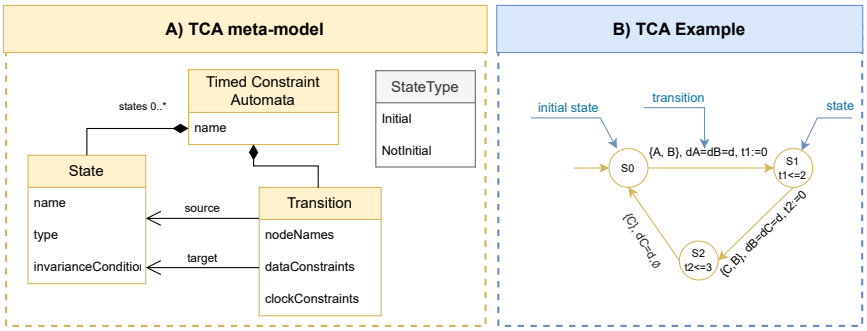


Fig. 11. Timed Constraint Automata meta-Model followed by an example.

5. The ordering of occurrence specifications along lifelines determines the source and target states of transitions.

Thus, ATL rules follow a consistent pattern: interaction events become states, and time-constrained messages become transitions connecting those states.

Example Rule: TSRMsg2Transition converts TSRMsg instances from TSRSD to Transition instances in TCA:

```
rule TSRMsg2Transition {
    from
        tsrMsg: TSRSD!TSRMsg,
        mos: TSRSD!TSRMsgOccurrenceSpecification
    to
        t: TCA!Transition (
            nodeNames <- {tsrMsg.inputNode.name, tsrMsg.outputNode.name},
            dataConstraints <- tsrMsg.channelType,
            clockConstraints <- tsrMsg.clock.name.concat(':=0'),
            source <- thisModule.resolveTemp(mos, 's'),
            target <- thisModule.resolveTemp(thisModule
            .NextTSRMsgOccSpec(mos.getCovered(), mos), 's')
        )
    if
        not tsrMsg.outputNode.isLast
}
```

This rule maps TSRMsg to Transition, indicating message transmission, sets node names, data constraints, and initializes clock constraints. It defines source and target states based on TSRMsg specifications, using a helper function to find subsequent occurrences on a lifeline.

4.4 Verification workflow from TSRSD to property checking

Once TSRSD is transformed into TCA using ATL rules, the resulting automaton serves as the formal semantic representation of the modeled CPS interactions. Verification is then performed in three steps:

1. System requirements are formalized using TSDSL formulas.

2. Each TSDSL formula is translated into its corresponding Büchi automaton.
3. The Büchi automaton is composed with the generated TCA, and the emptiness of the resulting automaton is checked following Arbab's method which is detailed in [4]. It ensures that a given TCA satisfies a specified TSDSL formula. It involves transforming the TSDSL formula into its negation, constructing a Büchi TCA for this negation, merging it with the original TCA, and checking the emptiness of the resulting combined TCA.

If the composed automaton is empty, the requirement is satisfied. Otherwise, the counterexample trace indicates a violation in the TSRSD model, guiding refinement. In the following section (Section 5) we illustrate this workflow on the Smart Medical Bed case study.

5 Smart Medical Bed Case Study

In this section, we present our case study on the Smart Medical Bed (SMB) system. We provide a brief overview of the SMB, gather relevant information for analysis, and use Timed SysReo models to specify requirements, design structure, and model behavior while accounting for timing constraints. The system is validated by generating TCAs from TSRSD using predefined ATL rules, nabling verification of timed properties using Timed Constraint Automata with Buchi acceptance conditions, following Arbab's method [5, 4].

5.1 SMB overview

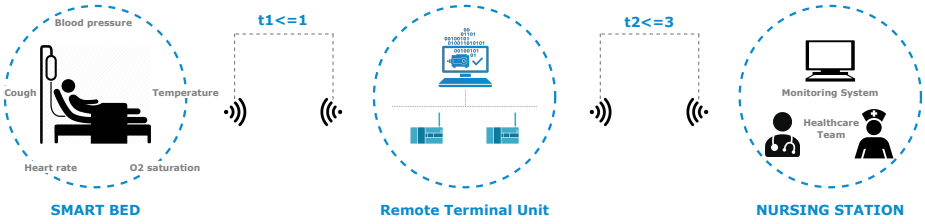


Fig. 12. Smart medical bed architecture.

A SMB system integrates sensors to monitor vital signs like temperature, transmitting data to a Remote Terminal Unit (RTU) within 1 Time Unit (TU), as shown in **Fig.12**. The RTU manages data flow between the Smart Bed (SB) and Nursing Station (NS), deciding whether to update patient information or alert healthcare teams within 3 TUs. The SMB system consists of the SB, RTU, and NS components.

Our study focuses on modeling and validating the SB-RTU interaction within the SMB. Using Timed SysReo, we analyze requirements and model architecture.

We employ Timed Reo IBD and Timed SysReo SD to represent internal structure and manage timing constraints, enhancing efficiency in modeling complex components and interaction protocols within the SMB environment.

5.2 Modeling process using Timed SysReo

Requirements During the modeling process, it is vital to ensure system functionality and usability. This starts with identifying specific system needs, outlining functional requirements of the SMB system. For instance, requirement R1 in **Table.1**, emphasizes the necessity for the SB to send the collected patient vital signs data (e.g., heart rate, blood pressure, temperature, and oxygen saturation) to the RTU within a latency of no more than 1 TU. This requirement is satisfied by the SB component and verified by the `sendTempData()` message.

Table 1. Requirement table of SMB.

ID	Requirement Description	Satisfied by	Verified By
R1	The SB shall send the collected patient vital signs data to RTU within a latency of no more than 1 TU .	SB.	sendTempData().
R2	The RTU must send an `emergencyAlert()` message to the NS within 3 TU for abnormal data or transmit an `updatePatientInfo()` message within the same time-frame for normal data.	RTU.	emergencyAlerts(), updatePatientInfo().

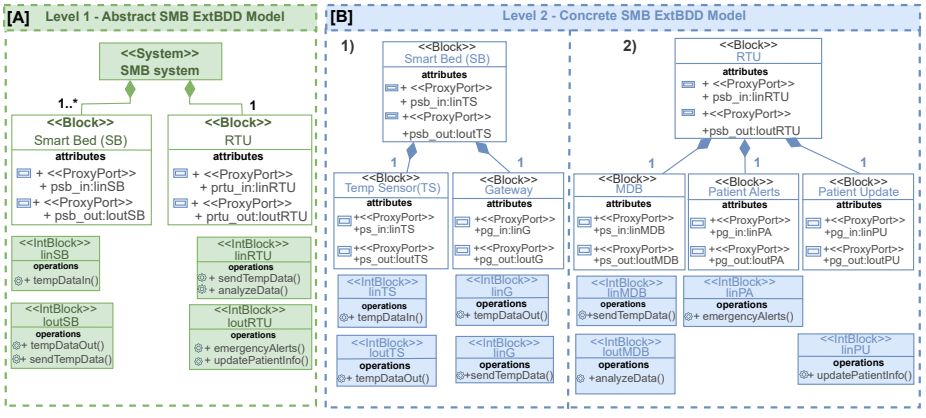
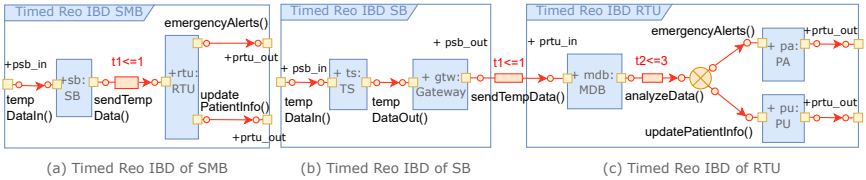


Fig. 13. The ExtBDD model of the SMB system.

Extended Block Definition Diagram (ExtBDD) The ExtBDD diagram displays the hierarchical structure of the SMB system, with each component from the requirement diagram (Table.1) depicted as a block. These blocks detail a component's internal operations, offered and required services, with input and output proxy ports. In **Fig. 13 [A]**, we show an abstract overview of the system, featuring main components like `SMB`, divided into `SB` and `RTU`. The elements labeled linSB, loutSB, linRTU, loutRTU, denote the internal input/output ports of the SB and RTU compoennts respectively. These ports make explicit how data enters and exits each CPS part through Timed Reo channels and clarify how components are connected to the coordination layer.

In **Fig. 13 [B]**, the concrete level illustrates sub-components within primary components. For instance, the smart bed includes `Temperature Sensor` and `Gateway`, with the former responsible for data measurement and transmission to the latter.



(a) Timed Reo IBD of SMB (b) Timed Reo IBD of SB (c) Timed Reo IBD of RTU

Fig. 14. Timed Reo IBD of the SMB system.

Timed Reo Internal Block Diagram (Timed Reo IBD) **Fig.14** presents the Timed Reo IBD of the smart medical bed system. The purpose of Timed Reo IBD is to enhance the modeling of system architecture by replacing the SysML Internal Block Diagram (IBD) with Timed Reo connectors, allowing for the precise specification and analysis of timing properties and synchronization patterns within the system.

In **Fig. 14(b,c)**, internal structures of SB and RTU components are illustrated, including their timed interaction protocols. For instance, the gateway (GTW) component sends `sendTempData()` to the medical database (MDB) component through a timed FIFO channel, constrained by *t1<=1 TU*. This channel imposes a deadline, *t1<=1*, where data availability follows a structured pattern: each data item in the FIFO buffer has a maximum duration of *t1<=1* before removal. Similarly, the MDB sends `analyzedData()` to the Exclusive router (EXR) via a timed FIFO channel with a deadline of *t2<=3 TU*. Following that, the EXR determines whether to dispatch `emergencyAlerts()` via a sync channel for immediate transmission to the patient alert (PA) component, or to send `updatePatientInfo()` via a sync channel directly to the patient update (PU) component.

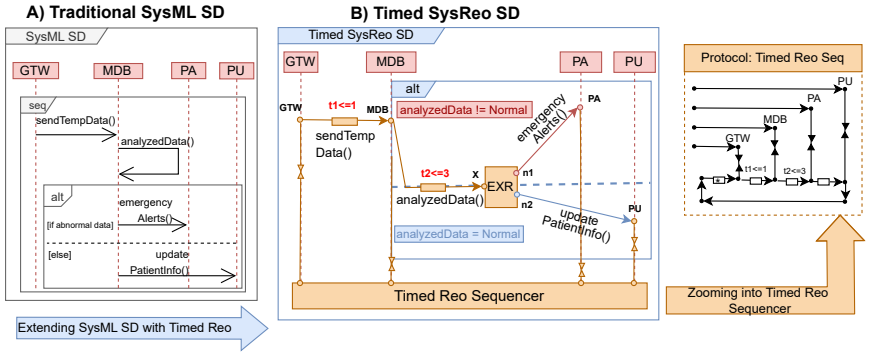


Fig. 15. Difference between traditional SysML SD and TSRSD of the SMB system.

Timed SysReo Sequence Diagram (TSRSD) **Fig. 15** presents the TSRSD of the SMB scenario. This diagram is a concrete instance of the TSRSD metamodel introduced in Section 3 and captures the timed interaction specific to the Smart Bed–RTU communication.

In this scenario, the Gateway (GTW) sends the `sendTempData()` message to the Medical Database (MDB). Unlike a traditional SysML sequence diagram where this message represents a direct call between components, TSRSD associates the message with a Timed Reo connector that explicitly specifies the coordination protocol between the ports GTW, MDB. The connector is a timed FIFO channel constrained by the clock guard ($t1 \leq 1$), expressing that the data produced by GTW must be consumed by MDB within 1 TU.

The orange circles in **Fig. 15** represent Timed Reo ports, while the Timed Reo sequencer defines how messages are coordinated independently from the internal implementation of the components. This separation makes the interaction protocol explicit and modifiable without altering component behavior.

Most importantly, the TSRSD is not only descriptive: it is the direct input for the verification phase. Using the ATL transformation rules presented in Section 4, each TSRSD element is automatically translated into a corresponding element of a TCA. Messages become TCA transitions labeled with port sets and data constraints, and Timed Reo connectors introduce the associated clock constraints.

For example, requirement R1 in **Table. 1** is captured here by the timed FIFO channel that constrains the `sendTempData()` interaction with ($t1 \leq 1$). This TSRSD is therefore the bridge between the SysML-level behavioral model and the formal TCA model used for verification.

5.3 Validation and Verification Process

By applying ATL rules to the TSRSD of SMB, we generate the corresponding TCA shown in **Fig. 16**. ATL rules systematically transform TSRSD elements

into TCA elements. Occurrence specifications are mapped to TCA states, TSR messages to transitions labeled with port sets and data constraints, and Timed Reo connectors introduce the corresponding clock constraints. For example, in **Fig. 15 B**, the `sendTempData()` message in TSRSD is translated into a transition `t` using the ATL rule `TSRMsg2Transition` from Section 4.3. This transition `t` involves nodes {GTW, MDB}, with data constraint $d_{GTW} = d_{MDB} = d$ and clock constraint t1:=0. Applying ATL rule to TSRSD facilitates the creation of its corresponding TCA, depicted in **Fig. 16**.

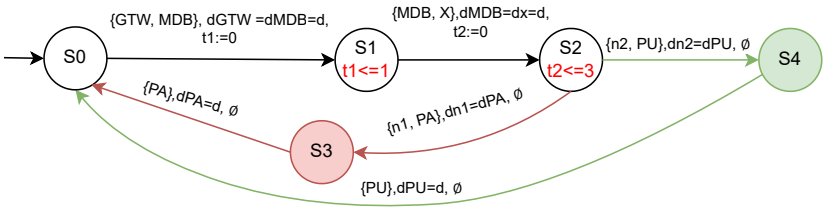


Fig. 16. The generated TCA from the smart medical bed TSRSD.

Verification of TSDSL Properties on TCA To ensure the correctness of the SMB system's behavior and coordination, we express predefined requirements using TSDSL [4], which integrates temporal logic with timing constraints. These TSDSL specifications are verified against the Büchi automata [10] of the TCA using Arbab's method [5, 4]. TSDSL allows expressing temporal and data-dependent requirements over Timed Reo executions. It can specify ordering of events, timing bounds, and data observations on ports, making it suitable for formalizing CPS interaction requirements. Requirement R1 is expressed from the perspective of the SB–RTU interaction. The formula states that whenever SB emits data, RTU must receive it within 1 TU. The case of "no observable transmission" corresponds to the absence of data emission from SB, rather than to any subsequent communication between RTU and NS :

$$\Box\,[[SB]]\,(\langle\,\langle\,RTU\,\rangle^{\leq 1}\,\rangle) \vee \neg\langle\,\langle\,.\,\rangle^{\leq 1}\,\rangle$$

Checking the emptiness of the combined TCA involves determining if there exists a run (sequence of states and transitions) that satisfies the negated formula. For requirement R1, this means verifying if there is no sequence where data transmission from SB occurs within 1 time unit without either a corresponding reception at RTU within the same time or no transmission at all within the subsequent 1 time unit. If the combined TCA is empty, meaning no such run exists, then the original TCA satisfies the TSDSL formula for requirement R1. This verification step ensures that the CPS model adheres to the specified timing and temporal logic requirements, thus confirming its correctness and reliability.

Decidability and Verification Complexity A central theoretical question in this verification approach is whether TSDSL properties can actually be decided when evaluated over the TCA generated from TSRSD models.

The method follows the work in [4]. In short, TSDSL specifications are first converted into Büchi automata. These are then combined with the TCA that represents the system's behavior. The resulting product automaton is checked for emptiness, which tells us whether the property holds. This process is decidable because of how TCA are structured. Although they extend standard automata with clocks to model time, they still rely on a finite region abstraction, similar to classical timed automata. This abstraction effectively turns the infinite number of possible clock valuations into a finite number of equivalence classes, which makes automated verification possible. Because of this, emptiness checking for the resulting Büchi TCA remains a decidable problem. However, the practical cost of verification can still be high. Like other timed automata-based methods, the complexity grows exponentially with the number of clocks and interacting components. In larger systems, this can easily lead to state-space explosion. To address this, Timed SysReo limits the analysis to only those TCA generated from the interaction scenarios described in TSRSD models, rather than the entire Timed SysReo system. This keeps the resulting automata smaller and more focused, which makes verification more manageable.

5.4 Applicability to Other CPS Domains

While the Smart Medical Bed is used as the running example in this paper, the proposed Timed SysReo approach is not limited to healthcare applications. It is designed to support the modeling and verification of a broader class of CPS that rely on timed coordination and communication between distributed components.

The separation between structural modeling using Timed Reo IBD and behavioral modeling using TSRSD, together with the formal analysis based on Timed Constraint Automata, makes the approach suitable for systems where components interact both synchronously and asynchronously under timing constraints. For instance, in smart healthcare environments, the same modeling principles can be applied to patient monitoring systems or ICU infrastructures, where sensor data is continuously exchanged with strict timing requirements. In automotive systems, it can be used to model interactions between sensors, controllers, and actuators, where safety-critical messages such as braking or warning signals must be delivered within bounded time. Similarly, in IoT-based applications, the approach naturally fits systems composed of distributed devices that communicate asynchronously and operate at different speeds.

6 Related Work

Research on CPS modeling and verification spans several communities, including systems engineering, formal methods, and software architecture. Existing

approaches typically emphasize either architectural modeling, timing specification, or formal verification, but rarely address all three concerns in an integrated manner. This section reviews the main lines of work related to Timed SysReo and highlights the gap that motivates our contribution.

6.1 SysML-based modeling with timing annotations

SysML is widely used in systems engineering to describe CPS requirements, structure, and behavior in a unified way [20]. To incorporate timing aspects, several works rely on the MARTE profile and CCSL constraints. For example, Huang et al. [13] combine SysML, MARTE, and CCSL to specify temporal properties and enable schedulability analysis. Mallet [18] discusses how MARTE/CCSL can be used to model logical and physical time in CPS. Other works propose SysML-based compositional verification techniques for safety-critical systems [28].

These approaches preserve traceability between requirements and system architecture while allowing designers to express temporal constraints. However, interactions in SysML Sequence Diagrams remain modeled as direct calls between components, without an explicit coordination semantics. In CPS, it is essential to adopt an *exogenous* coordination perspective, where the interaction protocol is separated from component behavior and modeled explicitly, enabling clearer reasoning about data flow, buffering, and timing, and facilitating formal verification.

6.2 Timed automata for CPS verification

Timed automata [2] constitute one of the most established formalisms for reasoning about real-time behavior and have been widely adopted in CPS verification due to their precise operational semantics and decidability properties. They provide a rigorous framework for expressing timing constraints and checking safety and liveness properties through well-supported verification tools.

Nevertheless, timed automata operate at a low level of abstraction that does not directly correspond to architectural system models. In practice, system structure, communication patterns, and interaction scenarios described in engineering models must be manually translated into automata, which may break traceability between design artifacts and formal verification models. For CPS, where architecture, coordination, and timing are strongly coupled, this gap complicates the systematic integration of modeling and verification activities.

6.3 Reo and Timed Reo for coordination

Reo introduces an exogenous coordination model in which components interact through connectors composed of channels [3]. Timed Reo extends this model with timing constraints and provides a formal semantics based on Timed Constraint Automata (TCA) [4]. Kokash et al. [16] show how Timed Reo networks can be translated into networks of timed automata for verification.

Reo is particularly effective at modeling synchronous, asynchronous, buffered, and data-dependent interactions. Nevertheless, it does not offer constructs for representing system architecture, requirements, or hierarchical decomposition, which limits its direct applicability in systems engineering processes.

6.4 Bridging SysML and formal coordination models

Several works aim to bridge SysML with formal verification techniques. The authors in [7] combine SysML with Modelica and formal requirement languages to enable CPS analysis. In [14], the authors discuss documenting component and connector views using UML to improve architectural reasoning.

Our previous work on SysReo [24, 25] combined SysML and Reo to analyze structural compatibility between CPS components. However, SysReo did not incorporate explicit timing constraints nor provide automatic transformation into formal models for temporal verification.

6.5 Positioning of Timed SysReo

The reviewed literature shows a clear separation between the main concerns involved in CPS modeling and verification. SysML-based approaches are well suited for describing system architecture and expressing timing information, yet they do not provide an explicit formal semantics for coordination between components. Timed automata, on the other hand, offer strong formal verification capabilities, but they operate at a low level of abstraction and do not support architectural modeling. Conversely, Reo and Timed Reo provide a precise formal framework for expressing coordination and interaction patterns, but they do not address the representation of CPS architecture and system requirements.

Timed SysReo is designed to bridge these gaps by integrating SysML architectural modeling with Timed Reo coordination and providing an automatic transformation into TCA for formal verification using TSDSL. This combination enables designers to reason about CPS interactions at the modeling level while maintaining a rigorous formal semantics suitable for verification.

7 Conclusion and Future Work

In this paper, we present a novel approach for modeling and verifying Cyber-Physical Systems (CPS) interactions with timing constraints using Timed SysReo. Our method enhances CPS designs by incorporating precise timing considerations through the integration of Timed Reo into SysML diagrams. This allows for an accurate representation of system requirements, behaviors, and interactions under specified timing constraints. Timed SysReo uses two key diagrams: the Timed Reo Internal Block Diagram (IBD) and the Timed SysReo Sequence Diagram (TSRSD). The Timed Reo IBD focuses on the structural aspects of CPS, offering a static view of the system architecture. In contrast, the TSRSD emphasizes dynamic behaviors and timed interactions, providing

a dynamic perspective on system operations. Together, these diagrams offer a comprehensive depiction of both the static and dynamic elements of CPS with respect to timing constraints. Additionally, we demonstrate the automation of transforming TSRSD diagrams into Timed Constraint Automata (TCA) using ATL, which significantly enhances the precision and ease of verifying CPS requirements specified in TSDSL logic. Our approach is exemplified through a case study involving a Smart Medical Bed (SMB), highlighting the practical application and benefits of Timed SysReo in real-world scenarios.

In our future efforts, we will assess Timed SysReo's performance across diverse sectors like automotive and aerospace CPS, moving beyond its current focus on healthcare. This evaluation aims to determine its suitability and address specific challenges and opportunities in each domain. Additionally, we intend to enhance Timed SysReo by modeling both discrete and continuous CPS behaviors. Currently, the Timed SysReo framework primarily handles discrete behaviors, but our expansion will integrate parametric diagrams and incorporate Reo connectors to model differential equations for continuous dynamics. This approach will enable comprehensive modeling of CPS hardware components, considering physical constraints to ensure robust system designs across different operational scenarios.

References

1. Eclipse ATL. https://eclipse.dev/atl/, accessed: May 02, 2024
2. Alur, R., Dill, D.L.: A theory of timed automata. Theoretical computer science **126**(2), 183–235 (1994)
3. 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
4. Arbab, F., Baier, C., de Boer, F., Rutten, J.: Models and temporal logical specifications for timed component connectors. Software & Systems Modeling **6**, 59–82 (2007)
5. Arbab, F., Baier, C., De Boer, F., Rutten, J.: Models and temporal logics for timed component connectors. In: Proceedings of the Second International Conference on Software Engineering and Formal Methods, 2004. SEFM 2004. pp. 198–207. IEEE (2004)
6. Barroso, S., Bustos, P., Nunez, P.: Towards a cyber-physical system for sustainable and smart building: a use case for optimising water consumption on a smartcampus. Journal of Ambient Intelligence and Humanized Computing **14**(5), 6379–6399 (2023)
7. Bouskela, D., Falcone, A., Garro, A., Jardin, A., Otter, M., Thuy, N., Tundis, A.: Formal requirements modeling for cyber-physical systems engineering: An integrated solution based on form-l and modelica. Requirements Engineering **27**(1), 1–30 (2022)
8. Czarnecki, K., Helsen, S., et al.: Classification of model transformation approaches. In: Proceedings of the 2nd OOPSLA Workshop on Generative Techniques in the Context of the Model Driven Architecture. vol. 45, pp. 1–17. USA (2003)

9. De Lara, J., Vangheluwe, H., Alfonseca, M.: Meta-modelling and graph grammars for multi-paradigm modelling in atom 3. Software & Systems Modeling **3**, 194–209 (2004)
10. Gastin, P., Oddoux, D.: Fast ltl to büchi automata translation. In: Computer Aided Verification: 13th International Conference, CAV 2001 Paris, France, July 18–22, 2001 Proceedings 13. pp. 53–65. Springer (2001)
11. Genius, D., Apvrille, L.: Hierarchical design of cyber-physical systems. In: Modelsward (2023)
12. Hause, M., et al.: The sysml modelling language. In: Fifteenth European Systems Engineering Conference. vol. 9, pp. 1–12 (2006)
13. Huang, P., Jiang, K., Guan, C., Du, D.: Towards modeling cyber-physical systems with sysml/marte/pccsl. In: 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC). vol. 1, pp. 264–269. IEEE (2018)
14. Ivers, J., Clements, P.C., Garlan, D., Nord, R., Schmerl, B., Oviedo-Silva, J.: Documenting component and connector views with uml 2.0 (2004)
15. Kim, K.D., Kumar, P.R.: Cyber–physical systems: A perspective at the centennial. Proceedings of the IEEE **100**(Special Centennial Issue), 1287–1308 (2012). https://doi.org/10.1109/JPROC.2012.2189792
16. Kokash, N., Jaghoori, M.M., Arbab, F.: From timed reo networks to networks of timed automata. Electronic Notes in Theoretical Computer Science **295**, 11–29 (2013)
17. Lau, K.K., Ornaghi, M., Wang, Z.: A software component model and its preliminary formalisation. In: International symposium on formal methods for components and objects. pp. 1–21. Springer (2005)
18. Mallet, F.: Marte/ccsl for modeling cyber-physical systems. Formal Modeling and Verification of Cyber-Physical Systems: 1st International Summer School on Methods and Tools for the Design of Digital Systems, Bremen, Germany, September 2015 pp. 26–49 (2015)
19. Miles, R., Hamilton, K.: Learning UML 2.0: a pragmatic introduction to UML. " O'Reilly Media, Inc." (2006)
20. Object Management Group: Omg systems modeling language (sysml) specification (2023), https://www.omg.org/spec/SysML/
21. OMG: OMG System Modeling Languag. https://www.omg.org/spec/SysML/, accessed: 04-09-2024
22. Panahi, V., Kargahi, M., Faghih, F.: Control performance analysis of automotive cyber-physical systems: A study on efficient formal verification. ACM Transactions on Cyber-Physical Systems (2022)
23. Tannoury, P.: An Incremental Model-Based Design Methodology to Develop CPS with SysML/OCL/Reo. In: Journées du GDR GPL. Vannes, France (Jun 2022), https://hal.science/hal-03893454
24. Tannoury, P., Chouali, S., Hammad, A.: Model driven approach to design an automotive cps with sysreo language. In: Proceedings of the 20th ACM International Symposium on Mobility Management and Wireless Access. pp. 97–104 (2022)
25. Tannoury, P., Chouali, S., Hammad, A.: Joint use of sysml and reo to specify and verify the compatibility of cps components. In: International Conference on Formal Aspects of Component Software. pp. 84–102. Springer (2023)
26. Tartarisco, G., Cicceri, G., Bruschetta, R., Tonacci, A., Campisi, S., Vitabile, S., Cerasa, A., Distefano, S., Pellegrino, A., Modesti, P.A., et al.: An intelligent medical cyber–physical system to support heart valve disease screening and diagnosis. Expert Systems with Applications **238**, 121772 (2024)

27. Visual Paradigm: What is SysML? (nd), https://guides.visual-paradigm.com/what-is-sysml/, accessed: 2026-04-20
28. Xie, J., Tan, W., Yang, Z., Li, S., Xing, L., Huang, Z.: Sysml-based compositional verification and safety analysis for safety-critical cyber-physical systems. Connection Science pp. 1–31 (2021)