Experimenting with Formal Verification and Model-based Development in Railways: the case of UMC and Sparx Enterprise Architect

Davide Basile, Franco Mazzanti, Alessio Ferrari

(FMICS 2023)

Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo"
Consiglio Nazionale delle Ricerche

Formal Methods and Tools Lab

We are hiring!

- The Formal Methods and Tools (FMT) lab of the Institute of Information Science and Technologies (ISTI) of the Italian National Research Council (CNR) offers two temporary positions for research in the field of formal modelling and analysis of critical software systems, in particular but not limited to the railway and service computing domains.

- Contact us:
  - maurice.terbeek@isti.cnr.it
  - davide.basile@isti.cnr.it

Overview

- Formal Methods in Railways, Model-based Development
- Sparx Enterprise Architect, UML Model Checker
- Mapping of Sparx EA and UMC models
- Case Study: RBC2RBC handover
- Conclusion

Introduction

- Formal methods in Railways
- Model-based Software/Systems Development (MBSD)
  - mainly based on the OMG UML Standard
- Integration of Formal Methods into MBSD
- Survey on formal verification of UML SM [1]
  - “counterexamples are rarely mapped back to the original models”
  - “UMC could be used to verify UML models”
- Integration of UMC with Sparx EA

[1] André E´., Liu, S., Liu, Y., Choppy, C., Sun, J., Dong, J.S.: Formalizing UML State Machines for Automated Verification–A Survey. ACM Comput. Surv. (2023).

MBSD, Sparx Enterprise Architect

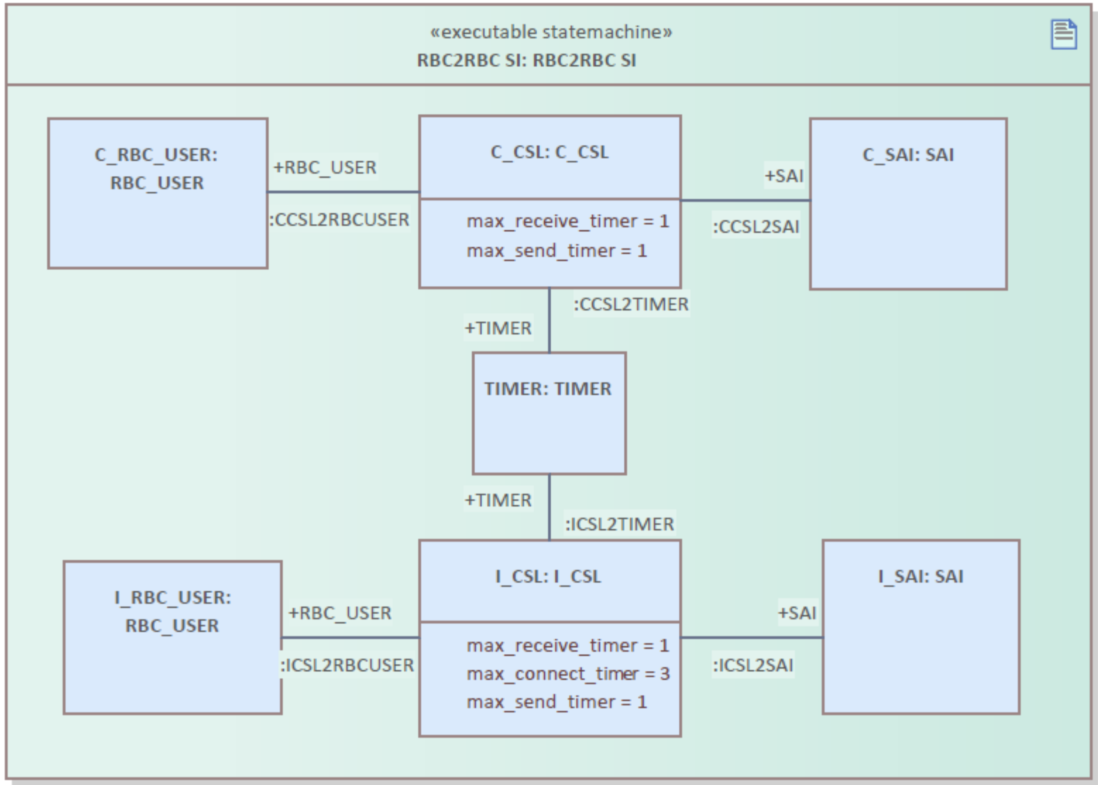
- Development process guided by *models*
- UML: object-oriented paradigm
- State machines: classified behaviour of a class
  - Labels of transitions: `trigger[conditions]/effects`
- Sparx EA: model-based tool based on OMG UML
- Selected within the H2020 Shift2Rail 4SECURail project based on different criteria
  - e.g., composition of state machine

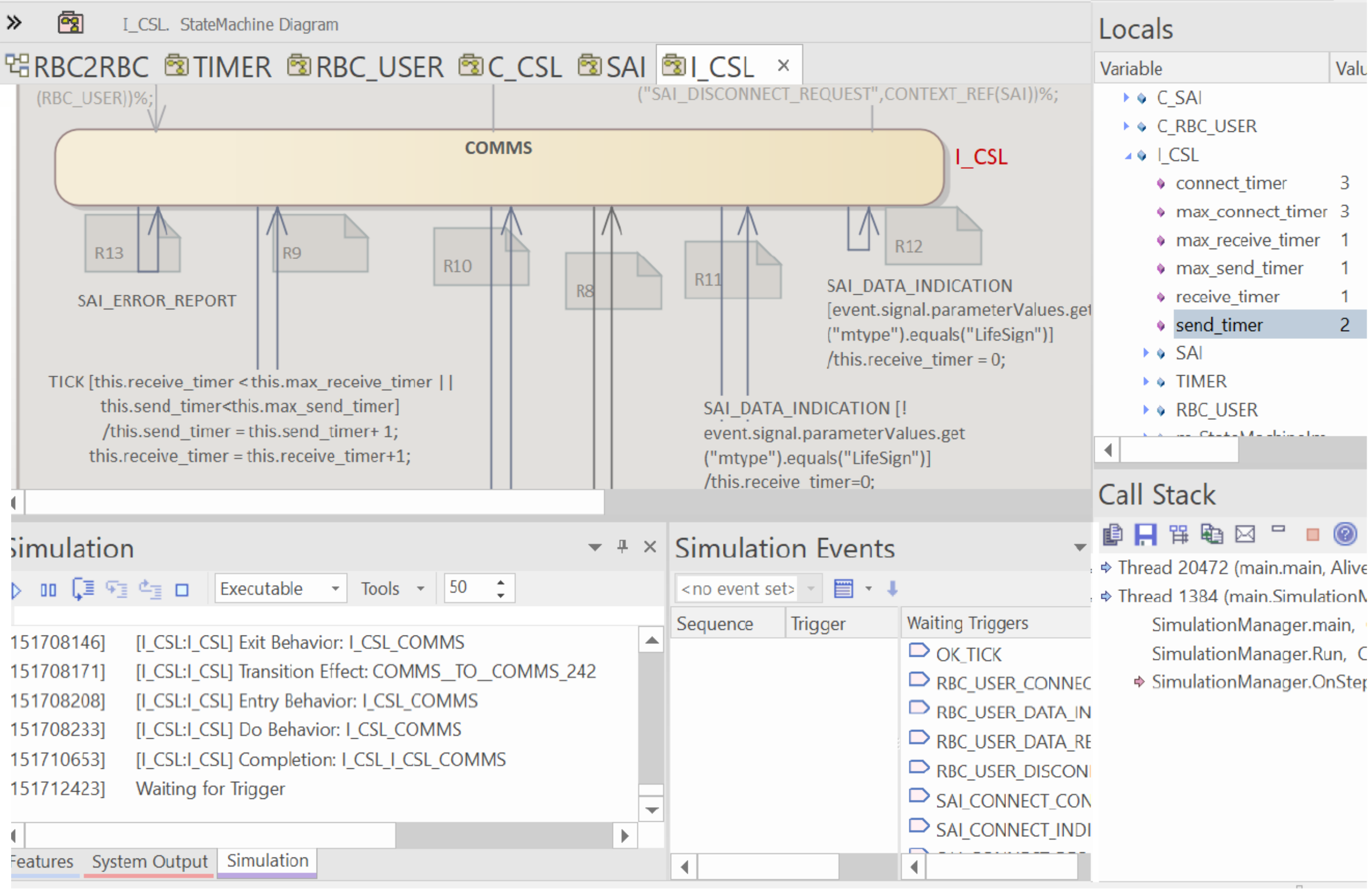
Enterprise Architect

ENTERPRISE ARCHITECT

Executable State Machines

- Executable State Machines
  - Composition of State Machines,
  - Simple instruction for interactions
- Compiled into code for simulation

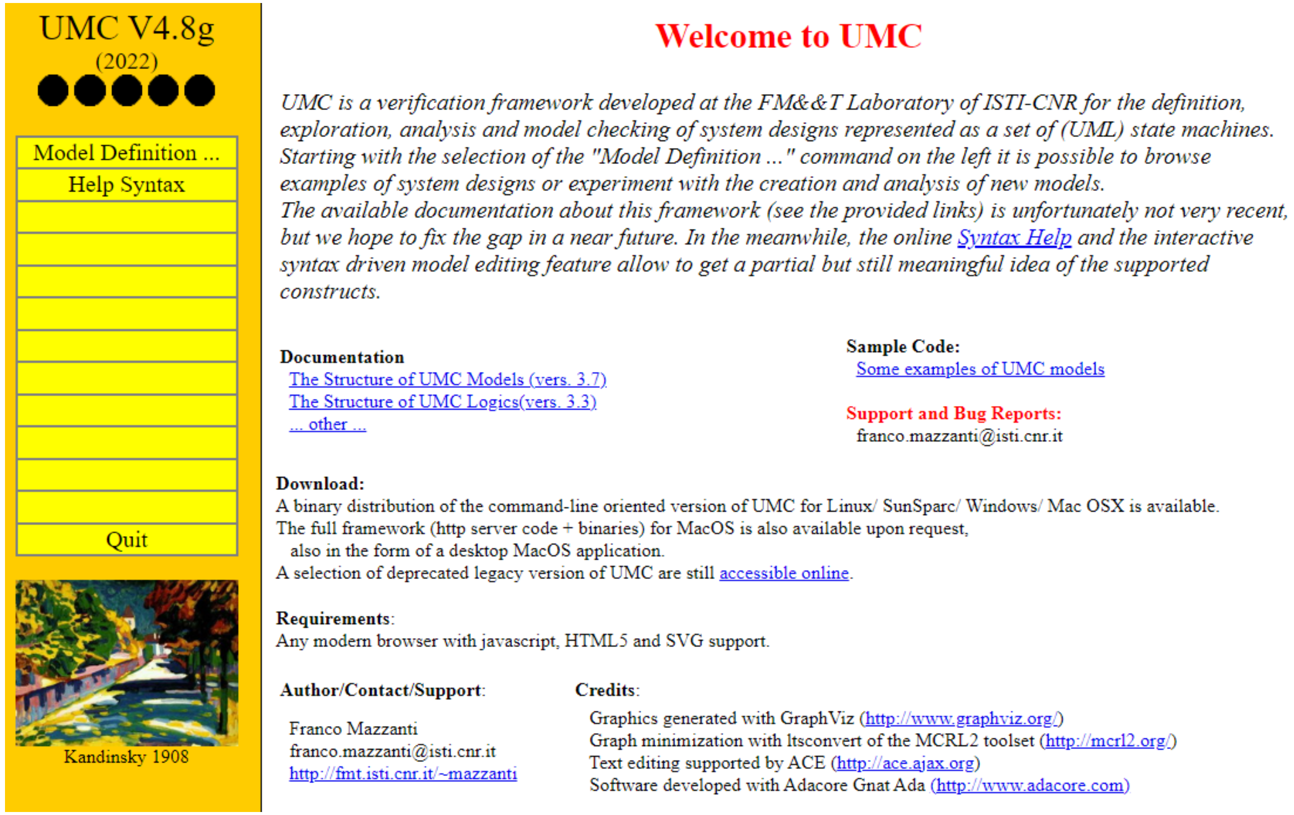




```
151708146]    [I_CSL:I_CSL] Exit Behavior: I_CSL_COMMS
151708171]    [I_CSL:I_CSL] Transition Effect: COMMS__TO__COMMS_242
151708208]    [I_CSL:I_CSL] Entry Behavior: I_CSL_COMMS
151708233]    [I_CSL:I_CSL] Do Behavior: I_CSL_COMMS
151710653]    [I_CSL:I_CSL] Completion: I_CSL_I_CSL_COMMS
151712423]    Waiting for Trigger
```

UML Model Checker (UMC)

- Freely available at https://fmt.isti.cnr.it/umc/V4.8/umc.html
- Currently maintained by Franco Mazzanti
- Oriented towards fast prototyping
- Verification of CTL properties of SM
- On-the-fly model checking [1]
- Automatic translation to [2]:
  - LOTOS NT, ProB
  - Formally verified translation



Welcome to UMC

UMC is a verification framework developed at the FM&&T Laboratory of ISTI-CNR for the definition, exploration, analysis and model checking of system designs represented as a set of (UML) state machines. Starting with the selection of the "Model Definition ..." command on the left it is possible to browse examples of system designs or experiment with the creation and analysis of new models. The available documentation about this framework (see the provided links) is unfortunately not very recent, but we hope to fix the gap in a near future. In the meanwhile, the online Syntax Help and the interactive syntax driven model editing feature allow to get a partial but still meaningful idea of the supported constructs.

Documentation
The Structure of UMC Models (vers. 3.7)
The Structure of UMC Logics(vers. 3.3)
... other ...

Sample Code:
Some examples of UMC models

Support and Bug Reports:
franco.mazzanti@isti.cnr.it

Download:
A binary distribution of the command-line oriented version of UMC for Linux/ SunSparc/ Windows/ Mac OSX is available.
The full framework (http server code + binaries) for MacOS is also available upon request,
also in the form of a desktop MacOS application.
A selection of deprecated legacy version of UMC are still accessible online.

Requirements:
Any modern browser with javascript, HTML5 and SVG support.

Author/Contact/Support:

Franco Mazzanti
franco.mazzanti@isti.cnr.it
http://fmt.isti.cnr.it/~mazzanti

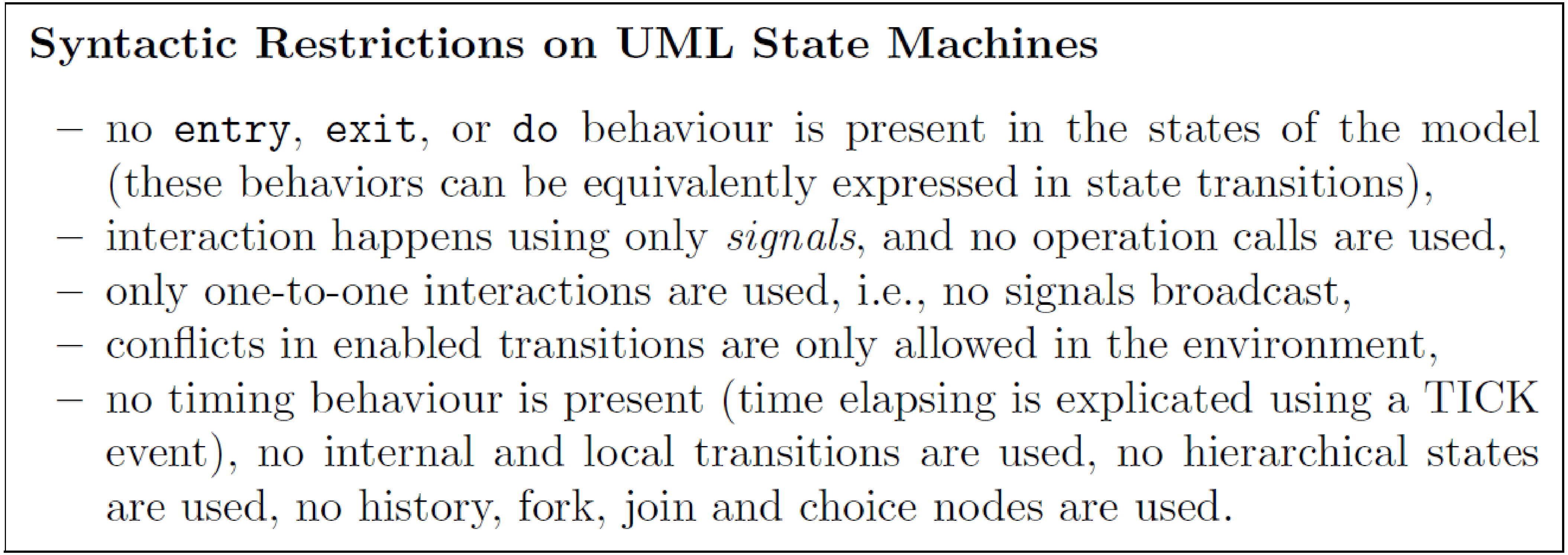
Credits:

Graphics generated with GraphViz (http://www.graphviz.org/)
Graph minimization with ltsconvert of the MCRL2 toolset (http://mcrl2.org/)
Text editing supported by ACE (http://ace.ajax.org)
Software developed with Adacore Gnat Ada (http://www.adacore.com)

[1] F. Mazzanti et al.: A state/event-based model-checking approach for the analysis of abstract system properties. Sci. Comput. Program. 76(2)

[2] F. Mazzanti et al.: Formal Modeling and Initial Analysis of the 4SECURail Case Study. MARS@ETAPS 2022

Bidirectional Approach
UMC SM ⇔ Sparx EA SM



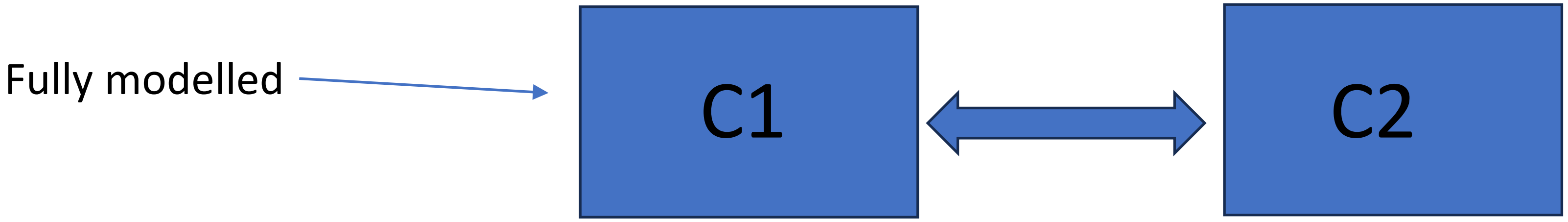
Syntactic Restrictions on UML State Machines

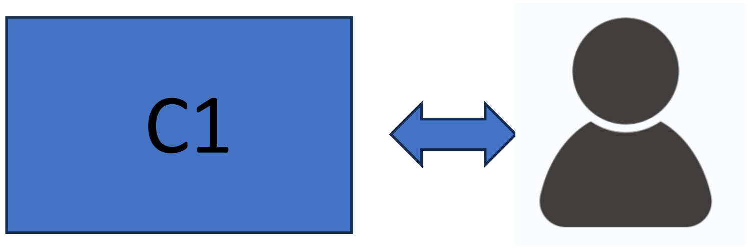
- no `entry`, `exit`, or `do` behaviour is present in the states of the model (these behaviors can be equivalently expressed in state transitions),
- interaction happens using only *signals*, and no operation calls are used,
- only one-to-one interactions are used, i.e., no signals broadcast,
- conflicts in enabled transitions are only allowed in the environment,
- no timing behaviour is present (time elapsing is explicated using a TICK event), no internal and local transitions are used, no hierarchical states are used, no history, fork, join and choice nodes are used.

Semantics correspondence

- Sparx State Machines do not have a formal semantics
- No state-space generation in Sparx EA
- Manual inspection of the engine code of ESM:
  - FIFO order of events
  - Deterministic model (no conflicts in enabled transitions)
  - Fixed scheduling of SM
- Semantics of Sparx EA *included* in the semantics of UMC
- Mapping of traces

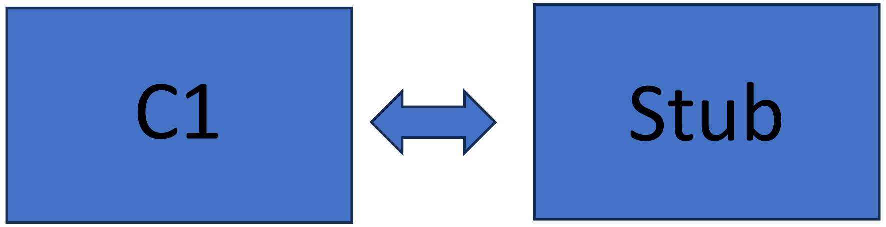
Environment: Interactive Simulation vs Model Checking





Interactive simulations: the human user acts as the environment.

No automation.



Model checking: the environment is explicitly modelled

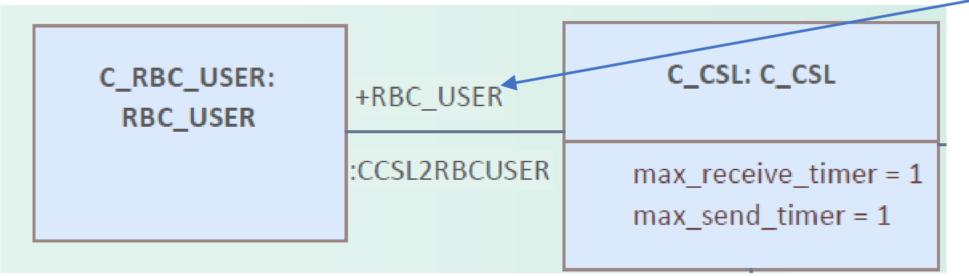
- to obtain a fully closed system on which the verification is automatic.

Rules for relating the model

- classes have a relation “has-a” with other classes,
    - every object has a reference to other objects to whom it is interacting with

```
Object.Signal(value1, value2)
```

```
%SEND_EVENT("TRIGGER.sig(value1,value2)",CONTEXT_REF(RECIPIENT))%
```



Rules for relating the model

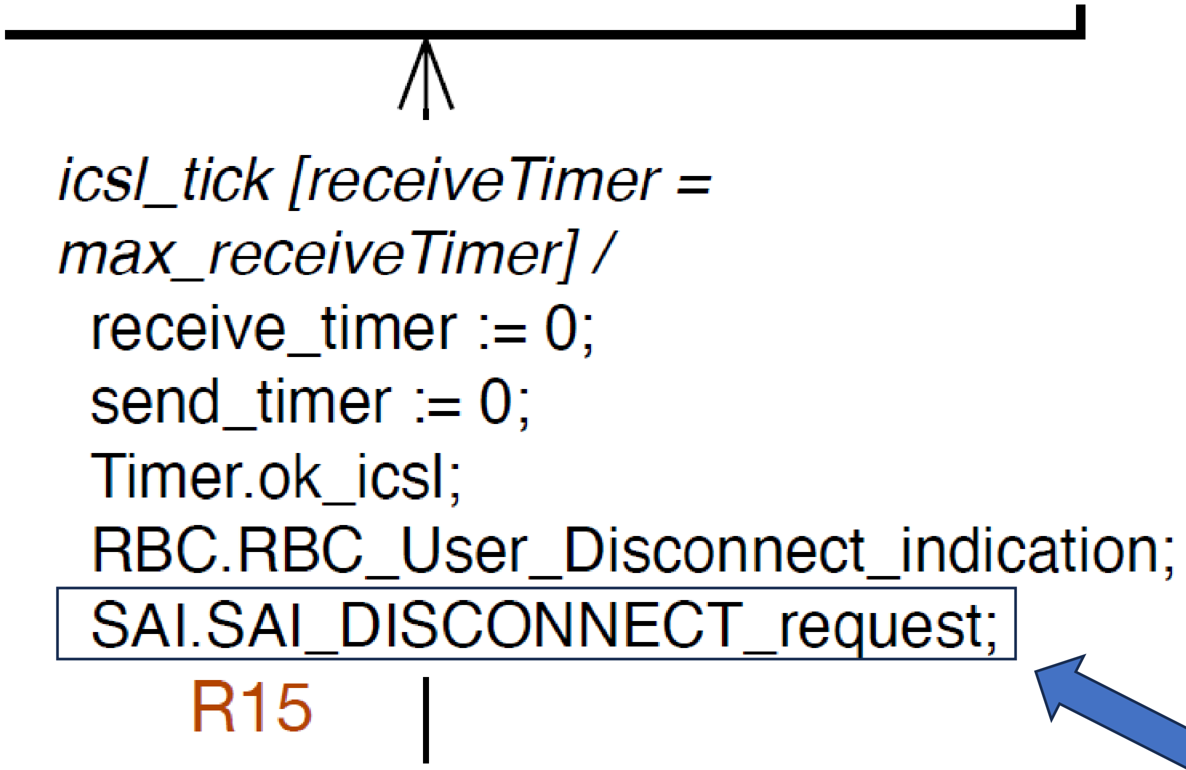
- Signals that are attributes of each class in UMC are in correspondence with global trigger events in the Sparx type Signal and have the same parameters as in UMC.

```
Object.Signal(value1, value2)
```

```
%SEND_EVENT("TRIGGER.sig(value1,value2)",CONTEXT_REF(RECIPIENT))%
```

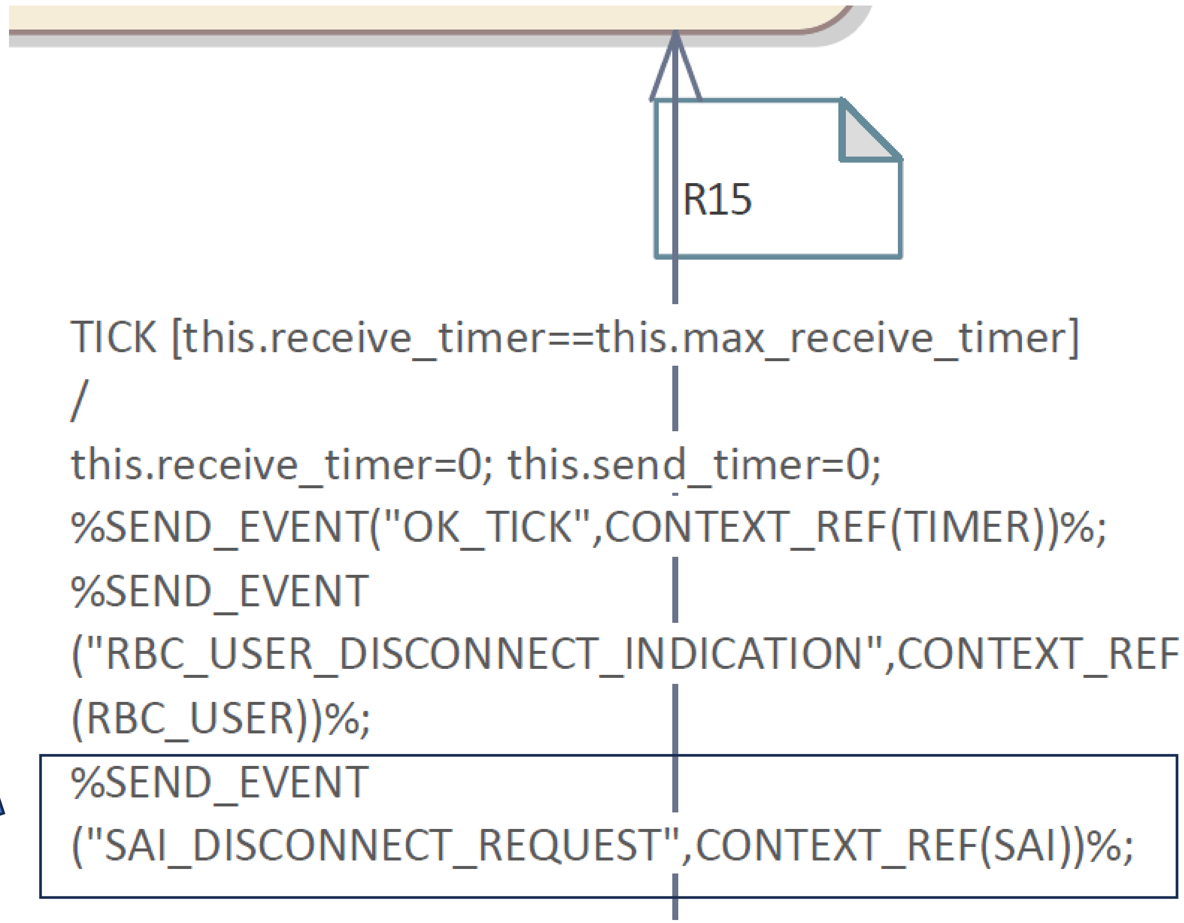
Send/write message

UMC ⟷ Sparx EA



```
icsl_tick [receiveTimer =
max_receiveTimer] /
  receive_timer := 0;
  send_timer := 0;
  Timer.ok_icsl;
  RBC.RBC_User_Disconnect_indication;
  SAI.SAI_DISCONNECT_request;
```

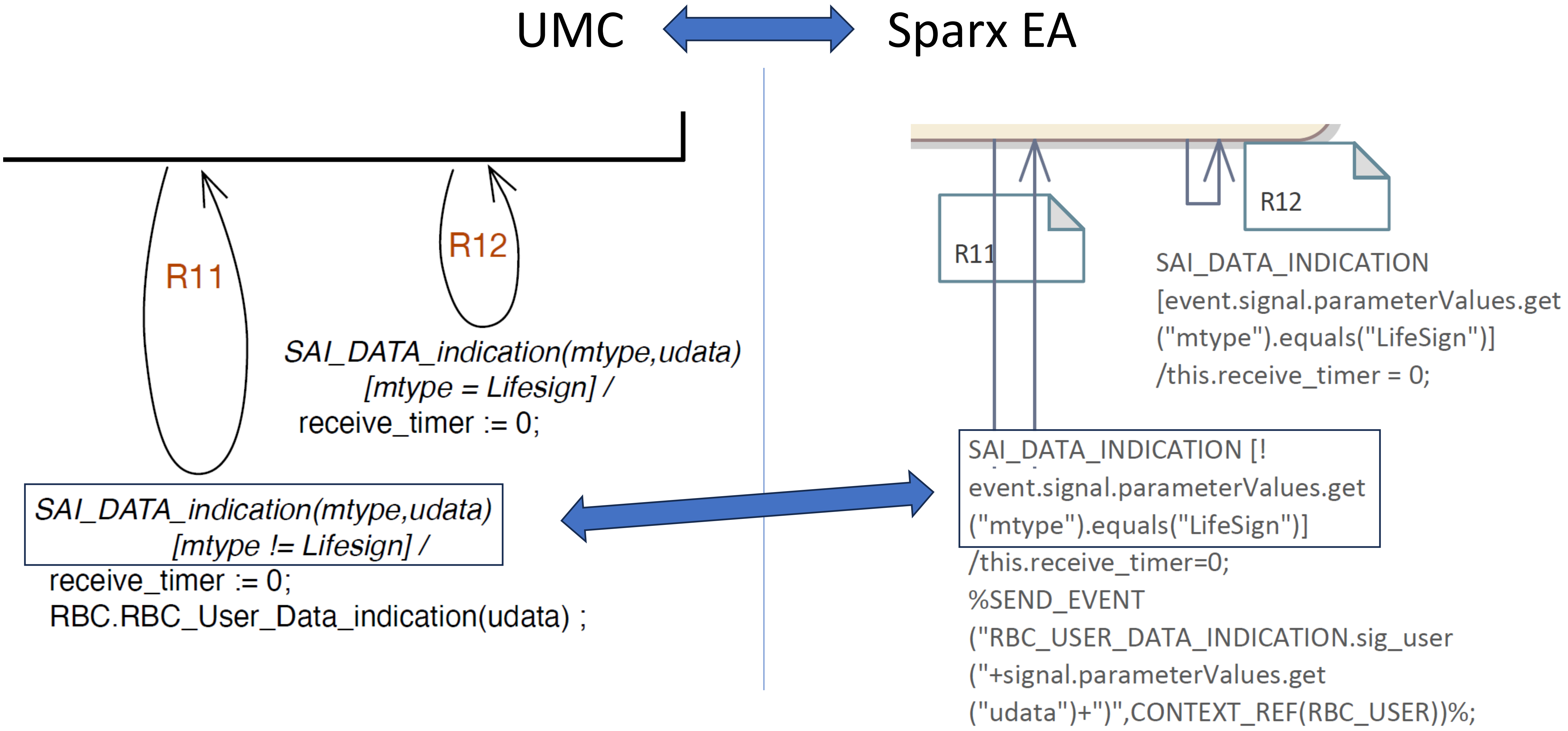
R15



R15

```
TICK [this.receive_timer==this.max_receive_timer]
/
this.receive_timer=0; this.send_timer=0;
%SEND_EVENT("OK_TICK",CONTEXT_REF(TIMER))%;
%SEND_EVENT
("RBC_USER_DISCONNECT_INDICATION",CONTEXT_REF
(RBC_USER))%;
%SEND_EVENT
("SAI_DISCONNECT_REQUEST",CONTEXT_REF(SAI))%;
```

Receive/read message



SAI_DATA_indication(mtype,udata)
[mtype = Lifesign] /
receive_timer := 0;

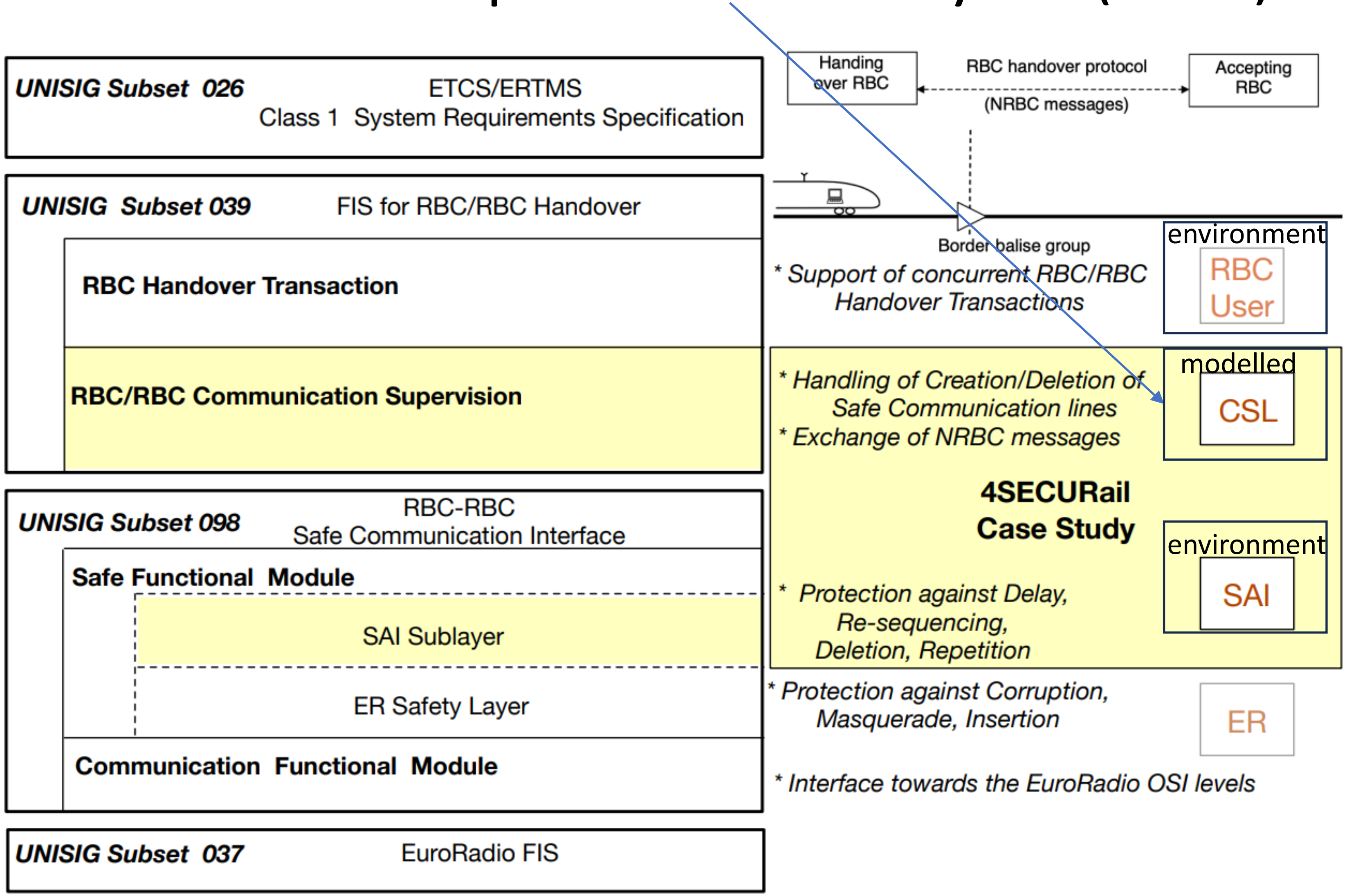
SAI_DATA_indication(mtype,udata)
[mtype != Lifesign] /
receive_timer := 0;
RBC.RBC_User_Data_indication(udata) ;

```
SAI_DATA_INDICATION
[event.signal.parameterValues.get
("mtype").equals("LifeSign")]
/this.receive_timer = 0;
```

```
SAI_DATA_INDICATION [!
event.signal.parameterValues.get
("mtype").equals("LifeSign")]
/this.receive_timer=0;
%SEND_EVENT
("RBC_USER_DATA_INDICATION.sig_user
("+signal.parameterValues.get
("udata")+")",CONTEXT_REF(RBC_USER))%;
```

Case Study: Communication Supervision Layer (CSL)

- RBC/RBC handover protocol (borrowed from the 4SECURail project)
- CSL responsible for:
  - opening/closing a communication
  - maintaining connection through life signs

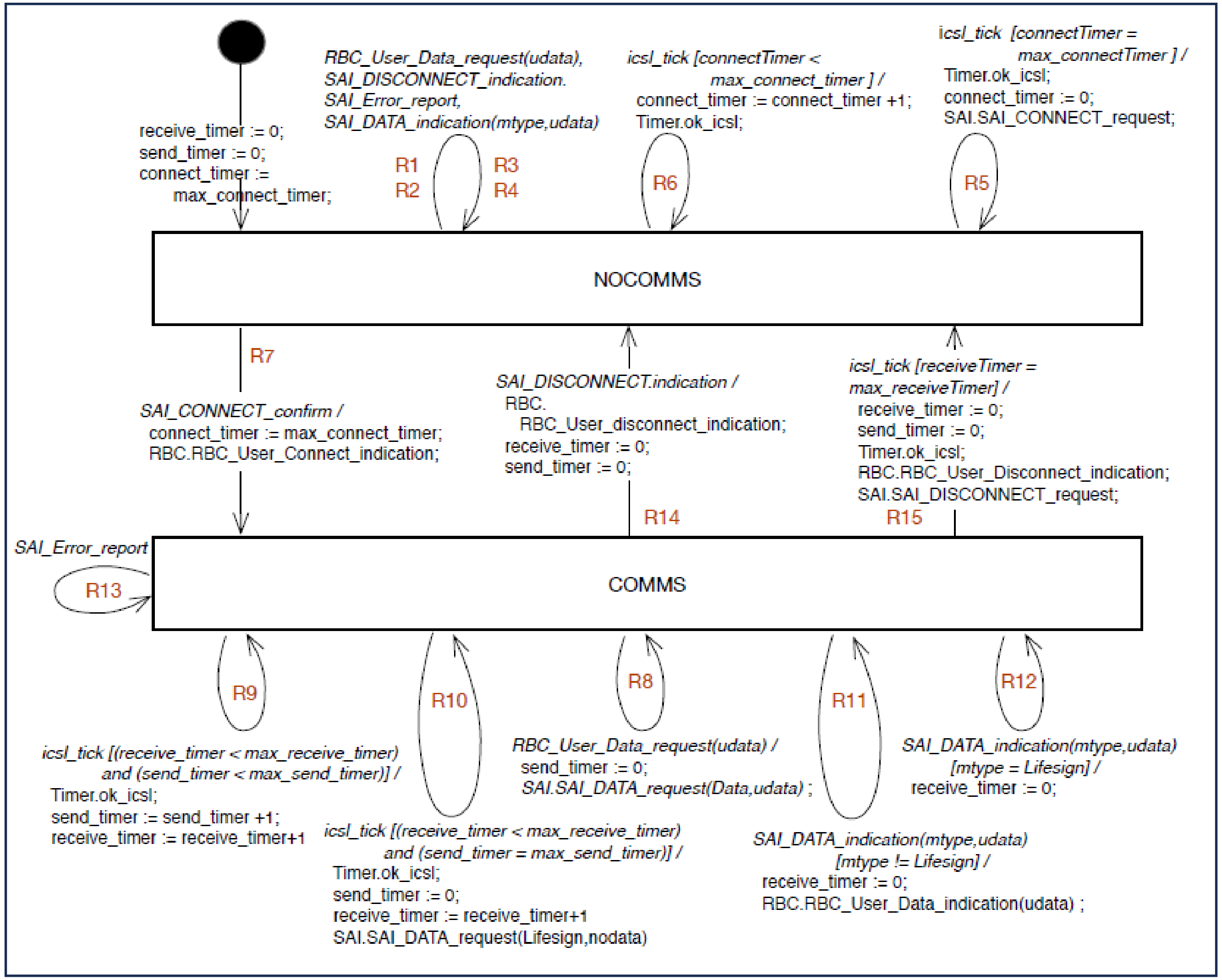

UNISIG Subset 026
ETCS/ERTMS
Class 1 System Requirements Specification
UNISIG Subset 039
FIS for RBC/RBC Handover
RBC Handover Transaction
RBC/RBC Communication Supervision
UNISIG Subset 098
RBC-RBC
Safe Communication Interface
Safe Functional Module
SAI Sublayer
ER Safety Layer
Communication Functional Module
UNISIG Subset 037
EuroRadio FIS
Handing over RBC
RBC handover protocol
(NRBC messages)
Accepting RBC
Border balise group
* Support of concurrent RBC/RBC Handover Transactions
environment
RBC User
modelled
CSL
* Handling of Creation/Deletion of Safe Communication lines
* Exchange of NRBC messages
4SECURail Case Study
environment
SAI
* Protection against Delay, Re-sequencing, Deletion, Repetition
* Protection against Corruption, Masquerade, Insertion
ER
* Interface towards the EuroRadio OSI levels

D. Basile et al.: Formal Analysis of the UNISIG Safety Application Intermediate Sub-layer - Applying Formal Methods to Railway Standard Interfaces. **FMICS 2021**

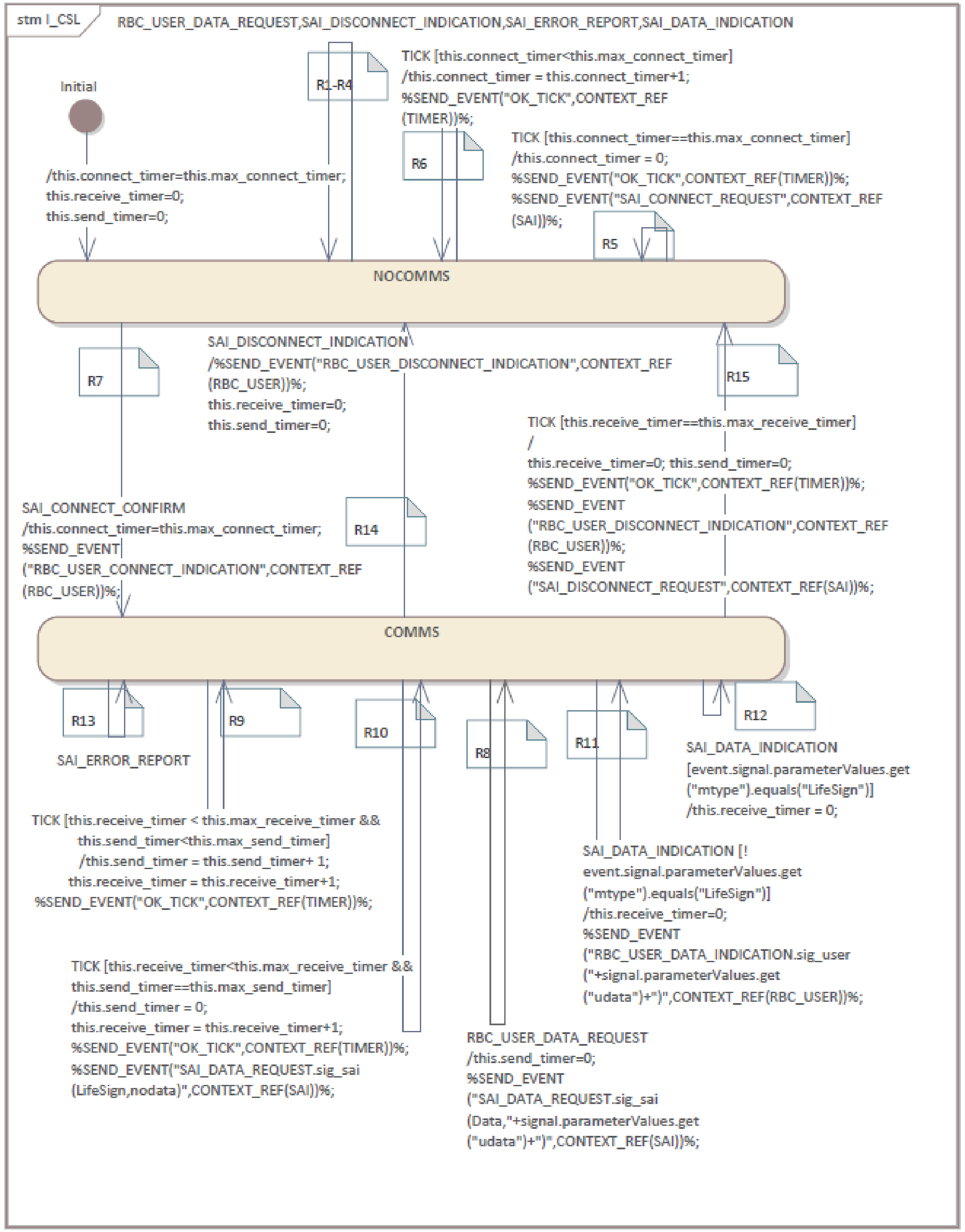
F. Mazzanti et al.: Formal Modeling and Initial Analysis of the 4SECURail Case Study. MARS@ETAPS 2022

F. Mazzanti et al.: A Case Study in Formal Analysis of System Requirements. SEFM Workshops 2022

F. Mazzanti et al.: The 4SECURail Formal Methods Demonstrator. RSSRail 2022

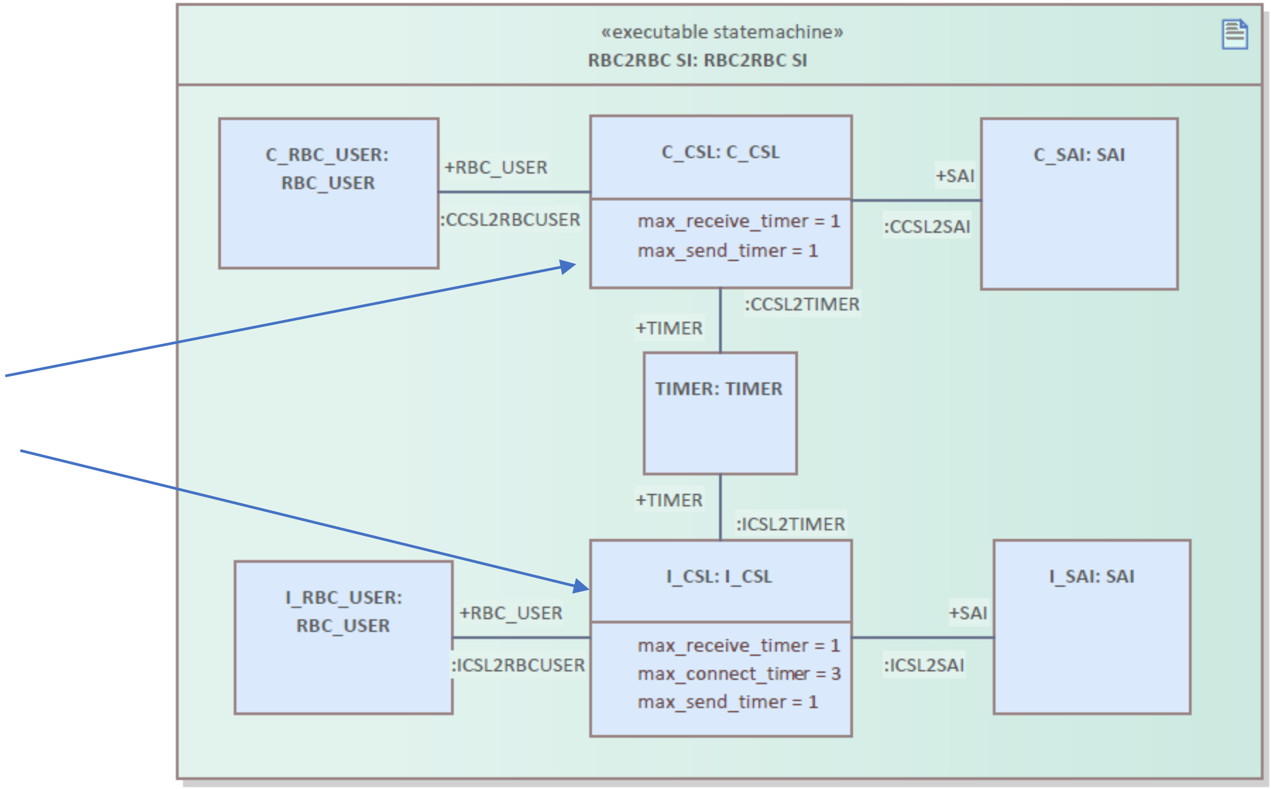

receive_timer := 0;
send_timer := 0;
connect_timer :=
max_connect_timer;
RBC_User_Data_request(udata),
SAI_DISCONNECT_indication.
SAI_Error_report,
SAI_DATA_indication(mtype,udata)
R1
R2
R3
R4
icsl_tick [connectTimer <
max_connect_timer] /
connect_timer := connect_timer +1;
Timer.ok_icsl;
R6
icsl_tick [connectTimer =
max_connectTimer] /
Timer.ok_icsl;
connect_timer := 0;
SAI.SAI_CONNECT_request;
R5
NOCOMMS
R7
SAI_CONNECT_confirm /
connect_timer := max_connect_timer;
RBC.RBC_User_Connect_indication;
SAI_DISCONNECT.indication /
RBC.
RBC_User_disconnect_indication;
receive_timer := 0;
send_timer := 0;
R14
icsl_tick [receiveTimer =
max_receiveTimer] /
receive_timer := 0;
send_timer := 0;
Timer.ok_icsl;
RBC.RBC_User_Disconnect_indication;
SAI.SAI_DISCONNECT_request;
R15
SAI_Error_report
R13
COMMS
R9
R10
R8
R11
R12
icsl_tick [(receive_timer < max_receive_timer)
and (send_timer < max_send_timer)] /
Timer.ok_icsl;
send_timer := send_timer +1;
receive_timer := receive_timer+1
icsl_tick [(receive_timer < max_receive_timer)
and (send_timer = max_send_timer)] /
Timer.ok_icsl;
send_timer := 0;
receive_timer := receive_timer+1
SAI.SAI_DATA_request(Lifesign,nodata)
RBC_User_Data_request(udata) /
send_timer := 0;
SAI.SAI_DATA_request(Data,udata) ;
SAI_DATA_indication(mtype,udata)
[mtype != Lifesign] /
receive_timer := 0;
RBC.RBC_User_Data_indication(udata) ;
SAI_DATA_indication(mtype,udata)
[mtype = Lifesign] /
receive_timer := 0;

UMC ICSL SM


stm I_CSL
RBC_USER_DATA_REQUEST,SAI_DISCONNECT_INDICATION,SAI_ERROR_REPORT,SAI_DATA_INDICATION
Initial
/this.connect_timer=this.max_connect_timer;
this.receive_timer=0;
this.send_timer=0;
R1-R4
TICK [this.connect_timer<this.max_connect_timer]
/this.connect_timer = this.connect_timer+1;
%SEND_EVENT("OK_TICK",CONTEXT_REF
(TIMER))%;
R6
TICK [this.connect_timer==this.max_connect_timer]
/this.connect_timer = 0;
%SEND_EVENT("OK_TICK",CONTEXT_REF(TIMER))%;
%SEND_EVENT("SAI_CONNECT_REQUEST",CONTEXT_REF
(SAI))%;
R5
NOCOMMS
R7
SAI_DISCONNECT_INDICATION
/%SEND_EVENT("RBC_USER_DISCONNECT_INDICATION",CONTEXT_REF
(RBC_USER))%;
this.receive_timer=0;
this.send_timer=0;
R15
TICK [this.receive_timer==this.max_receive_timer]
/
this.receive_timer=0; this.send_timer=0;
%SEND_EVENT("OK_TICK",CONTEXT_REF(TIMER))%;
%SEND_EVENT
("RBC_USER_DISCONNECT_INDICATION",CONTEXT_REF
(RBC_USER))%;
%SEND_EVENT
("SAI_DISCONNECT_REQUEST",CONTEXT_REF(SAI))%;
SAI_CONNECT_CONFIRM
/this.connect_timer=this.max_connect_timer;
%SEND_EVENT
("RBC_USER_CONNECT_INDICATION",CONTEXT_REF
(RBC_USER))%;
R14
COMMS
R13
SAI_ERROR_REPORT
R9
R10
R8
R11
R12
SAI_DATA_INDICATION
[event.signal.parameterValues.get
("mtype").equals("LifeSign")]
/this.receive_timer = 0;
TICK [this.receive_timer < this.max_receive_timer &&
this.send_timer<this.max_send_timer]
/this.send_timer = this.send_timer+ 1;
this.receive_timer = this.receive_timer+1;
%SEND_EVENT("OK_TICK",CONTEXT_REF(TIMER))%;
SAI_DATA_INDICATION [!
event.signal.parameterValues.get
("mtype").equals("LifeSign")]
/this.receive_timer=0;
%SEND_EVENT
("RBC_USER_DATA_INDICATION.sig_user
("+signal.parameterValues.get
("udata")+")",CONTEXT_REF(RBC_USER))%;
TICK [this.receive_timer<this.max_receive_timer &&
this.send_timer==this.max_send_timer]
/this.send_timer = 0;
this.receive_timer = this.receive_timer+1;
%SEND_EVENT("OK_TICK",CONTEXT_REF(TIMER))%;
%SEND_EVENT("SAI_DATA_REQUEST.sig_sai
(LifeSign,nodata)",CONTEXT_REF(SAI))%;
RBC_USER_DATA_REQUEST
/this.send_timer=0;
%SEND_EVENT
("SAI_DATA_REQUEST.sig_sai
(Data,"+signal.parameterValues.get
("udata")+")",CONTEXT_REF(SAI))%;

SPARX EA ICSL SM

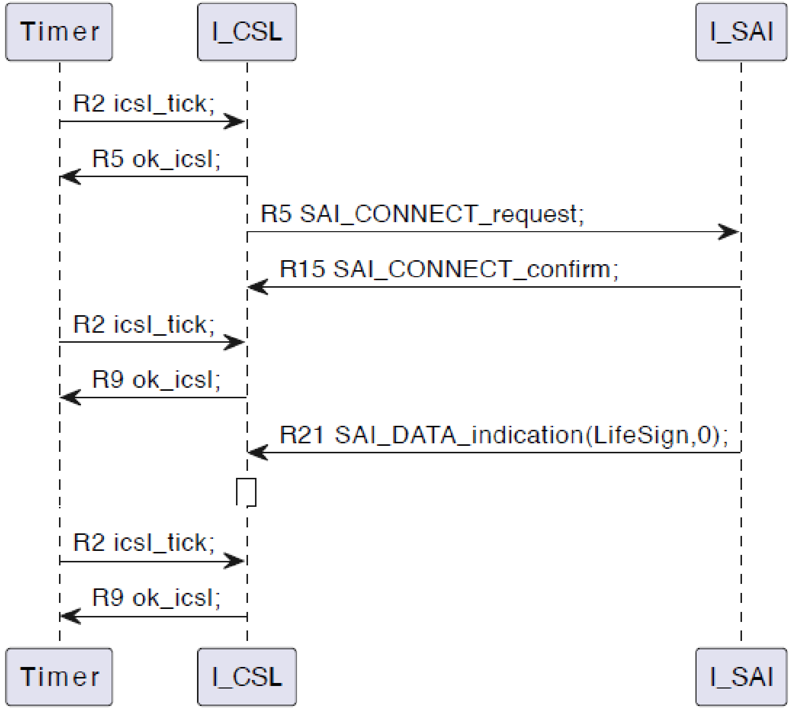
modelled

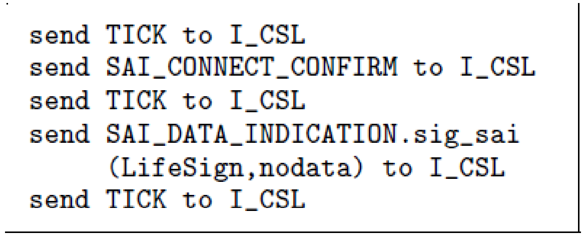

«executable statemachine»
RBC2RBC SI: RBC2RBC SI
C_RBC_USER: RBC_USER
+RBC_USER
:CCSL2RBCUSER
C_CSL: C_CSL
max_receive_timer = 1
max_send_timer = 1
+SAI
:CCSL2SAI
C_SAI: SAI
:CCSL2TIMER
+TIMER
TIMER: TIMER
+TIMER
:ICSL2TIMER
I_RBC_USER: RBC_USER
+RBC_USER
:ICSL2RBCUSER
I_CSL: I_CSL
max_receive_timer = 1
max_connect_timer = 3
max_send_timer = 1
+SAI
:ICSL2SAI
I_SAI: SAI

Formal verification

mutation

```
TICK [this.receive_timer < this.max_receive_timer &&
this.send_timer<this.max_send_timer]
/this.send_timer = this.send_timer+ 1;
this.receive_timer = this.receive_timer+1;
%SEND_EVENT("OK_TICK",CONTEXT_REF(TIMER))%;
```





```
send TICK to I_CSL
send SAI_CONNECT_CONFIRM to I_CSL
send TICK to I_CSL
send SAI_DATA_INDICATION.sig_sai
     (LifeSign,nodata) to I_CSL
send TICK to I_CSL
```

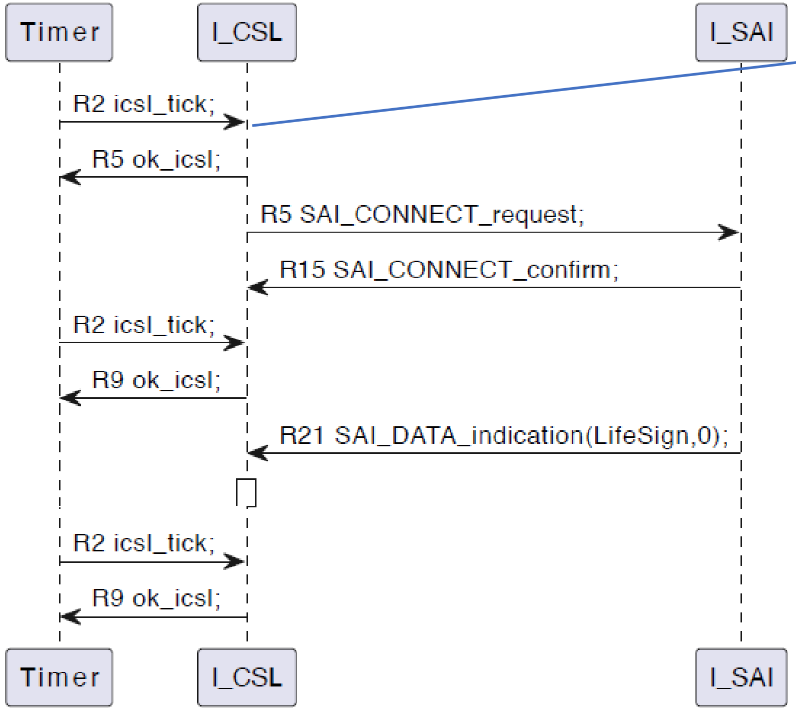
```
Abstractions {State: ICSL.sendtimer >
ICSL.maxsendtimer -> sendTimerError}
```

```
EF sendTimerError
```

Formal verification

mutation

TICK [this.receive_timer < this.max_receive_timer &&
this.send_timer<this.max_send_timer]
/this.send_timer = this.send_timer+ 1;
this.receive_timer = this.receive_timer+1;
%SEND_EVENT("OK_TICK",CONTEXT_REF(TIMER))%;



```
send TICK to I_CSL
send SAI_CONNECT_CONFIRM to I_CSL
send TICK to I_CSL
send SAI_DATA_INDICATION.sig_sai
     (LifeSign,nodata) to I_CSL
send TICK to I_CSL
```

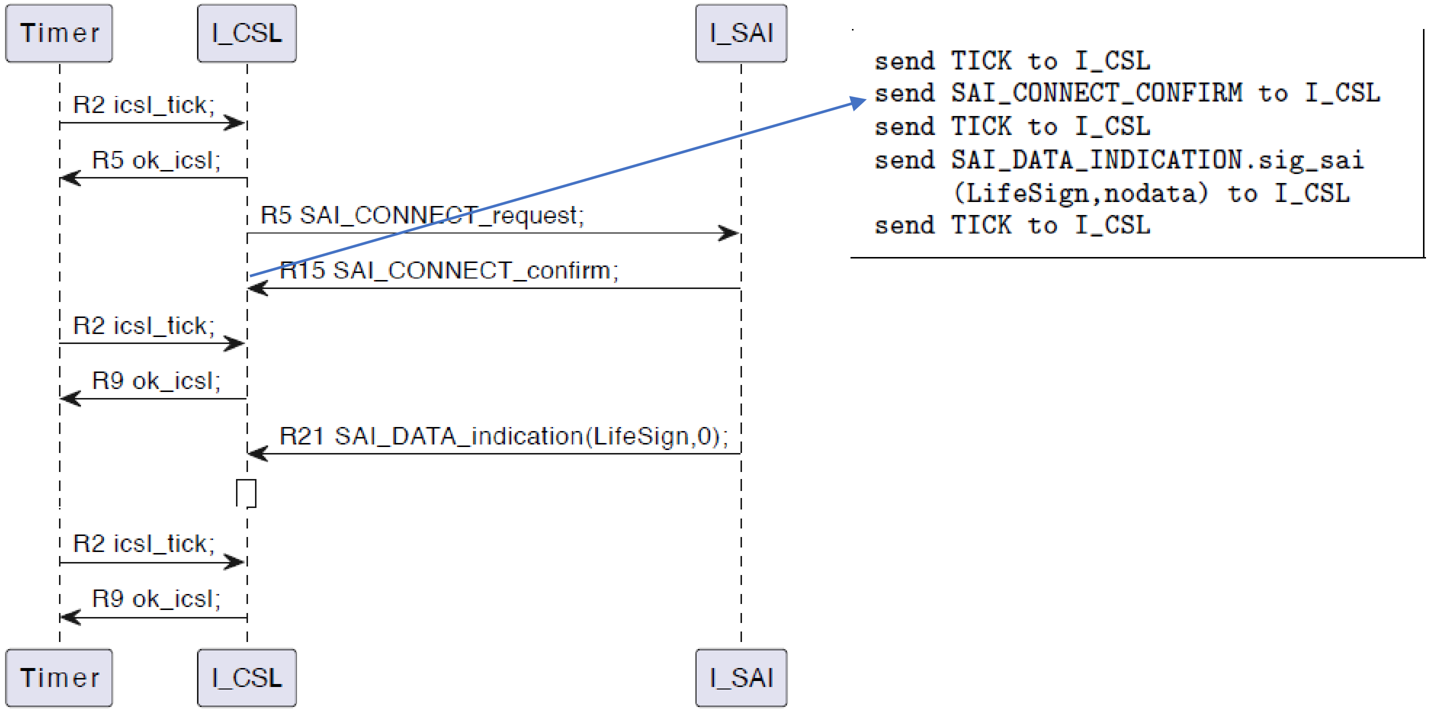
```
Abstractions {State: ICSL.sendtimer >
ICSL.maxsendtimer -> sendTimerError}
```

```
EF sendTimerError
```

Formal verification

mutation

```
TICK [this.receive_timer < this.max_receive_timer &&
this.send_timer<this.max_send_timer]
/this.send_timer = this.send_timer+ 1;
this.receive_timer = this.receive_timer+1;
%SEND_EVENT("OK_TICK",CONTEXT_REF(TIMER))%;
```



```
send TICK to I_CSL
send SAI_CONNECT_CONFIRM to I_CSL
send TICK to I_CSL
send SAI_DATA_INDICATION.sig_sai
     (LifeSign,nodata) to I_CSL
send TICK to I_CSL
```

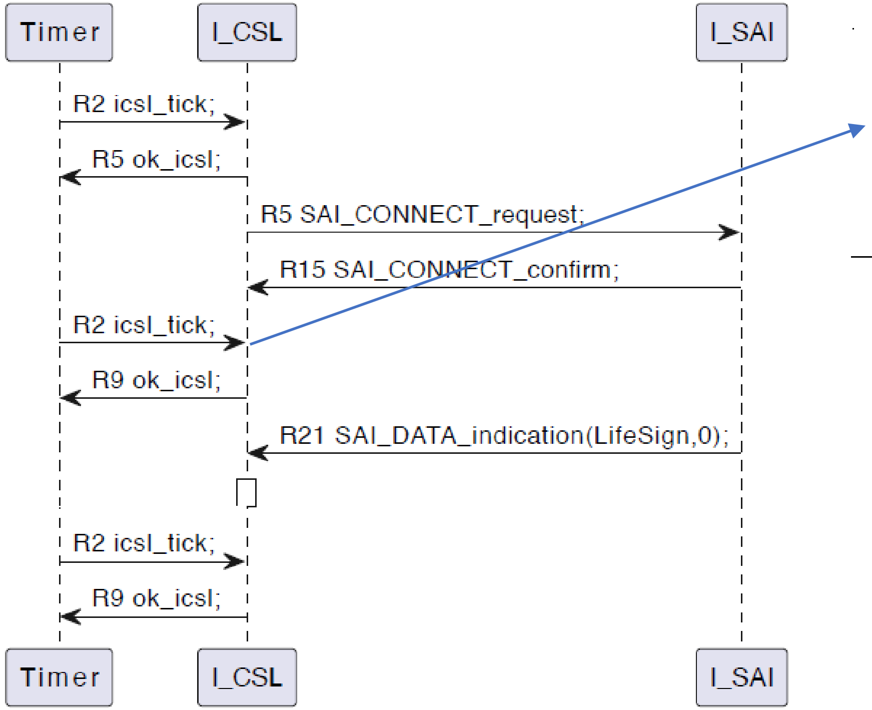
```
Abstractions {State: ICSL.sendtimer >
ICSL.maxsendtimer -> sendTimerError}
```

```
EF sendTimerError
```

Formal verification

mutation

```
TICK [this.receive_timer < this.max_receive_timer &&
this.send_timer<this.max_send_timer]
/this.send_timer = this.send_timer+ 1;
this.receive_timer = this.receive_timer+1;
%SEND_EVENT("OK_TICK",CONTEXT_REF(TIMER))%;
```



```
send TICK to I_CSL
send SAI_CONNECT_CONFIRM to I_CSL
send TICK to I_CSL
send SAI_DATA_INDICATION.sig_sai
     (LifeSign,nodata) to I_CSL
send TICK to I_CSL
```

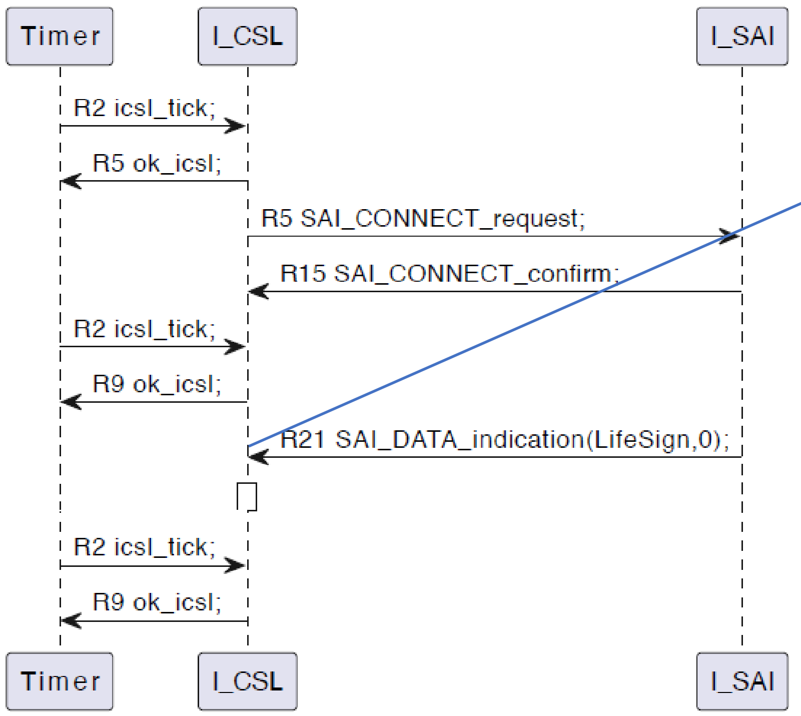
```
Abstractions {State: ICSL.sendtimer >
ICSL.maxsendtimer -> sendTimerError}
```

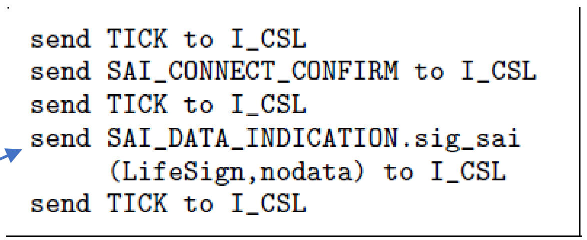
```
EF sendTimerError
```

Formal verification

mutation

```
TICK [this.receive_timer < this.max_receive_timer &&
this.send_timer<this.max_send_timer]
/this.send_timer = this.send_timer+ 1;
this.receive_timer = this.receive_timer+1;
%SEND_EVENT("OK_TICK",CONTEXT_REF(TIMER))%;
```





```
send TICK to I_CSL
send SAI_CONNECT_CONFIRM to I_CSL
send TICK to I_CSL
send SAI_DATA_INDICATION.sig_sai
     (LifeSign,nodata) to I_CSL
send TICK to I_CSL
```

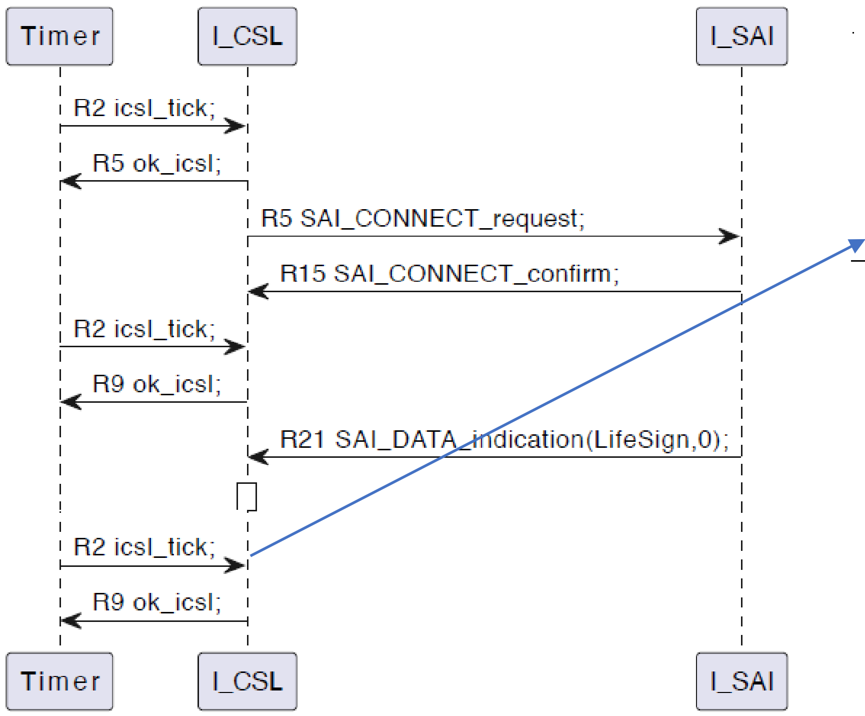
```
Abstractions {State: ICSL.sendtimer >
ICSL.maxsendtimer -> sendTimerError}
```

```
EF sendTimerError
```

Formal verification

mutation

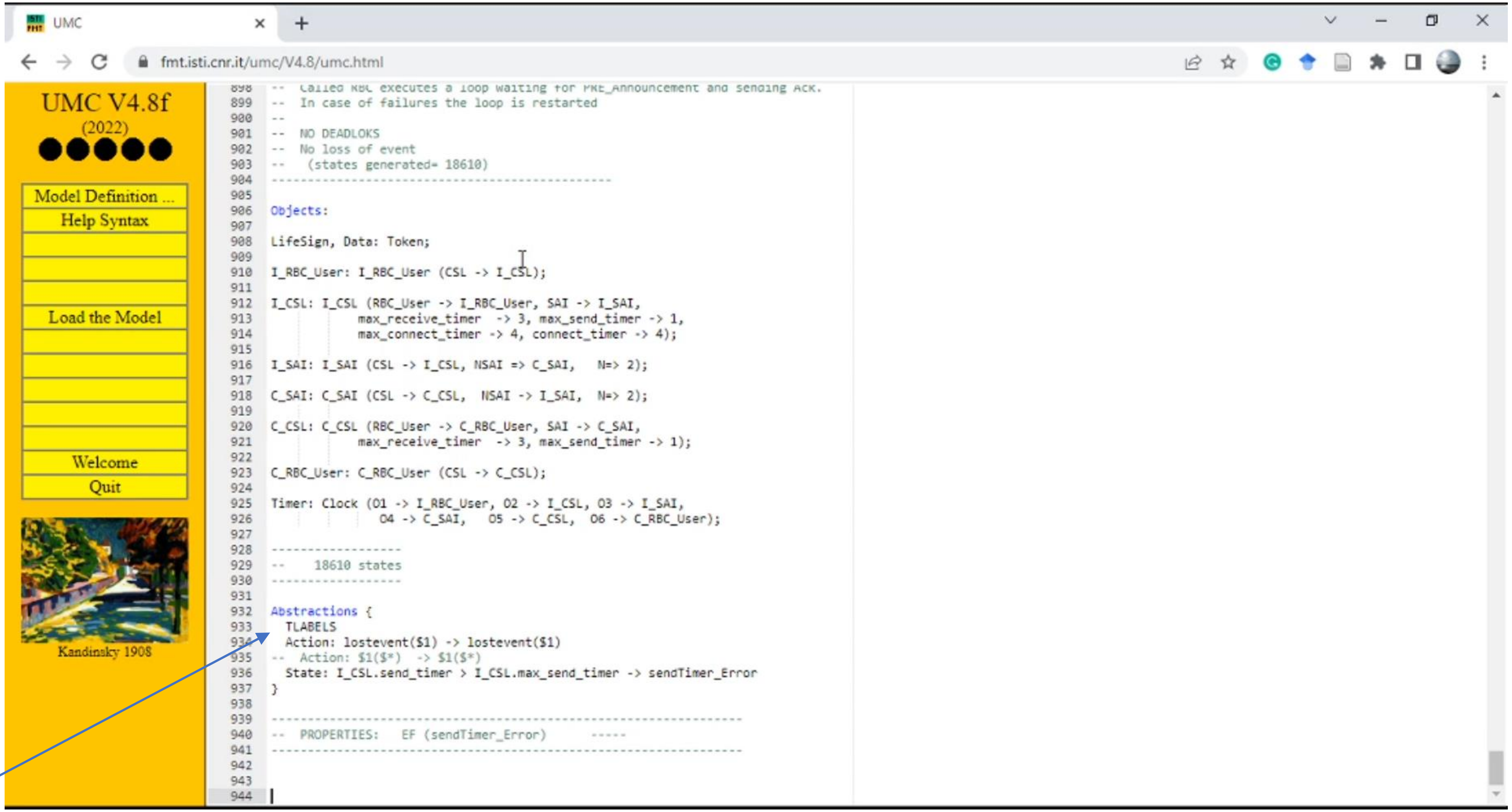
```
TICK [this.receive_timer < this.max_receive_timer &&
this.send_timer<this.max_send_timer]
/this.send_timer = this.send_timer+ 1;
this.receive_timer = this.receive_timer+1;
%SEND_EVENT("OK_TICK",CONTEXT_REF(TIMER))%;
```



```
send TICK to I_CSL
send SAI_CONNECT_CONFIRM to I_CSL
send TICK to I_CSL
send SAI_DATA_INDICATION.sig_sai
     (LifeSign,nodata) to I_CSL
send TICK to I_CSL
```

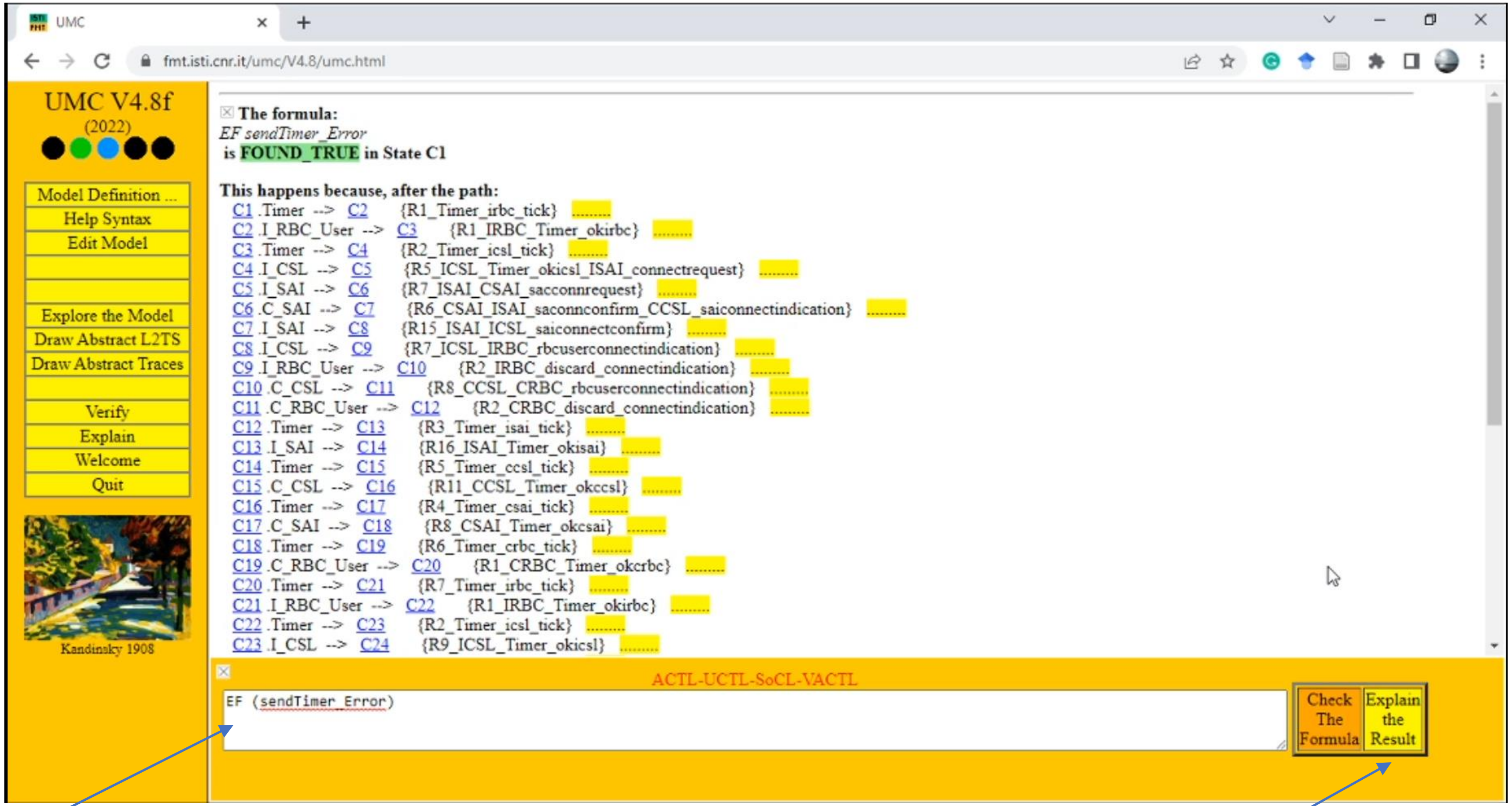
```
Abstractions {State: ICSL.sendtimer >
ICSL.maxsendtimer -> sendTimerError}
```

```
EF sendTimerError
```



```
--  Called RBC executes a loop waiting for PRE_Announcement and sending Ack.
--  In case of failures the loop is restarted
--
--  NO DEADLOKS
--  No loss of event
--   (states generated= 18610)
---------------------------------------------

Objects:

LifeSign, Data: Token;

I_RBC_User: I_RBC_User (CSL -> I_CSL);

I_CSL: I_CSL (RBC_User -> I_RBC_User, SAI -> I_SAI,
            max_receive_timer  -> 3, max_send_timer -> 1,
            max_connect_timer -> 4, connect_timer -> 4);

I_SAI: I_SAI (CSL -> I_CSL, NSAI => C_SAI,   N=> 2);

C_SAI: C_SAI (CSL -> C_CSL,  NSAI -> I_SAI,  N=> 2);

C_CSL: C_CSL (RBC_User -> C_RBC_User, SAI -> C_SAI,
            max_receive_timer  -> 3, max_send_timer -> 1);

C_RBC_User: C_RBC_User (CSL -> C_CSL);

Timer: Clock (O1 -> I_RBC_User, O2 -> I_CSL, O3 -> I_SAI,
               O4 -> C_SAI,   O5 -> C_CSL,  O6 -> C_RBC_User);

-----------------
--    18610 states
-----------------

Abstractions {
  TLABELS
  Action: lostevent($1) -> lostevent($1)
--  Action: $1($*)  -> $1($*)
  State: I_CSL.send_timer > I_CSL.max_send_timer -> sendTimer_Error
}

---------------------------------------------------------------
--  PROPERTIES:   EF (sendTimer_Error)      -----
---------------------------------------------------------------
```

Abstractions {State: ICSL.sendtimer > ICSL.maxsendtimer -> sendTimerError}



The formula:
EF sendTimer_Error
is FOUND_TRUE in State C1

This happens because, after the path:

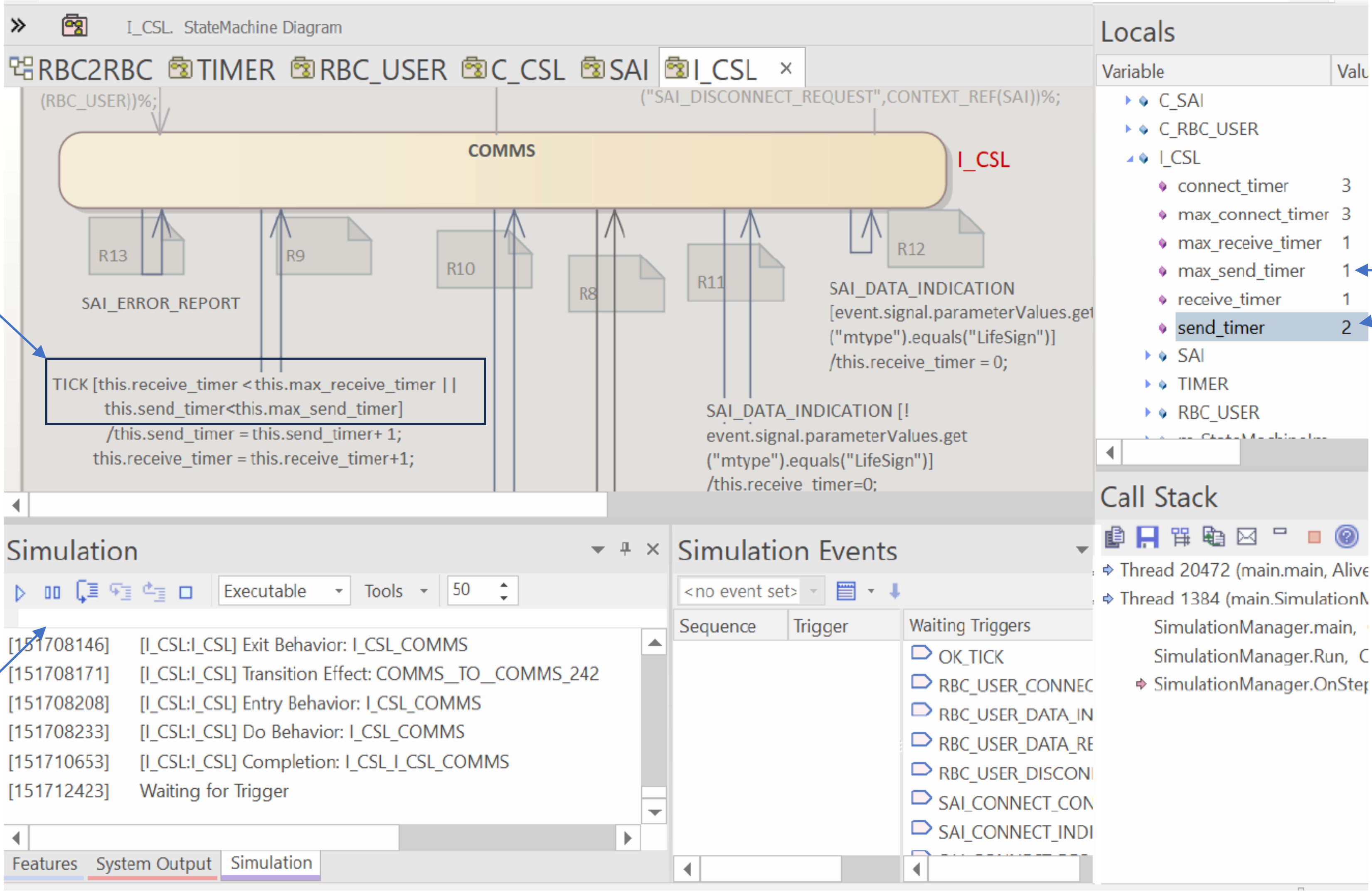
```
C1 .Timer --> C2     {R1_Timer_irbc_tick} .........
C2 .I_RBC_User --> C3     {R1_IRBC_Timer_okirbc} .........
C3 .Timer --> C4     {R2_Timer_icsl_tick} .........
C4 .I_CSL --> C5     {R5_ICSL_Timer_okicsl_ISAI_connectrequest} .........
C5 .I_SAI --> C6     {R7_ISAI_CSAI_sacconnrequest} .........
C6 .C_SAI --> C7     {R6_CSAI_ISAI_saconnconfirm_CCSL_saiconnectindication} .........
C7 .I_SAI --> C8     {R15_ISAI_ICSL_saiconnectconfirm} .........
C8 .I_CSL --> C9     {R7_ICSL_IRBC_rbcuserconnectindication} .........
C9 .I_RBC_User --> C10     {R2_IRBC_discard_connectindication} .........
C10 .C_CSL --> C11     {R8_CCSL_CRBC_rbcuserconnectindication} .........
C11 .C_RBC_User --> C12     {R2_CRBC_discard_connectindication} .........
C12 .Timer --> C13     {R3_Timer_isai_tick} .........
C13 .I_SAI --> C14     {R16_ISAI_Timer_okisai} .........
C14 .Timer --> C15     {R5_Timer_ccsl_tick} .........
C15 .C_CSL --> C16     {R11_CCSL_Timer_okccsl} .........
C16 .Timer --> C17     {R4_Timer_csai_tick} .........
C17 .C_SAI --> C18     {R8_CSAI_Timer_okcsai} .........
C18 .Timer --> C19     {R6_Timer_crbc_tick} .........
C19 .C_RBC_User --> C20     {R1_CRBC_Timer_okcrbc} .........
C20 .Timer --> C21     {R7_Timer_irbc_tick} .........
C21 .I_RBC_User --> C22     {R1_IRBC_Timer_okirbc} .........
C22 .Timer --> C23     {R2_Timer_icsl_tick} .........
C23 .I_CSL --> C24     {R9_ICSL_Timer_okicsl} .........
```

```
EF (sendTimer_Error)
```

`EF sendTimerError`

PlantUML sequence diagram

mutated guard



TICK [this.receive_timer < this.max_receive_timer | |
this.send_timer<this.max_send_timer]
/this.send_timer = this.send_timer+ 1;
this.receive_timer = this.receive_timer+1;

SAI_DATA_INDICATION
[event.signal.parameterValues.get
("mtype").equals("LifeSign")]
/this.receive_timer = 0;

SAI_DATA_INDICATION [!
event.signal.parameterValues.get
("mtype").equals("LifeSign")]
/this.receive_timer=0;

```
[151708146]    [I_CSL:I_CSL] Exit Behavior: I_CSL_COMMS
[151708171]    [I_CSL:I_CSL] Transition Effect: COMMS__TO__COMMS_242
[151708208]    [I_CSL:I_CSL] Entry Behavior: I_CSL_COMMS
[151708233]    [I_CSL:I_CSL] Do Behavior: I_CSL_COMMS
[151710653]    [I_CSL:I_CSL] Completion: I_CSL_I_CSL_COMMS
[151712423]    Waiting for Trigger
```

Variable	Valu
C_SAI	
C_RBC_USER	
I_CSL	
connect_timer	3
max_connect_timer	3
max_receive_timer	1
max_send_timer	1
receive_timer	1
send_timer	2
SAI	
TIMER	
RBC_USER	

```
Thread 20472 (main.main, Alive
Thread 1384 (main.SimulationN
    SimulationManager.main,
    SimulationManager.Run,  C
    SimulationManager.OnStep
```

ICSL.sendtimer >
ICSL.maxsendtimer

interactive console

https://zenodo.org/record/7956438

Conclusion

- Integration of UMC with Sparx EA
  - Notation restrictions
  - Translation Rules
  - Semantics correspondence
  - The output of the formal verification is traced back to Sparx EA
- Lessons learned and limitations
- Future work:
  - full implementation of an application that is formally verified using the proposed methodology.

Conclusion

- Integration of UMC with Sparx EA
  - Notation restrictions
  - Translation Rules
  - Semantics correspondence
  - The output of the formal verification is traced back to Sparx EA
- Lessons learned and limitations
- Future work:
  - full implementation of an application that is formally verified using the proposed methodology.
- https://twitter.com/davidebasile (video of the presentation)
- Thanks for your attention!