C2AADL_Reverse: A Model-Driven Reverse Engineering Approach to Development and Verification of Safety-critical Software

Zhibin Yang[1], Zhikai Qiu[1], Yong Zhou[1], Zhiqiu Huang[1],
Jean-Paul Bodeveix[2], Mamoun Filali[2]

[1]Nanjing University of Aeronautics and Astronautics (NUAA), China

[2]IRIT-Université de Toulouse, France

June 17, 2022

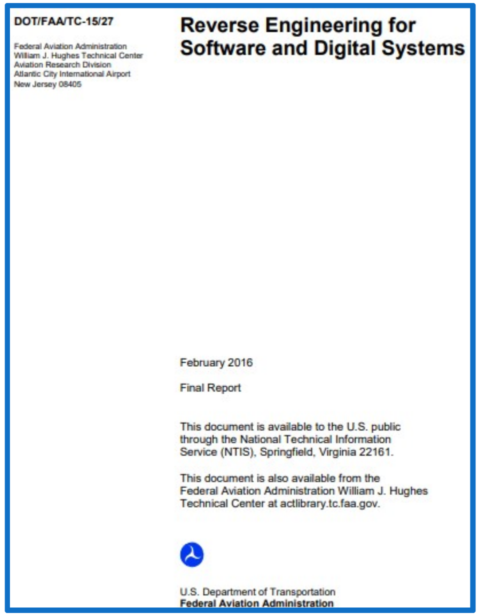
Agenda

- Background & Motivation
- Overview of the Main Contributions
- C2AADL_Reverse Approach
- Validation and Verification Approach for C2AADL_Reverse
- Prototype Tool
- Case Studies
- Conclusion and Future Work

Background & Motivation

✓ RE for Safety-critical software

- **Long-term maintenance** (20-30 years or more)
- Complex challenge: The SCS communities have been struggling to manage and maintain their legacy software.
- FAA: **Reverse Engineering (RE)** has been increasingly used.



DOT/FAA/TC-15/27

Federal Aviation Administration
William J. Hughes Technical Center
Aviation Research Division
Atlantic City International Airport
New Jersey 08405

Reverse Engineering for Software and Digital Systems

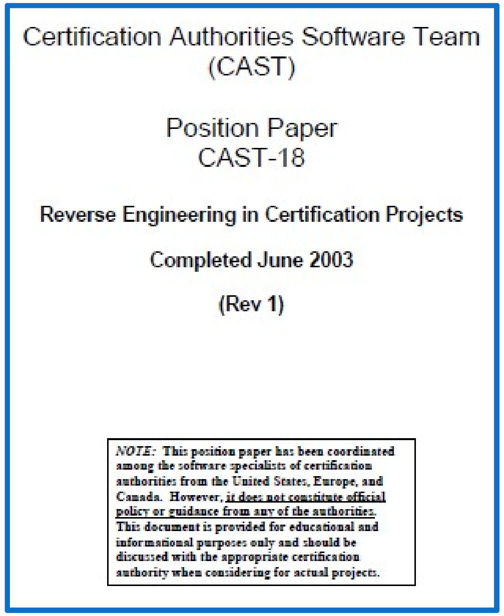
February 2016

Final Report

This document is available to the U.S. public through the National Technical Information Service (NTIS), Springfield, Virginia 22161.

This document is also available from the Federal Aviation Administration William J. Hughes Technical Center at actlibrary.tc.faa.gov.

U.S. Department of Transportation
Federal Aviation Administration



Certification Authorities Software Team (CAST)

Position Paper
CAST-18

Reverse Engineering in Certification Projects

Completed June 2003

(Rev 1)

NOTE: This position paper has been coordinated among the software specialists of certification authorities from the United States, Europe, and Canada. However, it does not constitute official policy or guidance from any of the authorities. This document is provided for educational and informational purposes only and should be discussed with the appropriate certification authority when considering for actual projects.

Background & Motivation

✓**Reverse Engineering (RE)**

- A process to build more abstract representations (such as **architectural models**, or **use cases**, etc) from a low-level representation of a (software) system (such as **source code**, or **execution traces**)

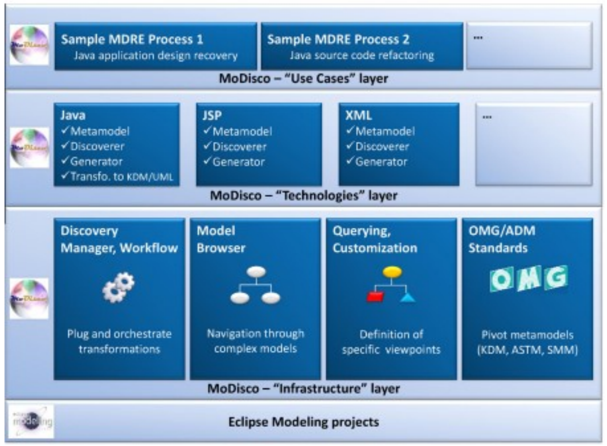
✓**The main objective of RE:**

- Provide a better understanding of the software system's current state, which can be used to **correct** (e.g. fix bugs), **update** (e.g. alignment with updated user requirements), **upgrade** (e.g. add new capabilities), or even **completely re-engineer** the system under study.

Generally, RE is a time-consuming and error-prone process.

Background & Motivation

- ✓ **Model-driven Reverse Engineering (MDRE)** [Spencer Rugaber, 2004]
  - The application of model driven engineering (MDE) principles and techniques to RE
  - Meta-model, model-based views on legacy systems
  - Raising the degree of automatic process through model transformations
- ✓ **Related work** [Claudia Raibulet,2017][André Pascal, 2019][Hugo Brunelière 2014]
  - General solutions
    - MoDisco (model discovery and model understanding, JAVA/JSP/XML -> UML2)
  - Specific solutions (desktop/business/…)
    - Src2MOF (Java -> UML)
    - BREX (Java -> business rules)
    - ITACG (C-> UML)
    - Wang et al. , STOOD (C -> AADL)



Background & Motivation

✓ **The characteristics of MDRE in desktop or business domains.** [Hugo Brunelière 2014]

- Genericity
- Extensibility
- Partial/Full coverage
- Direct (re) use and integration
- Automation

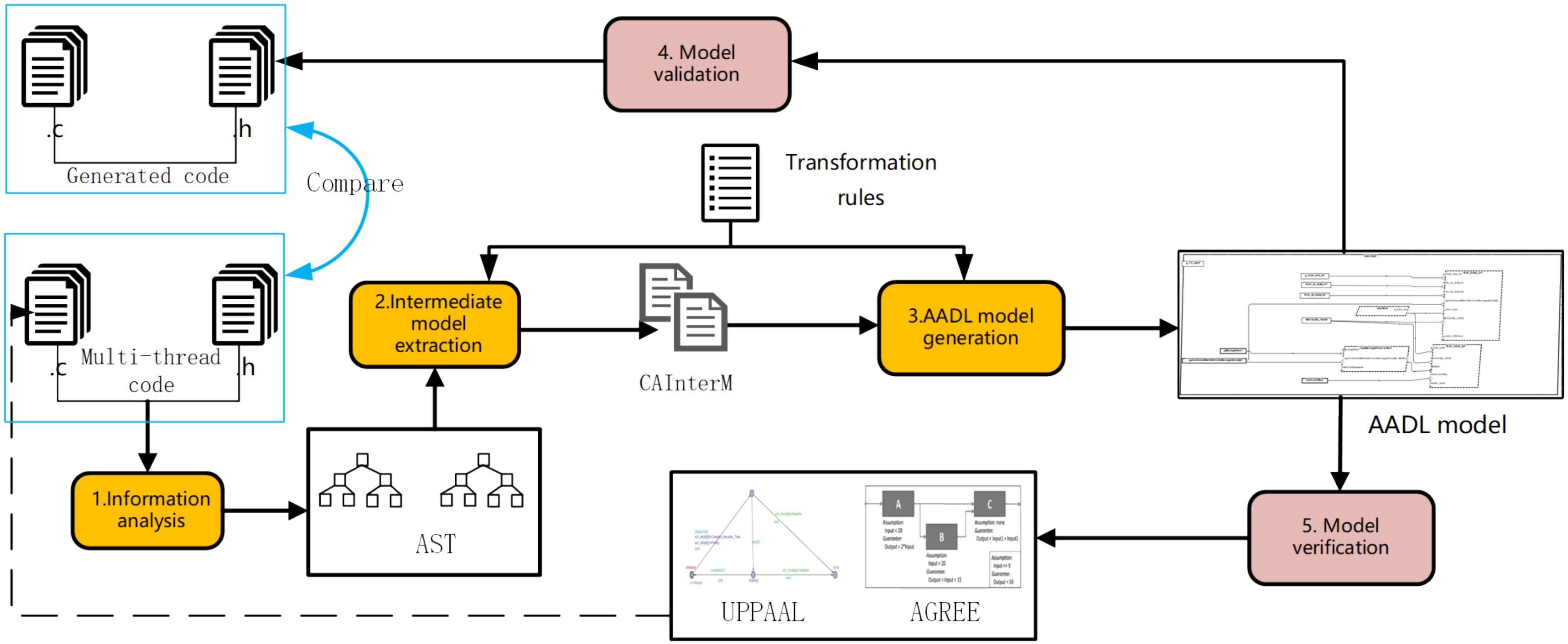
✓ **The characteristics of MDRE in the safety-critical domain.**

- Genericity
- Extensibility
- Partial/Full coverage
  - Architecture
  - Functional Behavior
  - Runtime
- Direct (re) use and integration
- Automation
- validation of RE process
- Verification of resulted models

Main contributions

- ✓ We propose C2AADL_Reverse, a MDRE approach for safety-critical software development and verification:
  - **Domain**: **SCSs** need the modeling of architecture, functional behaviors and runtime.
  - **Source artifacts**: **multi-task C source code** conforms with the "coding rules" in the aerospace industry.
  - **Target models**: compared with the modeling languages used in the existing works of MDRE such as UML, **AADL (Architecture Analysis and Design Language)** is a powerful modeling language for complex embedded system, which provides a unified formalism for the modeling of architecture, functional behaviors, and runtime.
- ✓ **Validation and verification** approach for C2AADL_Reverse
- ✓ **Prototype tool**
- ✓ **Industry case studies**

Main contributions



C2AADL_Reverse approach: step 1- step 3
V&V of C2AADL_Reverse: step 4, step 5

T1: C2AADL_Reverse approach

✓**The features of the source code**

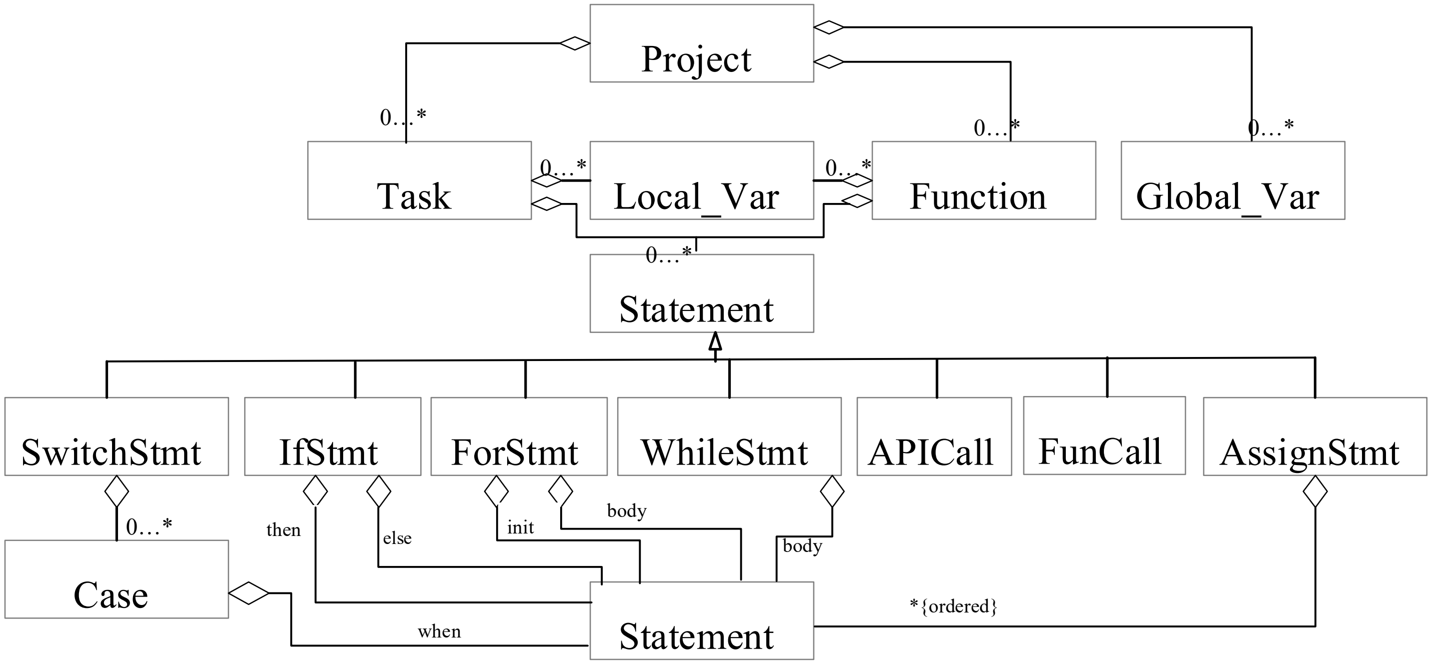
- **Our Case:** the code is structured, i.e., conforms with the coding/programming rules in aerospace industry
  - Multi-tasks
  - Strict development patterns, for example with clear separation of communications, data types, components types, etc.
  - Safety programming: Cyclomatic complexity <10, LOC of each function <100, …
- The code is not structured, then it needs pre-processing (code annotations written manually)

It makes that the RE from C to AADL is feasible

T1: C2AADL_Reverse approach

✓ **Code analysis to build code structure model**

- **Simplified meta-mode of multi-task C code structure**



**Statement* is duplicated for readability

T1: C2AADL_Reverse approach

Example:

```c
void FRAME_LCU_DT_SingleOrder(int info){
        UNION_Cont_Order msg;
        msg.Info=1;
        msg.STRUCT_Bits.MisNnum=1;
        msg.STRUCT_Bits.ContType=0;
        msg.STRUCT_Bits.ContOrder=1;
        msg.STRUCT_Bits.ResChk=1;
        Mailbox_post(MainCont,&msg,BIOS_WAIT_FOREVER);
}
```

```c
void LCU_CT_MainCont(){
        UNION_Cont_Order msg;
        Semaphore_pend(Task_start,BIOS_WAIT_FOREVER);
        while(1){
                Mailbox_pend(MainCont,&msg,BIOS_WAIT_FOREVER);
                if(msg.Info==1){
                        LCU_CT_ContOrd(msg);
                }
        }
}
```

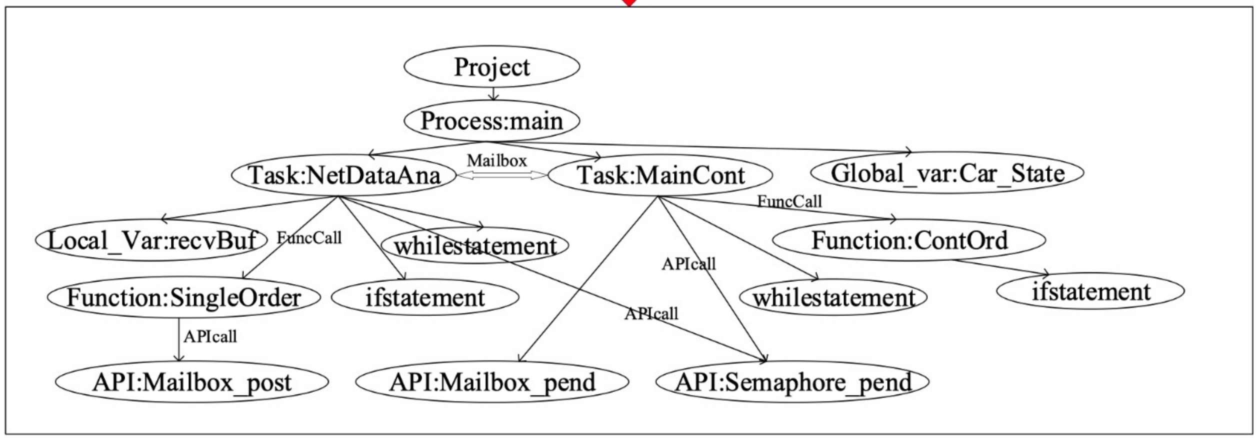
```c
void FRAME_LCU_DT_NetDataAna(){
        Semaphore_pend(Task_start,BIOS_WAIT_FOREVER);
        int recvBuf=0;
        while(1){
                Mailbox_pend(NetRecv,&recvBuf,BIOS_WAIT_FOREVER);
                if(recvBuf==0){
                        FRAME_LCU_DT_SingleOrder(recvBuf);
                }
        }
}
```

```c
void LCU_CT_ContOrd(UNION_Cont_Order msg){
 if(&msg!=NULL){
  if(msg.STRUCT_Bits.ContType==0){
   if(msg.STRUCT_Bits.ContOrder==1){
    FRAME_PC_MisReady1(msg.STRUCT_Bits.MisNnum);
    LCU_WD_PlatInfo(msg.STRUCT_Bits.MisNnum);
    if(msg.STRUCT_Bits.ResChk==1){
     LCU_WD_PowerOn1Chk(msg.STRUCT_Bits.MisNnum);
    }
   }
  }
 }
}
```

```c
int Car_State;
int main()
{
    Task_Params task_NetDataAna;
    Task_Params_init(&task_NetDataAna);
    task_NetDataAna.priority = 5;

    Task_Params task_LCU_CT_MainCont;
    Task_Params_init(&task_LCU_CT_MainCont);
    task_LCU_CT_MainCont.priority = 7;

    Task_create(LCU_CT_MainCont, &task_LCU_CT_MainCont, NULL);
    Task_create(FRAME_LCU_DT_NetDataAna, &task_NetDataAna, NULL);
    MainCont = Mailbox_create(sizeof(UNION_Cont_Order),1,NULL,NULL);
    Car_State=0;
    BIOS_start();   /* does not return */
    return(0);
}
```



T1: C2AADL_Reverse approach

✓ **Structure transformation**

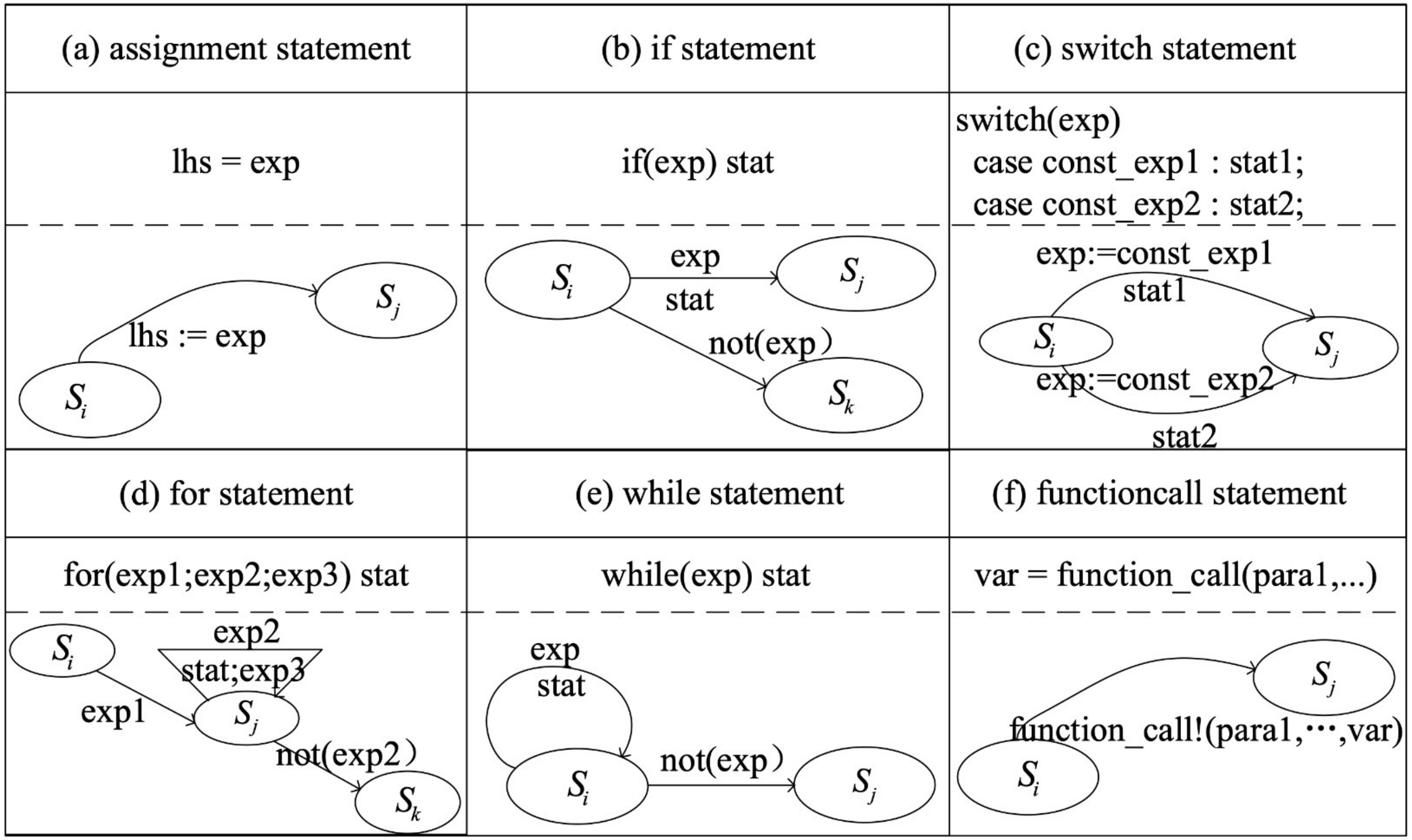
- **Plain code -> namespaces, source code files -> create hierarchy**
- Data types -> data components
- function definition -> subprogram component
- Task definition -> thread component

C language	AADL
int a; int *a char a; char *a bool a; bool *a float a; float *a; struct_name a; struct_name *a;	Base_Types::Integer; requires data access Base_Types::Integer Base_Types::Character requires data access Base_Types::Character Base_Types::Boolean requires data access Base_Types::Boolean Base_Types::Float requires data access Base_Types::Float User_Defined::struct_name.impl; requires data access User_Defined::struct_name.impl;
signed int a; unsigned int b;	Base_Types::Integer_32; Base_Types::Unsigned_32; **data** Integer_32 **extends** Integer **properties** Data_Model::Number_Representation ⇒ Signed; **end** Integer_32; **data** Unsigned_32 **extends** Integer **properties** Data_Model::Number_Representation ⇒ Unsigned; **end** Unsigned_32;
struct dataname{ type_spec var_name; }; enum; union;	**data** dataname **properties** Data_Model::Data_Representation⇒ (Struct/Union/Enum); **end** dataname **data implementation** dataname.impl **subcomponents** var_name: data package_name::type_spec **end** dataname.impl
function definition	subprogram component
task structure	thread component

T1: C2AADL_Reverse approach

✓ **Behavior annex transformation**

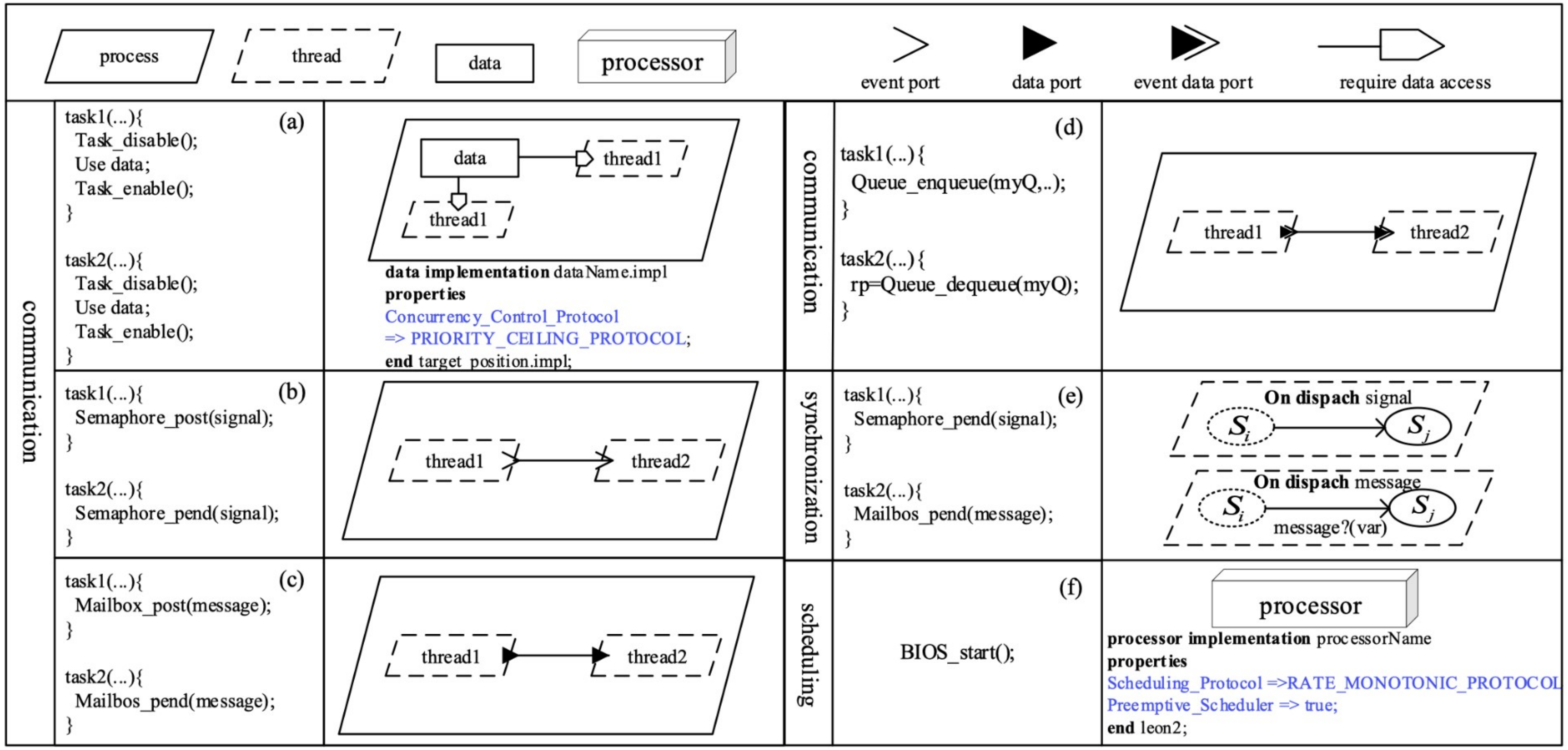
- **Modeling and verification feasibility: cyclomatic complexity <10, LOC of each function <100, …**



T1: C2AADL_Reverse approach

✓ **Run-time information transformation**

- **The APIs of OS or runtime execution platform**

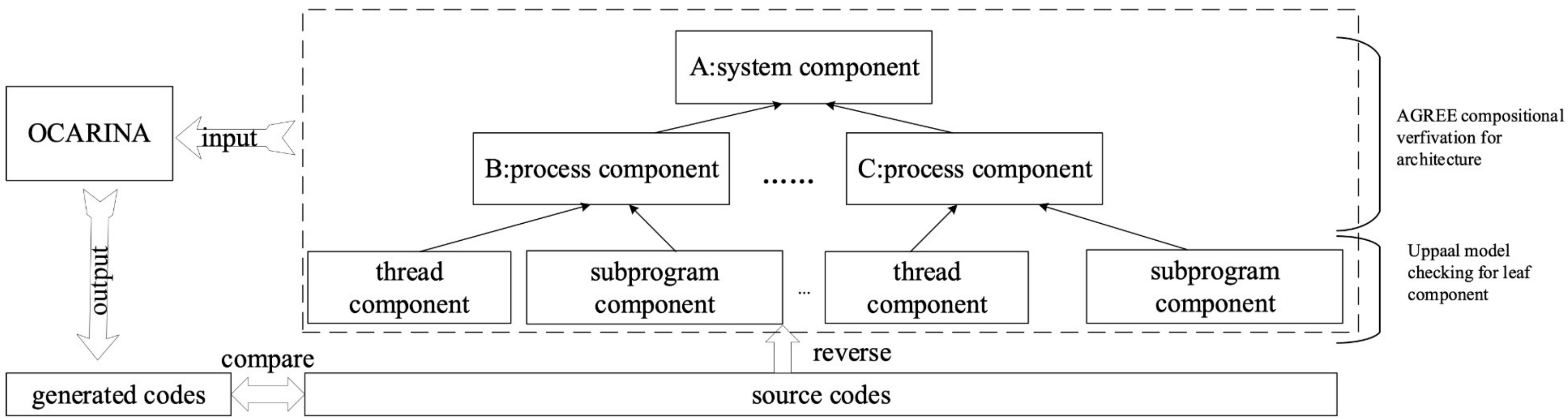


*Without loss of generality, we consider TI SYS/BIOS Real-time Operating System (SYS/BIOS) which is broadly used in the aerospace domain.

T2: V&V of C2AADL_Reverse

✓ **Global view**

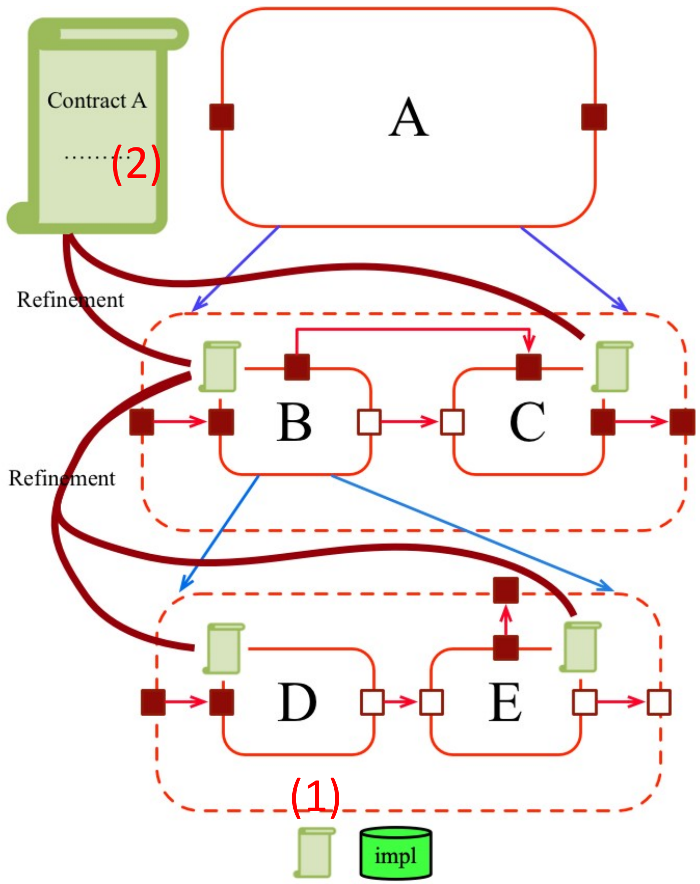
- Validation of the RE process by using a comparison between two-versions code (see case study)
- Compositional verification of the architecture model
- Verification of the leaf components



T2: V&V of C2AADL_Reverse

✓ **The principle of compositional verification**

- The state-explosion problem
- The verification of a composite system is reduced to the verification of its parts.
- Requirements are decomposed and formalized into contracts and subcontracts: **<Assume, Guarantee>**
- **AADL AGREE annex and tool**



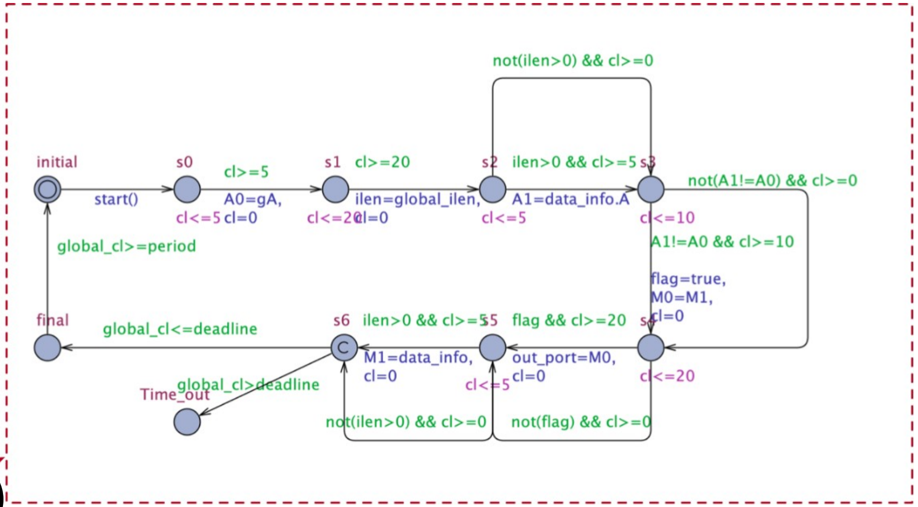
T2: V&V of C2AADL_Reverse

✓ **Verification of leaf components with UPPAAL**

- **Why we use UPPAAL? It has been used in industry**
- **Properties: safety, liveness, no deadlock (Component-level contracts: Assume-> initialization function, Guarantee -> TCTL)**

```
thread implementation E00.impl
    annex behavior_specification {**
        variables
            A0: Base_Types::Unsigned_16;
            A1: Base_Types::Unsigned_16;
            gA: Base_Types::Unsigned_16;
            ilen : Base_Types::Integer;
            flag: Base_Types::Boolean;
            M0: info.impl;
            M1: info.impl;
        states
            s0: initial complete state;
            ……;
            s6: state;
        transitions
            T_0: s0-[on dispatch]->s1{A0:=gA;
                computation(5ms)
            };
            T_1: s1-[on dispatch in_port]->s2{in_port?(ilen);
                computation(20ms)
            };
            T_2: s2-[ilen>0]->s3{A1=data_info.A;
                computation(5ms)
            };
            T_3: s2-[not(ilen>0)]->s3;
            T_4: s3-[A1!=A0]->s4{flag:=true;M0:=M1;
                computation(10ms)
            };
            T_5: s3-[not(A1!=A0)]->s4;
            T_6: s4-[flag]->s5{out_port!(M0);
                computation(20ms)
            };
            T_7: s4-[not(flag)]->s5;
            T_8: s5-[ilen>0]->s6{M1=data_info;
                computation(5ms)
            };
            T_9: s5-[not(ilen>0)]->s6;
    **};
```

(1)




```
annex agree {**
    assume "assume" : in_port>0;
    guarantee "guarantee":out_port=data_info;
**};
```

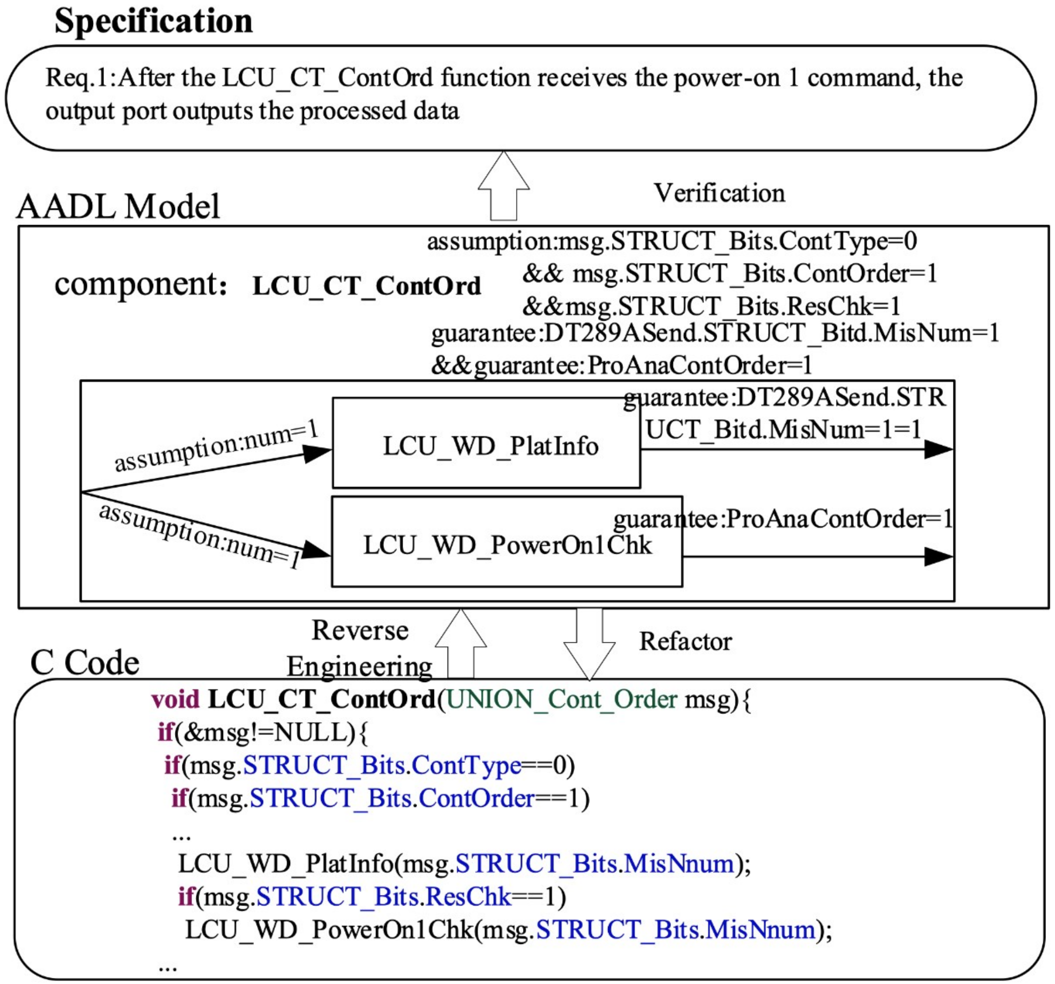
(2)

```
A[] Process.s6 imply out_port==Process.M0
A[] !Process.Time_out
A[] !deadlock
```

T2: V&V of C2AADL_Reverse

✓ **Compositional verification of AADL architecture model**

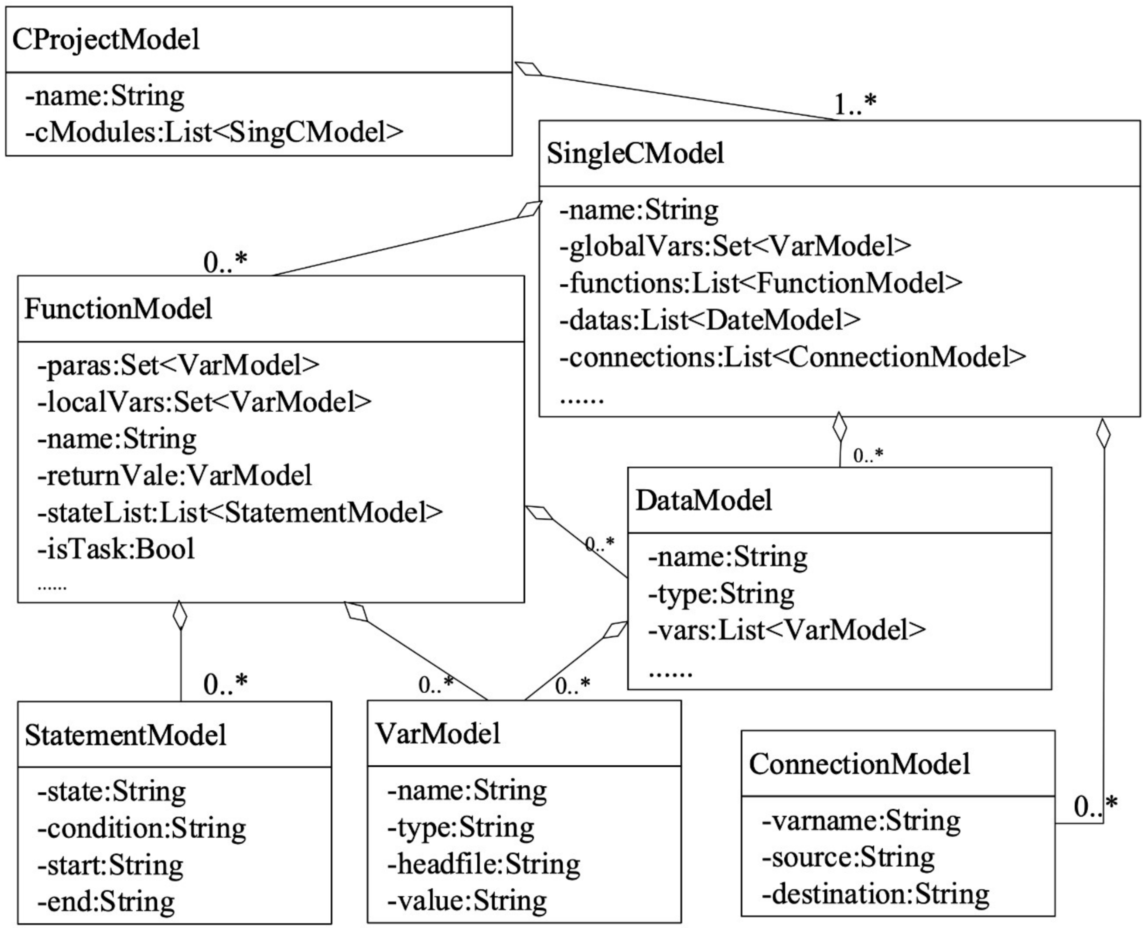
- **System-level properties (system-level contracts)**



T3: Prototype tool

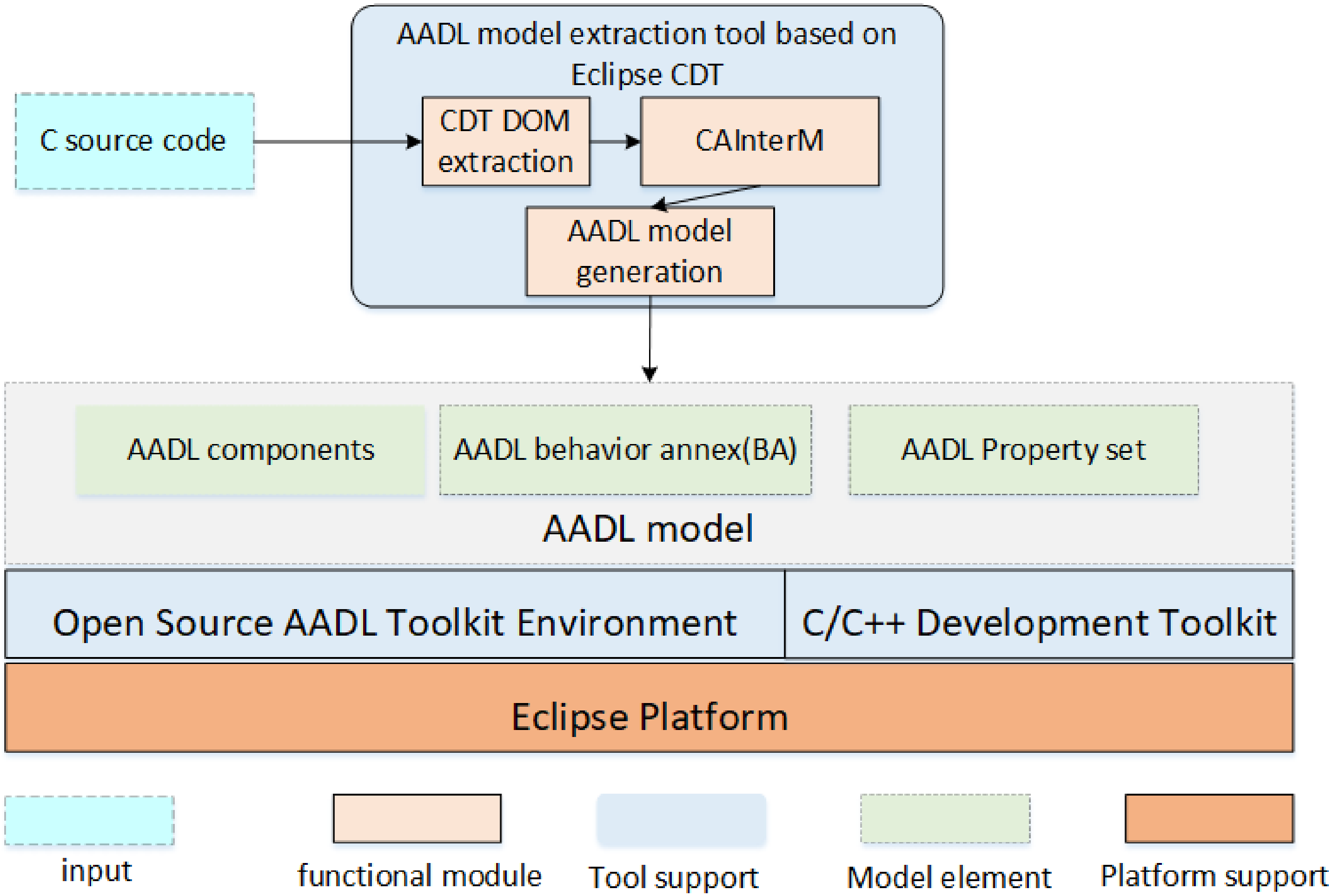
✓Intermediate model

- **Consider the extensibility**



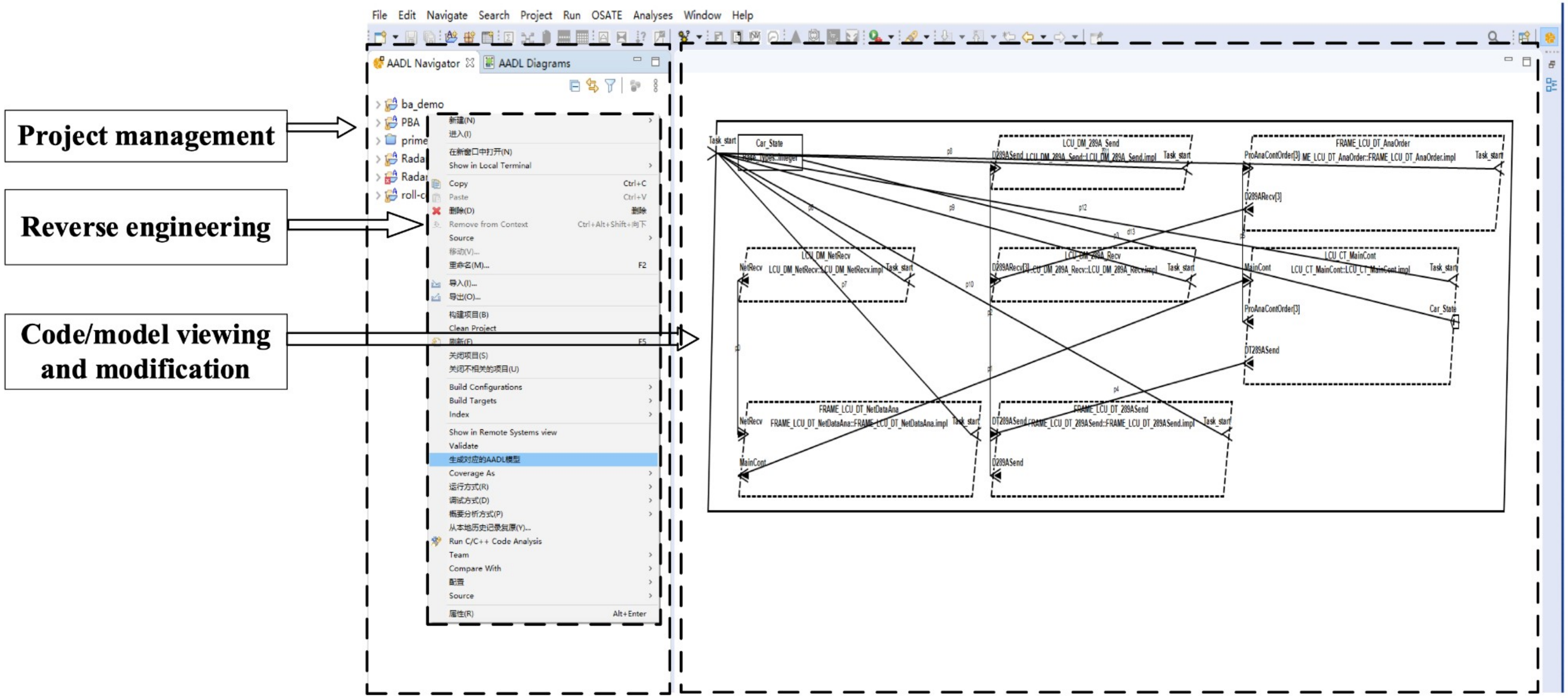
T3: Prototype tool

✓ **Implementation of the tool**



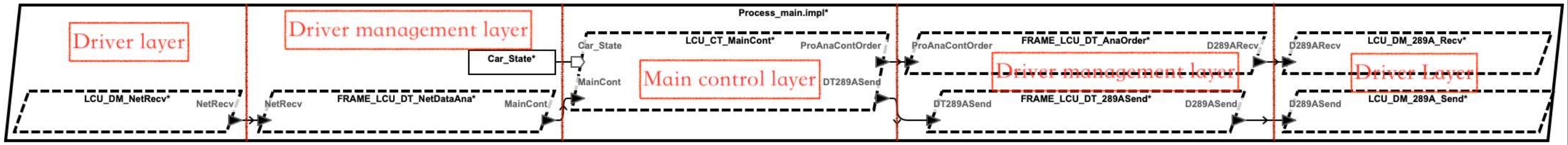
T3: Prototype tool

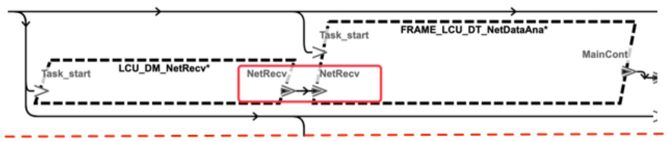
✓ **Implementation of the tool**



T4: Evaluation with case studies

✓**Generated AADL models**





Out data port

```
thread LCU_DM_NetRecv
    features
        Task_start : in event port ;
        NetRecv : out event data port Base_Types::Integer;
    properties
        Priority=>2;
end LCU_DM_NetRecv;

thread implementation LCU_DM_NetRecv.impl
    annex behavior_specification {**
        variables
            recvBuf: Base_Types::Integer;
        states
            s0: initial complete  state;
            s1:  state;
        transitions
            T_0: s0-[on dispatch Task_start]->s1{
                recvBuf:=0;
                NetRecv!(recvBuf);
                recvBuf:=recvBuf + 1};
    **};
end LCU_DM_NetRecv.impl;
```

In data port

Receive data form the port

```
thread FRAME_LCU_DT_NetDataAna
    features
        Task_start : in event port ;
        NetRecv : in event data port Base_Types::Integer;
        MainCont : out event data port Globle_Define::UNION_Cont_Order.impl;
    properties
        Priority=>5;
end FRAME_LCU_DT_NetDataAna;

thread implementation FRAME_LCU_DT_NetDataAna.impl
    annex behavior_specification {**
        variables
            recvBuf: Base_Types::Integer;
        states
            s0: initial complete  state;
            s1: state;
            s2:  complete state;
            s3:  state;
        transitions
            T_0: s0-[on dispatch Task_start]->s1{
                recvBuf:=0};
            T_1: s1-[true]->s2;
            T_2: s2-[on dispatch NetRecv]->s3{
                NetRecv?(recvBuf)};
    **};
```

function_name

parameter

Global Variable

Behavior Annex

```
subprogram LCU_CT_ContOrd
    features
        msg: in parameter Globle_Define::UNION_Cont_Order.impl;
        DT289ASend : out event data port Globle_Define::UNION_Data_Bind_Info.impl;
        ProAnaContOrder : out event data port Base_Types::Integer[3];
        Car_State:requires data access Globle_Define::Car_State.impl;
end LCU_CT_ContOrd;

subprogram implementation LCU_CT_ContOrd.impl
    subcomponents
        LCU_WD_PlatInfo: subprogram LCU_WD_PlatInfo::LCU_WD_PlatInfo.impl;
        LCU_WD_PowerOn1Chk: subprogram LCU_WD_PowerOn1Chk::LCU_WD_PowerOn1Chk.impl;
        FRAME_PC_MisReady1: subprogram FRAME_PC_MisReady1::FRAME_PC_MisReady1.impl;
    connections
        p0: port LCU_WD_PlatInfo.DT289ASend->DT289ASend;
        p1: port LCU_WD_PowerOn1Chk.ProAnaContOrder->ProAnaContOrder;
        p2: data access Car_State<->FRAME_PC_MisReady1.Car_State;
    annex behavior_specification {**
        states
            s0: initial state;
            s1: state;
            s2: state;
            s3: state;
            s4: final state;
        transitions
            T_0: s0-[msg!="NULL"]->s1;
            T_1: s1-[msg.STRUCT_Bits.ContType=0]->s2;
            T_2: s2-[msg.STRUCT_Bits.ContOrder=1]->s3{
                FRAME_PC_MisReady1!(msg.STRUCT_Bits.MisNnum,Car_State);
                LCU_WD_PlatInfo!(msg.STRUCT_Bits.MisNnum)};
            T_3: s3-[msg.STRUCT_Bits.ResChk=1]->s4{
                LCU_WD_PowerOn1Chk!(msg.STRUCT_Bits.MisNnum)};
            T_4: s3-[not(msg.STRUCT_Bits.ResChk=1)]->s4;
            T_5: s2-[not(msg.STRUCT_Bits.ContOrder=1)]->s4;
            T_6: s1-[not(msg.STRUCT_Bits.ContType=0)]->s4;
            T_7: s0-[not(msg!="NULL")]->s4;
    **};
end LCU_CT_ContOrd.impl;
```

T4: Evaluation with case studies

✓**Generated AADL models**

	AADL model (line)	Threads	Subps	Coverage
Exhaust cover control	1800+	4	14	93%
Rocket hatch control	1600+	3	12	92%
Control of rocket launch preparation/cancellation	1600+	4	11	94%
Rocket power-on control	4000+	18	32	93%
Control of rocket hatch unlock/lock	2000+	6	13	95%
Thermal battery activation control	1800+	6	15	93%
Control of safety mechanism unlock/lock	1900+	6	15	95%
Rocket ignition control	1700+	4	13	93%
Rocket power-off control	1200+	4	9	94%
Rocket launch control system	17600+	55	134	94%

*The reason why the coverage rate of the generated model does not reach 100% is that some codes are not easily expressed in the behavior annex, such as bit operation and type mandatory conversion, etc.

It allows us to complement and refine the models.

T4: Evaluation with case studies

✓Validation of the C2AADL RE process

- **Comparison between two-versions source code**

code reviewing

```
void LCU_DM_NetRecv(){
        System_printf("enter LCU_DM_NetRecv()\n");
        Semaphore_pend(Task_start,BIOS_WAIT_FOREVER);
        System_printf("LCU_DM_NetRecv() start\n");
        int recvBuf=0;
        while(1){
                System_printf("LCU_DM_NetRecv send %d message\n",recvBuf);
                Mailbox_post(NetRecv,&recvBuf,BIOS_WAIT_FOREVER);
                recvBuf++;

        }

        System_printf("exit LCU_DM_NetRecv()\n");
}
```

```
void FRAME_LCU_DT_NetDataAna(){
        System_printf("enter FRAME_LCU_DT_NetDataAna()\n");
        Semaphore_pend(Task_start,BIOS_WAIT_FOREVER);
        System_printf("FRAME_LCU_DT_NetDataAna() start\n");
        int recvBuf=0;
        while(1){
                Mailbox_pend(NetRecv,&recvBuf,BIOS_WAIT_FOREVER);
                if(recvBuf==0){
                        System_printf("recvBuf is %d\n",recvBuf);
                        FRAME_LCU_DT_SingleOrder(recvBuf);
                }
        }
}
```

OCARINA

```
extern uint8_t netrecv_id;
void* lcu_dm_netrecv_job (void)
{
  pok_ret_t ret;
  while (1)
  {
    /*  Send the OUT ports*/
    ret = pok_buffer_send (netrecv_id, &(netrecv_dvalue), sizeof (int), 0);
    ASSERT_RET(ret);
    pok_thread_period ();
  }
}

/*  Periodic task : FRAME_LCU_DT_NetDataAna*/
base_types__integer netrecv_dvalue;
extern uint8_t netrecv_id;
globle_define__union_cont_order maincont_dvalue;
extern uint8_t maincont_id;
void* frame_lcu_dt_netdataana_job (void)
{
  pok_ret_t ret;
  pok_port_size_t netrecv_length;
  while (1)
  {
    /*  Get the IN ports values*/
    ret = pok_buffer_receive (netrecv_id, 0, &(netrecv_dvalue)
&(netrecv_length));
    ASSERT_RET_WITH_EXCEPTION(ret, POK_ERRNO_EMPTY);
    /*  Copy directly the data from IN ports to OUT ports*/
    /*  Send the OUT ports*/
    ret = pok_buffer_send (maincont_id, &(maincont_dvalue), sizeof (int), 0);
    ASSERT_RET(ret);
    pok_thread_period ();
  }
}
```

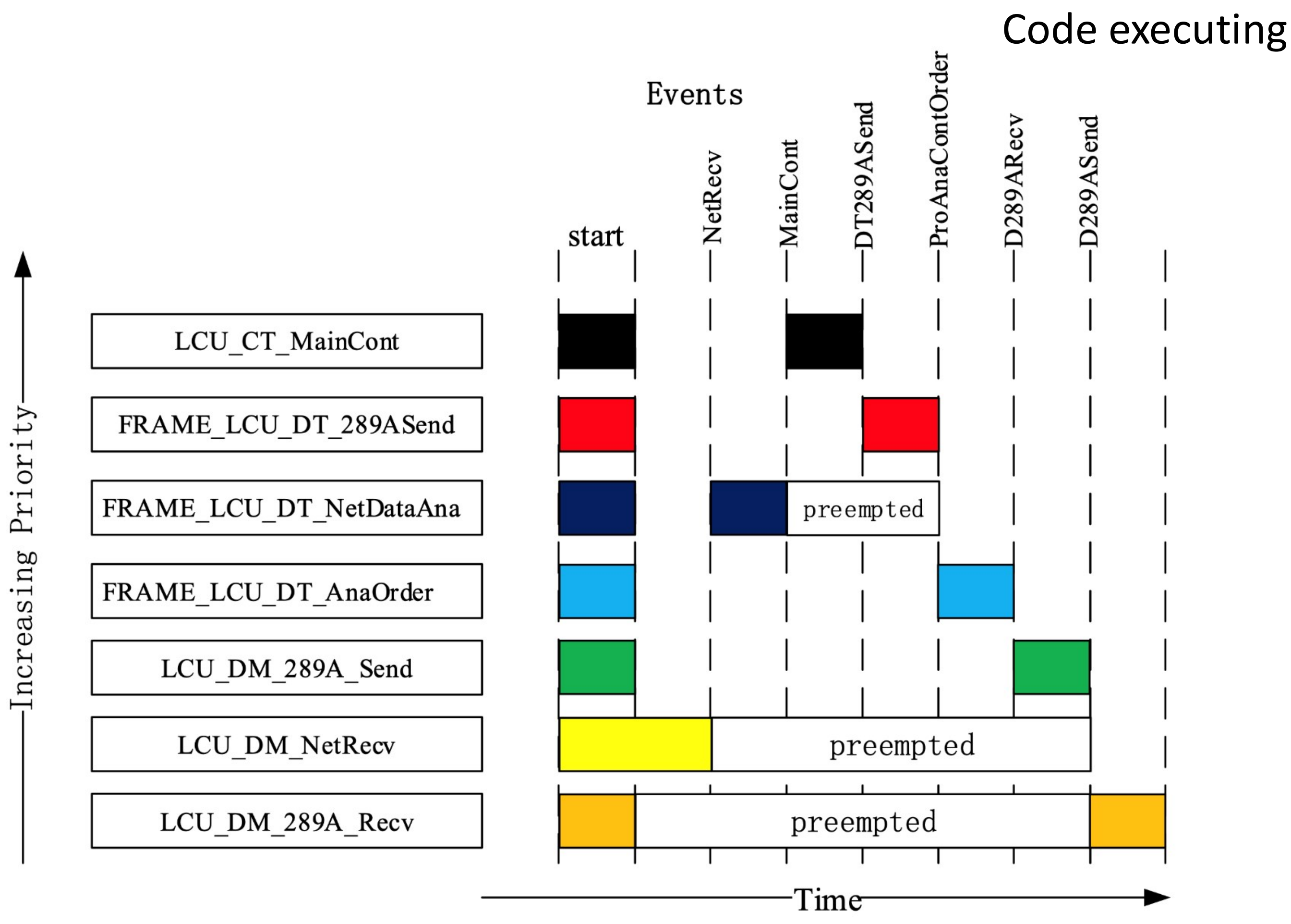
SYS/BIOS: Mailbox_post, Mailbox_pend

POK: pok_buffer_send, pok_buffer_receive

T4: Evaluation with case studies

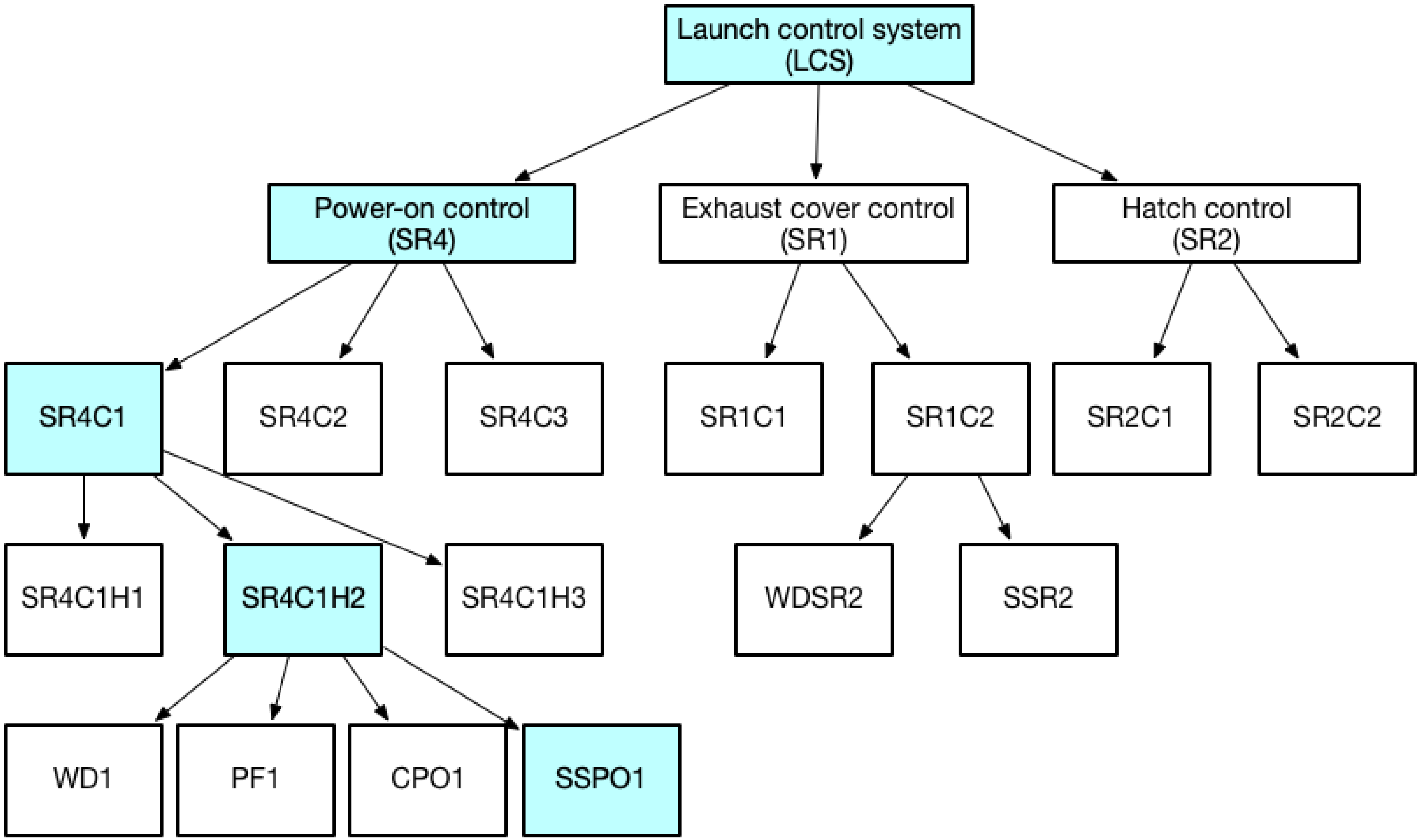
✓**Validation of the C2AADL RE process**

- **Comparison between two-versions source code**



T4: Evaluation with case studies

✓ **Compositional Verification of the generated AADL models**



T4: Evaluation with case studies

✓ **Compositional Verification of the generated AADL models**

```
thread FRAME_LCU_DT_NetDataAna
    features
        Task_start : in event port ;
        NetRecv : in event data port Base_Types::Integer;
        MainCont : out event data port Globle_Define::UNION_Cont_Order.impl;
    properties
        Priority=>5;
    annex agree {**
    property judge_MainCont=
    MainCont.Info=1 and MainCont.STRUCT_Bits.MisNnum=1 and MainCont.STRUCT_Bits.ContType=0;
    assume "A:FRAME_LCU_DT_NetDataAna receive data from NetRecv" : NetRecv = 0;
    guarantee "G:FRAME_LCU_DT_NetDataAna send data to MainCont" : judge_MainCont;
**};
end FRAME_LCU_DT_NetDataAna;
```

```
thread LCU_CT_MainCont
    features
        Task_start : in event port ;
        MainCont : in event data port Globle_Define::UNION_Cont_Order.impl;
        DT289ASend : out event data port Globle_Define::UNION_Data_Bind_Info.impl;
        ProAnaContOrder : out event data port Base_Types::Integer;
        Car_State:requires data access Globle_Define::Car_State.impl;
    properties
        Priority=>7;
    annex agree {**
        property judge_MainCont=
            MainCont.Info=1 and MainCont.STRUCT_Bits.MisNnum=1 and MainCont.STRUCT_Bits.ContType=0;

        property judge_DT289ASend=
            DT289ASend.STRUCT_Bits.MisNnum=1 ;

        assume "A:LCU_CT_MainCont receive cont from NetDataAna" : judge_MainCont;
        guarantee "G:LCU_CT_MainCont send order to FRAME_LCU_DT_289ASend" : ProAnaContOrder=1;
        guarantee "G:LCU_CT_MainCont send data to FRAME_LCU_DT_AnaOrder" :  judge_DT289ASend;
    **};
end LCU_CT_MainCont;
```

```
thread FRAME_LCU_DT_AnaOrder
    features
        Task_start : in event port ;
        ProAnaContOrder : in event data port Base_Types::Integer;
        D289ARecv : out event data port Base_Types::Integer;
    properties
        Priority=>4;
    annex agree {**
        assume "A:FRAME_LCU_DT_AnaOrder receive data from LCU_CT_MainCont":ProAnaContOrder=1;
        guarantee "G:FRAME_LCU_DT_AnaOrder send data to LCU_DM_289A_Recv":D289ARecv=1;
    **};
end FRAME_LCU_DT_AnaOrder;
```

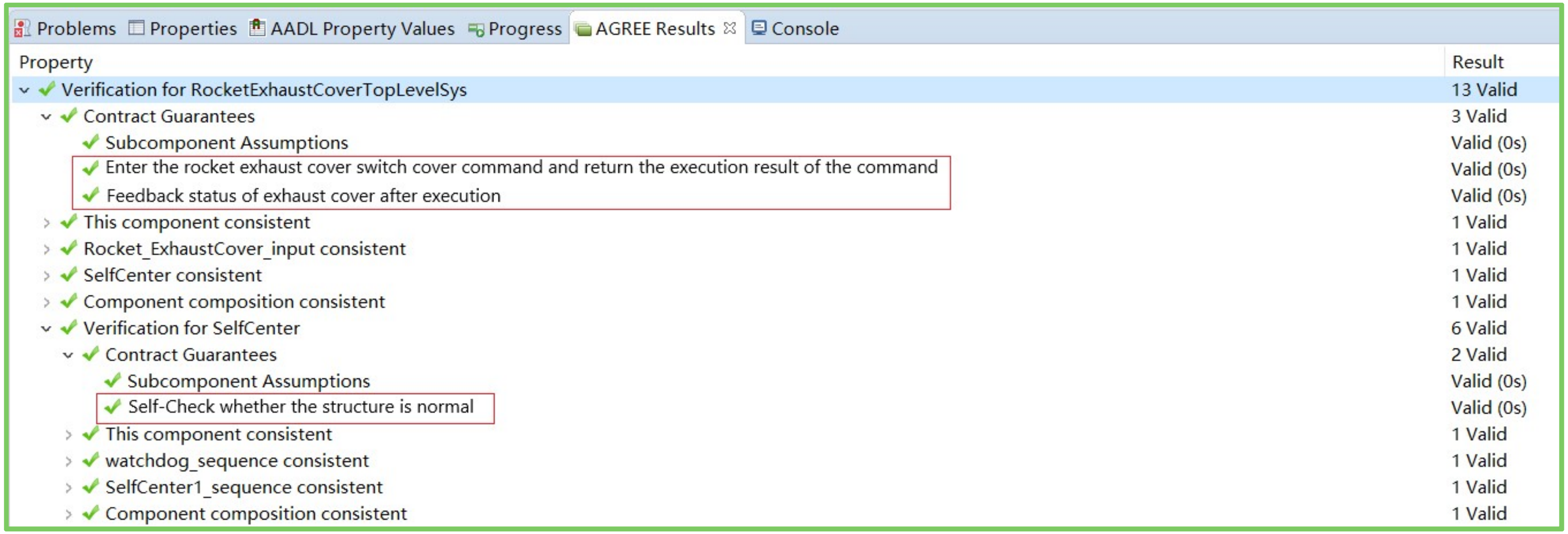
T4: Evaluation with case studies

✓ **Compositional Verification of the generated AADL models**

Property	Result
▼ ✔ Verification for Process_main.impl	70 Valid
▼ ✔ Contract Guarantees	8 Valid
✔ LCU_DM_NetRecv assume: A:LCU_DM_NetRecv	Valid (0s)
✔ FRAME_LCU_DT_NetDataAna assume: A:FRAME_LCU_DT_NetDataAna receive data from NetRecv	Valid (0s)
✔ LCU_DM_289A_Send assume: A:LCU_DM_289A_Send	Valid (0s)
✔ LCU_DM_289A_Recv assume: A:LCU_DM_289A_Recv	Valid (0s)
✔ FRAME_LCU_DT_289ASend assume: A:FRAME_LCU_DT_289ASend	Valid (0s)
✔ FRAME_LCU_DT_AnaOrder assume: A:FRAME_LCU_DT_AnaOrder	Valid (0s)
✔ LCU_CT_MainCont assume: A:LCU_CT_MainCont receive cont from NetDataAna	Valid (0s)
✔ Subcomponent Assumptions	Valid (0s)
▶ ✔ This component consistent	1 Valid
▶ ✔ LCU_DM_NetRecv consistent	1 Valid
▶ ✔ FRAME_LCU_DT_NetDataAna consistent	1 Valid
▶ ✔ LCU_DM_289A_Send consistent	1 Valid
▶ ✔ LCU_DM_289A_Recv consistent	1 Valid
▶ ✔ FRAME_LCU_DT_289ASend consistent	1 Valid
▶ ✔ FRAME_LCU_DT_AnaOrder consistent	1 Valid
▶ ✔ LCU_CT_MainCont consistent	1 Valid
▶ ✔ Component composition consistent	1 Valid
▶ ✔ Verification for LCU_DM_NetRecv	4 Valid
▶ ✔ Verification for FRAME_LCU_DT_NetDataAna	9 Valid
▶ ✔ Verification for LCU_DM_289A_Send	3 Valid
▶ ✔ Verification for LCU_DM_289A_Recv	3 Valid
▶ ✔ Verification for FRAME_LCU_DT_289ASend	4 Valid
▶ ✔ Verification for FRAME_LCU_DT_AnaOrder	4 Valid
▶ ✔ Verification for LCU_CT_MainCont	26 Valid

T4: Evaluation with case studies

✓ **Compositional Verification of the generated AADL models**

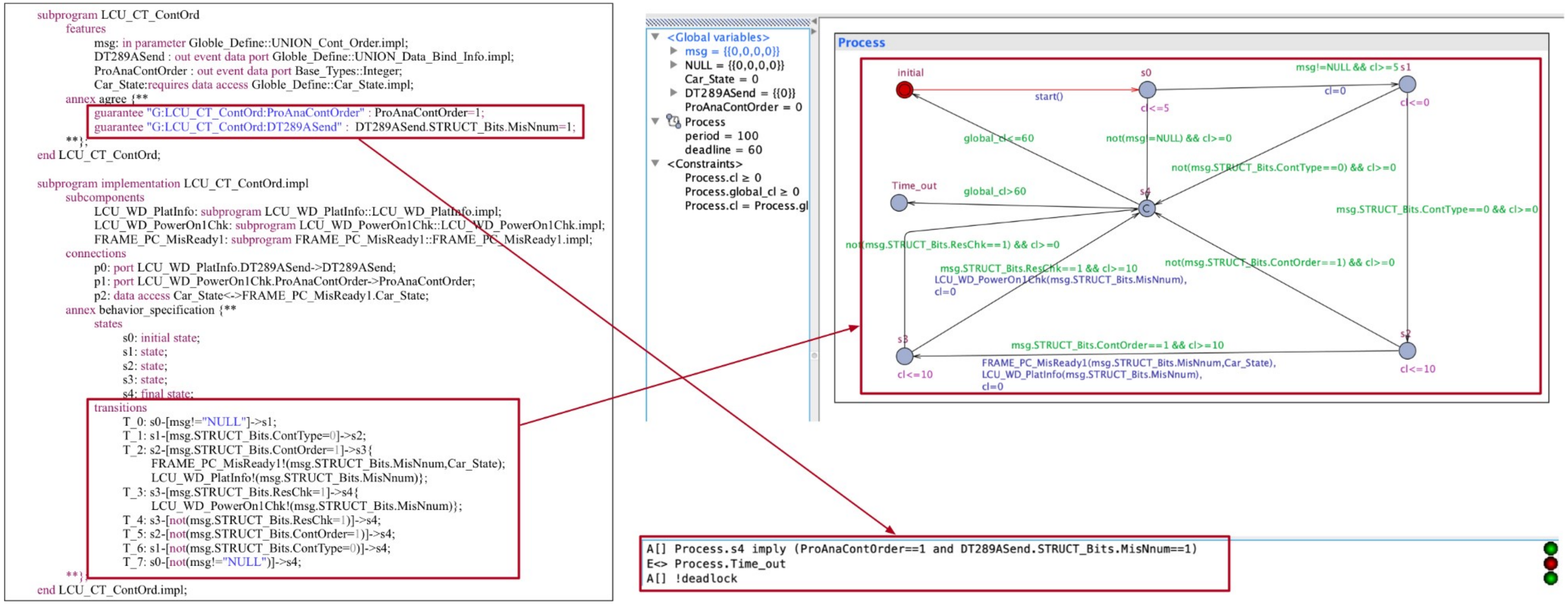


Problems | Properties | AADL Property Values | Progress | AGREE Results | Console

Property	Result
Verification for RocketExhaustCoverTopLevelSys	13 Valid
Contract Guarantees	3 Valid
Subcomponent Assumptions	Valid (0s)
Enter the rocket exhaust cover switch cover command and return the execution result of the command	Valid (0s)
Feedback status of exhaust cover after execution	Valid (0s)
This component consistent	1 Valid
Rocket_ExhaustCover_input consistent	1 Valid
SelfCenter consistent	1 Valid
Component composition consistent	1 Valid
Verification for SelfCenter	6 Valid
Contract Guarantees	2 Valid
Subcomponent Assumptions	Valid (0s)
Self-Check whether the structure is normal	Valid (0s)
This component consistent	1 Valid
watchdog_sequence consistent	1 Valid
SelfCenter1_sequence consistent	1 Valid
Component composition consistent	1 Valid

T4: Evaluation with case studies

- ✓ **Model checking of the leaf components**



T4: Evaluation with case studies

✓**Effectiveness**

- Three industrial case studies
- Two OS platforms

✓**Comparison with other MDRE tools**

Comparisons with a part of MDRE tools.

Tools	Parameters					
	Source Artifact	Target Model	Structural	Behavioral	Runtime	V&V
MoDisco	Java,JSP	UML	Y	N	N	N
fREX	Java	UML	N	Y	N	N
RE-CMS	PHP	AST of PHP	Y	N	N	N
Src2MoF	Java	UML	Y	Y	N	N
srcYUML	C++	UML	Y	N	N	N
Wang [13]	C	AADL	Y	N	N	N
C2AADL_reverse	C	AADL	Y	Y	Y	Y

Conclusion and Future work

- Conclusion:
  - An MDRE approach named C2AADL_Reverse: the transformation from multi-task C source code to AADL includes three parts: Structural, Behavioral and Run-time transformations.
  - Validation and verification approach of C2AADL_Reverse
  - The prototype tool
  - Industrial case studies
- Future work
  - We are currently working on the semantics preservation proof of C2AADL within Coq
    - Coq semantics of AADL synchronous fragments
    - Coq semantics of specific C multi-thread libraries
    - Semantic-preserving transformation from C to AADL
  - Compositional verification of the AADL asynchronous execution model (X-AGREE)

Thank you very much!