IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received December 8, 2018, accepted December 25, 2018, date of publication January 24, 2019, date of current version February 12, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2892745

Integrating Model Checking With SysML in Complex System Safety Analysis

HONGLI WANG, DEMING ZHONG, TINGDI ZHAO, AND FUCHUN REN
School of Reliability and Systems Engineering, Beihang University, Beijing 100191, China

Corresponding author: Tingdi Zhao (ztd@buaa.edu.cn)

This work was supported in part by Grants from the Civil Aviation Joint Funds Established through the National Nature Science Foundation of China and the Civil Aviation Administration of China under Grant U1533201, and in part by the Project of the Ministry of Industry and Information Technology of China under Grant JSZL2015601C008.

ABSTRACT Modern complex systems are characterized by numerous complex interactions and high levels of integration of functions, which present new challenges from the viewpoints of system safety analysis and design. Model checking can be employed to perform safety analysis, identify potential hazards, and prove the correctness of complex systems. However, many types of construction models are expressed in different ways, and there exists no unified model. Thus, the integration of model checking with system modeling language is proposed herein to analyze the safety of complex systems. System modeling language (SysML) is introduced to establish a unified system model that can describe a hybrid system of hardware and software but cannot be applied directly to safety analysis. Therefore, the semi-formal model SysML is transformed into the formal model new symbolic model checker/verifier, and the transformation rules are defined. The proposed unified model can not only help designers and safety and software engineers to execute various tasks but also efficiently, completely, and accurately analyze and verify the safety of complex systems. Finally, an integrated modular avionics case is presented to illustrate how to analyze the safety of complex systems. The results of the case study show that the proposed method can help increase the efficiency of safety analysis work and improve system safety.

INDEX TERMS Complex system, safety analysis, model checking, system modeling language (SysML), integrated modular avionics (IMA), potential hazard.

I. INTRODUCTION

With increasing system scale and growing functional requirements, modern complex systems tend to be highly integrated, incorporating all types of complex embedded components and functional structures of software and hardware coupling [1]. Inevitably, this brings greater difficulties and challenges to system safety analysis and design. The most representative of such systems is the Integrated Modular Avionics (IMA) system. The IMA system executes high levels of sharing and reuse of aircraft functions and resources through resource integration, functional fusion, and task synthesis, thus improving the efficiency of the avionics system. In addition, IMA increases system complexity and coupling, such as functional correlation and interaction of hardware and software, owing to its own characteristics of resource sharing. Meanwhile, it creates new types of hazards, such as logical contradictions or defects. Classic safety analysis techniques such as Failure Mode and Effect Analysis (FMEA), Fault tree analysis (FTA), and Hazard and Operability Analysis (HAZOP) [2]–[4], have been used to analyze the safety and reliability of complex systems for many years. However, these methods are not completely applicable to the analysis of modern integrated complex systems because they mainly identify the hazards associated with critical equipment or a limited set of components. Modern safety analysis methods (such as Functional Resonance Analysis Method [5], [6], System-theoretical Process Analysis [7]) can solve certain hazards to some extent, but it is easy to miss potential hazards in the system, thus reducing the accuracy and integrity of safety analysis results.

Formal methods have been developed in recent years to meet the challenges associated with analyzing and verifying complex systems [1]. The Model-Based Safety Analysis (MBSA) method has a more advanced model description capability and an automated analysis process that is more objective and efficient compared to traditional methods.

Scholars have introduced Model Checking [8], [9], a part of formal methods, to the safety analysis of complex systems [10]. Model checking can not only research all system states system mathematically by using the ergodic method but also use computer tools to perform safety analysis automatically. Moreover, model checking has been adopted widely in diverse fields, such as aerospace, nuclear, and train control. At present, a few powerful model checking tools are available. The common model checkers include Simple Promela Interpreter (SPIN) [11], Symbolic Model Verifier (SMV) [12], New Symbolic Model Checker/Verifier (NuSMV) [13], and UPPAAL [14], [15]. Although the formal method plays a very important role in validating the correctness of system design, modeling, and reasoning, its limitations lie in its use of specific model checker input languages for system model description in different model checkers and the frequent need to transform system models into specific automata languages. Moreover, the availability and applicability of different modeling language and validation tools differ. In the process of model transformation, it is difficult to guarantee model consistency, which makes the application of this method extremely difficult. And how model checking is much better for application in engineering is a difficult and necessary problem.

System modeling language (SysML) [16]–[18] is a graphical modeling language that is very intuitive and is widely used to model complex systems. Compared with Unified Modeling Language (UML) [19], SysML fully supports both hard/software systems and specific process information, is capable of modeling various problems of system engineering, and easy for engineers to learn. Although SysML is abundant in expression, it is not a high-level formal language that can be used for automatic analysis. Its graphical symbols often lack precise semantics and cannot be applied directly to verification and safety analysis, which makes it very difficult to formalize reasoning by using SysML. For example, there is no way to verify that a SysML diagram satisfies a given property. Moreover, model checking and verification based on SysML input is often difficult to master and operate in practical applications. To ensure the correctness of system design, modeling, and inference, it is necessary to formalize and verify the SysML model. Transformation of the SysML model into the corresponding formal model through model transformation guarantees system safety.

Most studies in the literature describe the transformation of traditional models into non-formal models [20], [21]. In recent years, the transformation of semi-formal models into formal models to perform analytical verification have attracted considerable attention. Scholars [22] have used the formalized description language Timed Automata (TA) to model, simulate, and verify systems. In [23], researchers added clock sterethpe to SysML and transformed SysML models into formal models (timed automa) for verification to improve software safety. More theoretical methods are usually less attractive for engineers and software developers. It has been suggested that SysML/UML be formatted as process algebra [24], [25] and Petri Nets [26], [27]. In [28], the development of a distributed reconfigurable control system was researched by combining semi-formal and formal methods. Model checking of hierarchical state machines was proposed in [29], such as formalization of the Kripke structure.

These more formal approaches are orthogonal works that go beyond the scope of providing a recipe for translating SysML/UML in terms of intuitive (intermediate) models, for the practical-mind. For a more detailed survey on model checking state charts, the authors of this study referred to [30].

In the present paper, an integrated model checking scheme based on SysML is proposed for application to complex system safety analysis. In terms of modeling methods, SysML is adopted to build system model and fully exploit the graphical functions of SysML. In terms of model transformation, a method for model transformation from SysML to NuSMV symbolic model checker input language is proposed, and transformation rules are defined. In terms of verification analysis, the formal analysis method is mainly adopted to verify and find defects from different aspects based on the NuSMV tool. Finally, by taking the IMA platform as an example, modeling and safety analysis are executed to realize automation from modeling, verification, to safety analysis.

The remainder of this paper is organized as follows. Section II is dedicated to the analysis process proposed in this study that integrates SysML and model checking. Section III introduces the system model method based on SysML. In section IV, the conditions and rules of translation from SysML to NuSMV model are proposed; then analysis and verification of system safety are performed. The applicability of the proposed technique is demonstrated by means of a case study of the IMA flaps control system in Section V. Finally, Section VI provides a few concluding remarks.

II. INTEGRATED ANALYSIS PROCESS OF SysML AND MODEL CHECKING

In the field of complex system safety, the model-based safety analysis method has increasingly attracted greater attention. System development activities such as simulation, verification, testing, and code generation can be organized around a formal system model. The integration between SysML and model checking is based on the use of the SysML model and model checking to achieve automatic and efficient safety analysis, which aims to exploit the intuition and usability of SysML, and the automatic analysis and verification capabilities of model checking.

The key steps involved in this process are illustrated in Fig. 1. The method starts with system modeling, as well as the construction of a system model and a set of safety specifications. This model can either be an early functional model of the system or an architectural model of the system depending on the stage of system development. SysML is applied to all stages of system development, including early

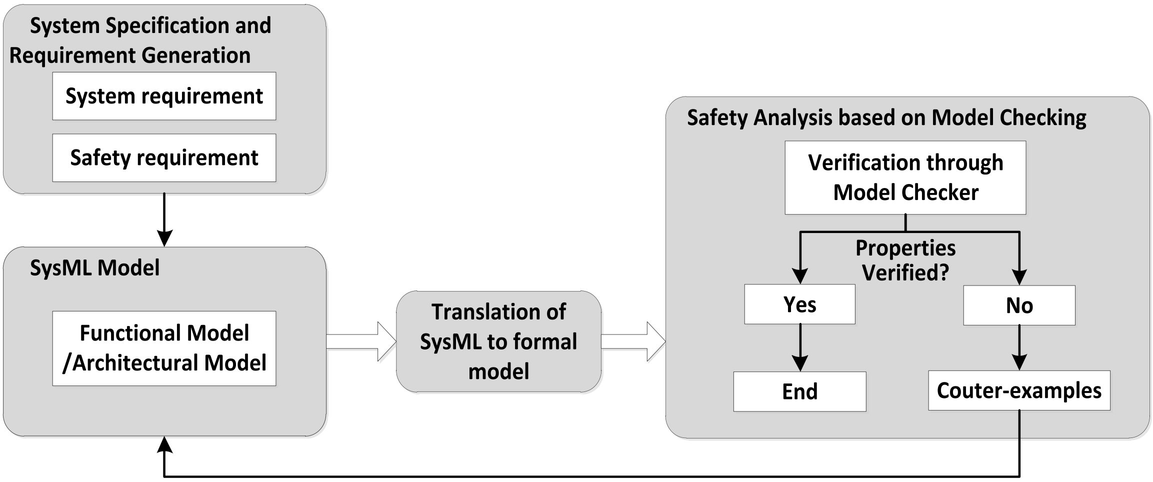

System Specification and Requirement Generation
System requirement
Safety requirement
SysML Model
Functional Model /Architectural Model
Translation of SysML to formal model
Safety Analysis based on Model Checking
Verification through Model Checker
Properties Verified?
Yes
No
End
Couter-examples

FIGURE 1. Integrated analysis process of SysML and model checking.

functional design, where design details are not mature. In the next step, this SysML model is formalized. The rules for transformation of the SysML model into the model checking model are determined, and the model checking model of the system is established. Model checking is then used to verify whether this dynamic system model conforms to safety requirements. If the conformity is verified, the process proceeds to either further refinement of the model and iteration of the above process or its implementation. Else, counter examples are produced to show how the model fails to fulfill certain requirements, and track, locate, modify, and re-verify the errors by using the counter examples.

III. SYSTEM MODEL BASED ON SysML

The OMG SysML is a general standard modeling language for systems that may include anything from hardware and software to staff and facilities. SysML can support the modeling of a variety of complex systems, including detailed description, analysis, design, validation, and verification. SysML adopts the graphical method to describe the system to improve the accuracy of description and reduce ambiguity and inconsistency in the description, as well as simultaneously enhance readability. SysML defines three types and nine subtypes of diagrams: 1) Requirement Diagram; 2) Structure Diagram: it comprises Package Diagram, Block Definition Diagram, Internal Block Definition Diagram, and Parametric Diagram; and 3) Behavior Diagram: it comprises Activity Diagram, Sequence Diagram, State MachilIe Diagram, and Use case Diagram.

On the one hand, SysML can be used for intuitive modeling of systems. On the other hand, SysML can be employed similarly to a meta-modeling language that defines the syntactic composition of the SysML modeling concepts considered by the proposed approach.

Take a traffic light control system as an example, as shown in Fig. 2. In this figure, the east-west and south-north lanes are one-way streets. Traffic lights are set in both lanes to ensure orderly driving of vehicles in both directions. There are two cars, namely car A from east to west and car B from south to north. They have two states: stop and move. Each signal light has two states: on and off. The SysML model of the traffic light control system is established as follows.

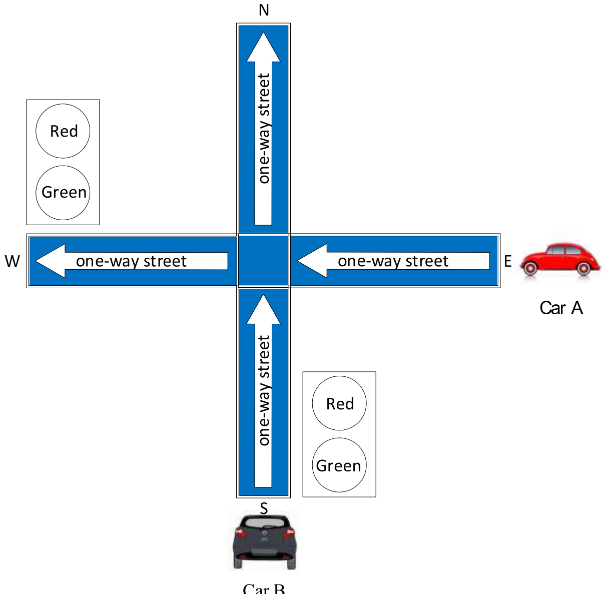


FIGURE 2. Traffic light control system.

A. BDD

Block Definition Diagram (BDD) is a structure diagram that mainly describes the structural composition of the system and the relationships among constituent elements. A block can include properties of certain types and references to other blocks. The BDD of the traffic light control system is shown in Fig. 3.

It can be seen from Fig. 3 that the traffic light control system is composed of traffic lights, car A, and car B.

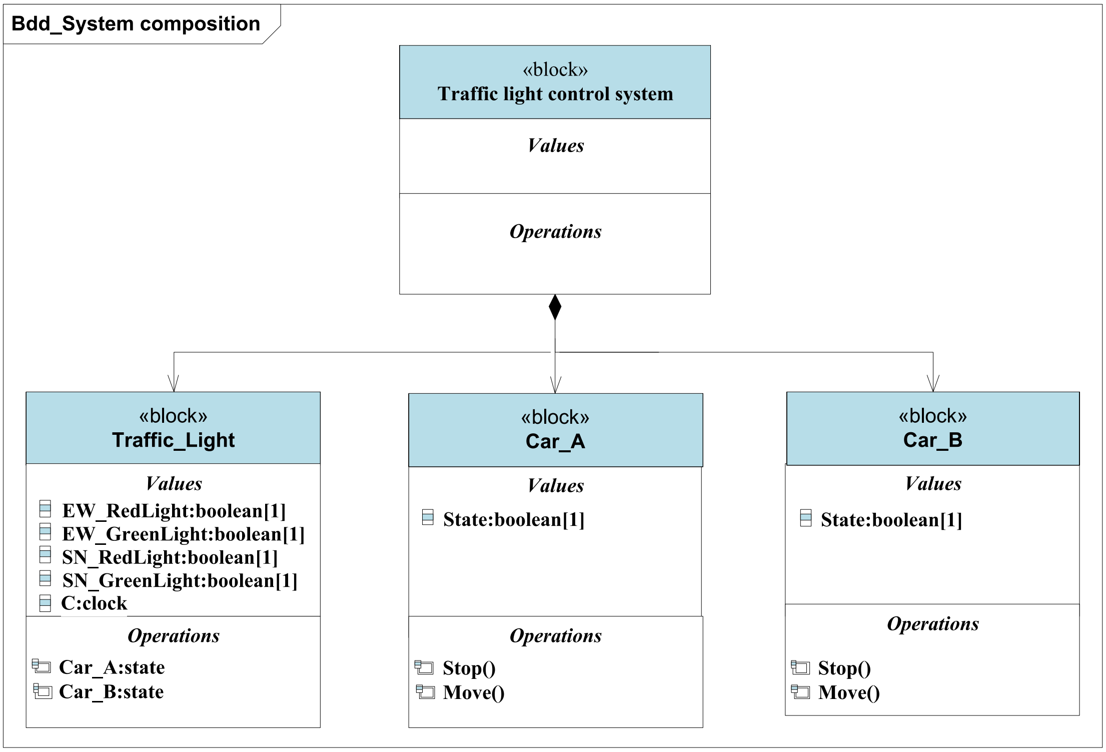


FIGURE 3. Block definition diagram of traffic light control system.

Each of the lights has a property value "state" to describe its own state. For instance, the light being on/off corresponds to the Boolean property in Fig. 3 being set to True/false in the BDD. The signal changes at intervals of 30 s, and a clock variable c is defined. In addition, the BDD can capture the relationships between blocks, such as correlations and dependencies.

The behaviors can be associated with the BDD through the properties of type StateChart. In Fig. 3, for instance, car A and car B are associated with a behavior via the "operation" property. At this point, it is important to mention that in the proposed approach, concurrent behavior is modelled by synchronizing multiple BDDs via events. Events occur in the context of triggers that specify points in the definition of a behavior at which some effect can be observed.

B. STM

State Machine Diagram (STM) is a state diagram used to model system behavior. The STM of the traffic light control system considered herein is shown in Figs. 4, 5, and 6.

STMs or statecharts, are a form of finite state automata used to model system behavior. States in an STM can express different statuses associated with the behavior of a system. For instance, a car being either stopped or moved is captured by two simple states "Car_A_stop" and "Car_A_move," respectively, in Figs. 4 and 5.

The invoking activities in a state are called "do" activities, and they are either continuous or discrete. The states are represented as an enclosed area of many behavioral fragments that can be executed simultaneously. Each region contains nested separated states and corresponding transitions. Therefore, the following types of composite states exist: 1) simple composite state-whenever the state contains exactly one region; and 2) orthogonal state-whenever the state

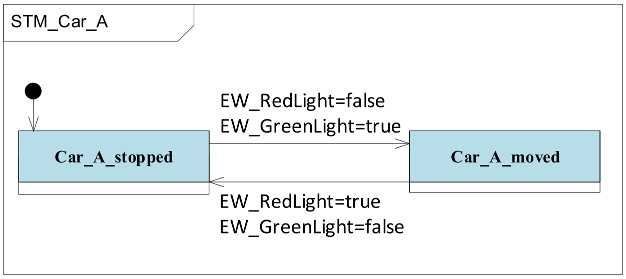


FIGURE 4. Car A behavior.

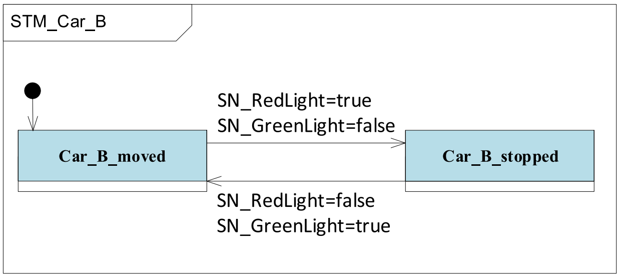


FIGURE 5. Car B behavior.

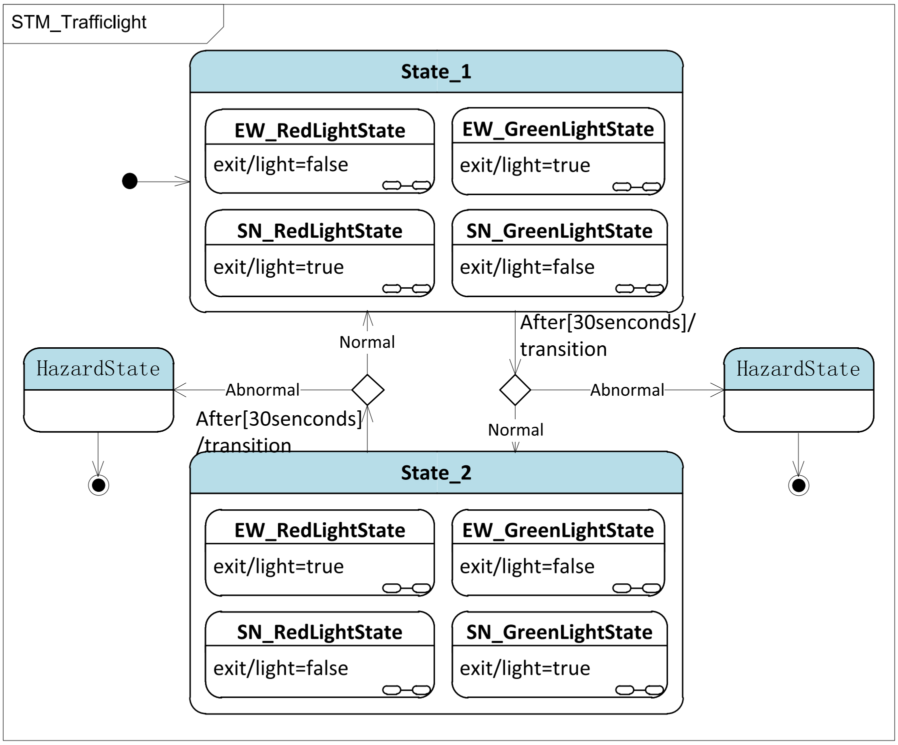


FIGURE 6. Traffic control light behavior.

TABLE 1. The elements and paths of STM and extended STM.

Element	Symbol
Starting point of state	●
Terminal point of state	◉
Intersection nodes	◇ [else] [guard]
Transfer Path	→
Extension of element	
Normal state	S State1
Hazard state	H State3

contains multiple regions. In this paper, only simple composite states are considered. A submachine state refers to an entire STM nested within the state.

The key elements and paths in STM and extended STM are summarized in Table 1.

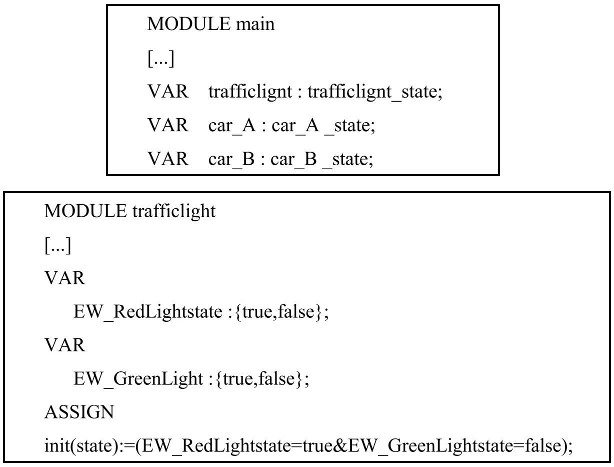
Notably, only the BDD and the STM in SysML are employed in this study.

IV. TRANSLATION OF SysML TO MODEL CHECKING

Many technical tools are available for model checking. In the present study, the NuSMV model checker is employed as a good example of this technology to promote comprehensive automatic analysis of SysML. NuSMV is fundamentally a symbolic model checker, and are recommended for larger real-life complex systems. NuSMV can support the description of temporal logic computation tree logic (CTL), also descripts linear temporal logic (LTL). The NuSMV tool is an opensource program, which allows it to be tailored more effectively into a future integrated support tool [1].

The New Symbolic Model Verifier (NuSMV) [13] is a symbolic model checking tool that which checks a finite state system against specifications in CTL and LTL, by using BDD-based and SAT-based model checking techniques. In this section, the we focus is on the parts of the NuSMV input language relevant to the present for our work. For a thorough description of NuSMV inputs, we refer the interested readers may refer to to the user manual in the distribution package of the NUSMV model checker.

Intuitively, a NuSMV program consists of a list of modules that are further instantiated into so-called processes that model interleaving. A "process" has a special Boolean variable associated with it called "running," the value of which is true if and only if the corresponding process instance is currently selected for execution. Each module is associated an identifier and a series of parameters. The body of a module consists of elements that can denote variable declarations, variable initializations/assignments,



```
MODULE main
[...]
VAR   trafficlignt : trafficlignt_state;
VAR   car_A : car_A _state;
VAR   car_B : car_B _state;
```

```
MODULE trafficlight
[...]
VAR
    EW_RedLightstate :{true,false};
VAR
    EW_GreenLight :{true,false};
ASSIGN
init(state):=(EW_RedLightstate=true&EW_GreenLightstate=false);
```

FIGURE 7. Corresponding NuSMV module and variable declaration.

LTL specifications or, for instance, behaviors defined based on transitions. Transitions are introduced by the "TRANS" keyword, followed by a Boolean expression expressing whether two states belong to the transition relation. Therefore, the aforementioned Boolean expression can include the "next" operator to relate the current and the next state variables and express transitions in the state-machine corresponding to the behavior of the module.

A. TRANSLATION PROCESS OF BDD

The NuSMV model is composed of MODULE main and several submodules MODULE n. The MODULE main describes the constitution of the system, as well as introduces all the submodules and the properties to be verified. MODULE main in NuSMV contains a list of module declarations given by the BDD component in the SysML model. Each submodule in NuSMV corresponds to an element in the BDD of SysML. All variables (properties) are then declared in MODULE main. These attributes are further initialized by the initial values of the associated elements in BDD. If they are not, they are denoted by as the default values. The configuration is executed in the corresponding module of each variable. Taking the traffic light control system as an example, the corresponding NuSMV module and variable declaration are shown in as follows in Fig. 7.

B. TRANSLATION PROCESS OF STM

The translation process of STM is less straightforward. In the NuSMV code, the state itself is integrated into the translation system. In the present study, sState behaviors in our study areis converted to variable changes duringwhen processing.

1) STATUS

S1: Add a status variable in the Module corresponding to the event in STM; In NuSMV, there will be:

MODULE Car_A
VAR
  status: {move, stop};

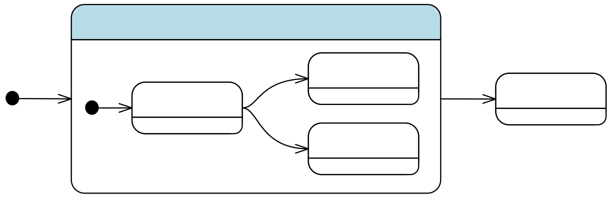


FIGURE 8. A simple example of STM.

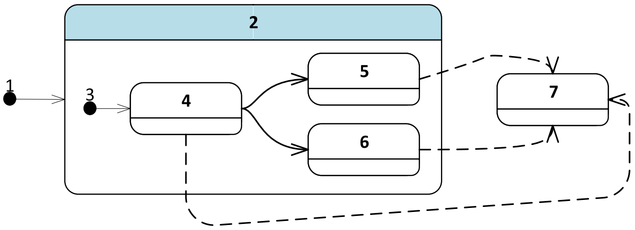
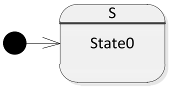


FIGURE 9. STM with numbered states and distributed transitions.

S2: Initial state in STM, that is, the initial state of status.

Use  as the initial value of status. Thus, there are:

ASSIGN
  init(status):= state0;

S3: Generate enumerated type status in NuSMV, and the status set will be translated to:

status: {state0, state1, state2, . . ., staten};
substatus: {sub1, sub2, sub3, . . ., subn};

Moreover, the states in STM have a hierarchical structure. In this study, they are simple composite states. A simple state diagram is shown in Fig. 8. In Fig. 9, for example, values 1, 2, and 6 are assigned to the states of the STM by using recursive numbering.

2) EVENT

S4: The "event" is translated into Boolean variables.

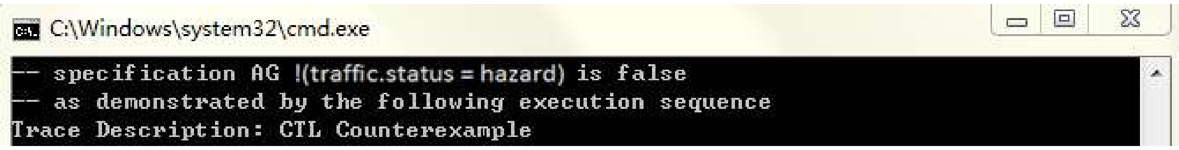
See, for instance, the variable declaration **VAR EW_RedLightstate: Boolean**; in MODULE *main*. Its value is set to *true* when a state or transition includes a trigger for the event in its behavior, or to *false* after execution of a transition that requires the event to be enabled.

3) TRANSITION

The transition structure in NuSMV is introduced via the *TRANS* keyword, followed by a Boolean statement.

S5: Each transition of STM is the value assignment of each status, which is a transition to *next* of status. Ignore the "event" name for each transition.

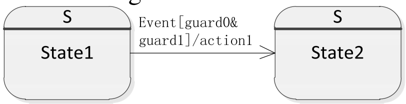
S6: The status before transition and the critical conditions "Guard" are the condition of *next(status)*. The status set before transition is the result of *next(status)*.



```
-- specification AG !(traffic.status = hazard) is false
-- as demonstrated by the following execution sequence
Trace Description: CTL Counterexample
```

FIGURE 10. The analysis results of traffic control system.

S7: If the transition in STM has parameter change "action", each parameter change is transitioned as a variable transition *next*.

According to the above rules, *Transition* 

is translated to:

TRANS

next(status) :=

case

status = state1 &g uard0&guard1: {state2};

1: status;// Means if the condition is unconformity, the status unchanged;

esac;

C. SAFETY ANALYSIS OF TRAFFIC CONTROL SYSTEM

By using the above transformation rules, the SysML model is converted into a NuSMV model. As a demonstration, this example of traffic control system takes "the traffic control system has no hazard" status as the safety requirement, and converts it into the following CTL expression: *AG !(traffic.status = hazard).* Figure 10 is obtained after automatic analysis and verification.

The counter examples are obtained, the system does not hold its specifications. The number of reachable states is 142, and one of them is picked for analysis, namely, "EW_Redlightstate=false & EW_Greenlightstate=true & SN_Redlightstate=false & SN_Greenlightstate=true & CarA_state=move & Cae_B_state=move," in which the likelihood of an accident occurring is high. To avoid this problem, the control design logic is modified, for instance, two green lights do not light up simultaneously, which improves system safety.

Notably, translations of unnecessary information and nonformal content are ignored in this section.

V. CASE STUDY: INTEGRATED MODULAR AVIONICS

This case study illustrates application of the proposed integrated method to the IMA [31], [32] system. Now, considering the novelty of avionics architecture, the efficiency of IMA, universally considered a safety-critical and software-intensive system, can be effectively improved via weight and power consumption reduction by means of comprehensive resource integration or high-level resource sharing compared with traditional avionics [33], [34]. However, a new series of problems or potential hazards emerges with the high complexity, for example, the number of faults increases, and the system becomes prone to fault propagation. For analysis purpose, in this section, an IMA application system called the flap control system is considered.

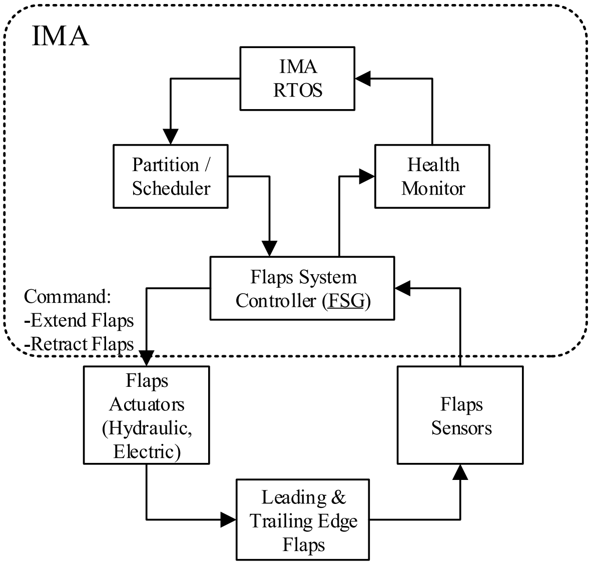

IMA
IMA RTOS
Partition / Scheduler
Health Monitor
Flaps System Controller (FSG)
Command:
-Extend Flaps
-Retract Flaps
Flaps Actuators (Hydraulic, Electric)
Flaps Sensors
Leading & Trailing Edge Flaps

FIGURE 11. IMA flaps control structure.

The flap control system is composed of the IMA platform (including operating system) and an application. Figure 11 shows a functional control structure diagram of an IMA flap system.

A. FLAP CONTROL SYSTEM MODEL BASED ON SysML

For the flap control system, the objects involved in the system are described and modeled in combination with BDD and STM of the SysML. Figure 12 shows a BDD of the system in which the structures of the objects in the entire system are constructed, and the relationships among the objects are defined.

As shown in Fig. 12, the IMA platform application system is composed of the IMA platform and the flap system applications, where the main functions of the IMA platform are partition management, health monitoring, and hardware resource management. The main task of IMA platform is to provide the required resources for applications running on the IMA platform. The real-time operating systems (RTOS) controls partitions and hardware resources. The health monitor controls hardware modules, and the partitions manage hardware resources. The flap system consists of a flap controller, physical flap, mechanical system, hydraulic system, and sensor.

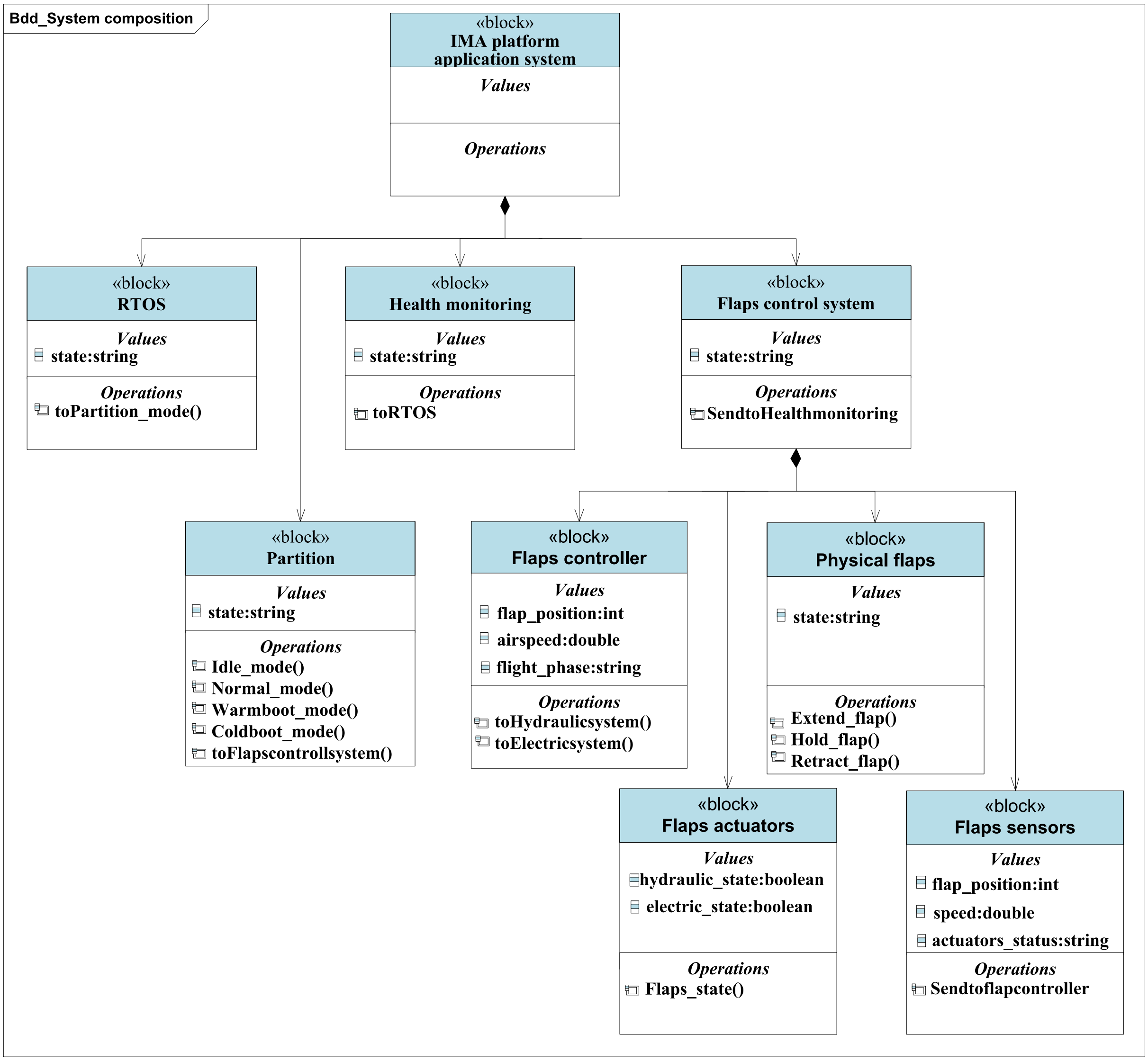


FIGURE 12. Block definition diagram of flaps control system.

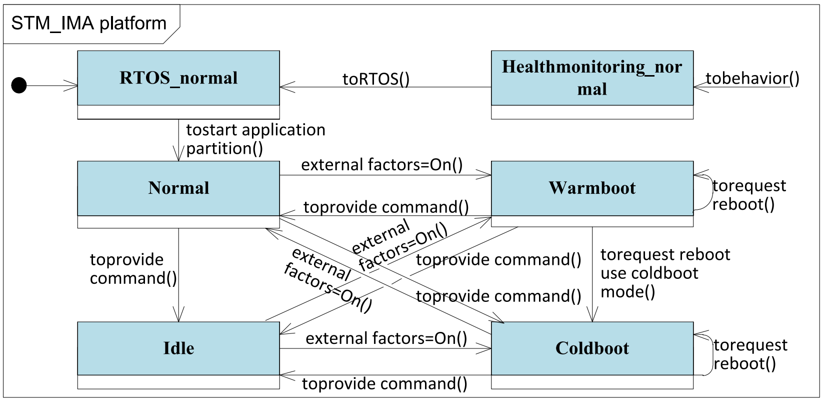


FIGURE 13. State machine diagram of IMA platform.

Figures 13 and 14 show the STM of the flap control system. Figure 13 shows the STM of the IMA platform. When the RTOS starts the application partition, the partition goes into the normal operation mode. Partition management is performed by the operating system. The partition has four operation modes: idle, normal, cold boot, and warm boot.

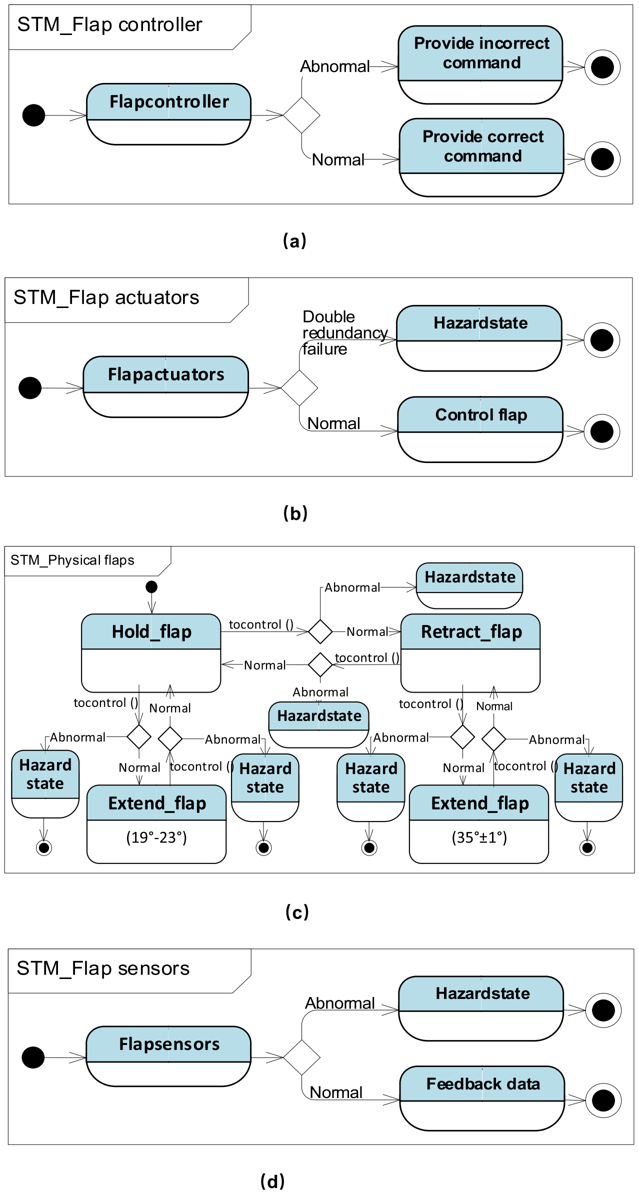
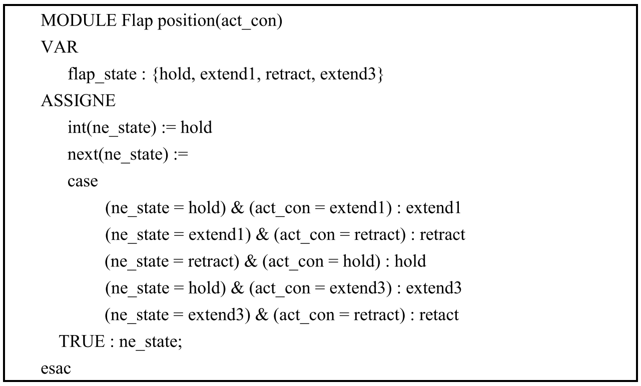


FIGURE 14. State machine diagram of flaps control system. (a) Flap controller behavior. (b) Flap actuators behavior. (c) Physical flaps behavior. (d) One of flap sensors behavior.

Under the IMA architecture, the flap system controller is an application running on the IMA platform, and it controls retraction and extension of the flaps. Figure 14(a) shows changes in the flap controller state. The flap controller controls coordination of the electrical control system and the mechanical drive system, lowering the flap to the desired angle, and providing lift or drag to the aircraft. The flap controller receives data from the flap sensors and other system components.

Figure 14(b) shows changes in the states of the flap actuators. The hydraulic and electrical systems receive and execute commands from the flap controller to change the positions of the physical flaps. Both hydraulic and electrical systems adopt a double redundancy design to avoid malfunction in the event of a fault. Figure 14(c) shows changes in the states of the physical flaps.



```
MODULE Flap position(act_con)
VAR
    flap_state : {hold, extend1, retract, extend3}
ASSIGNE
    int(ne_state) := hold
    next(ne_state) :=
    case
        (ne_state = hold) & (act_con = extend1) : extend1
        (ne_state = extend1) & (act_con = retract) : retract
        (ne_state = retract) & (act_con = hold) : hold
        (ne_state = hold) & (act_con = extend3) : extend3
        (ne_state = extend3) & (act_con = retract) : retact
   TRUE : ne_state;
esac
```

FIGURE 15. The example of physical flap state transformation.

Figure 14(d) shows changes in the states of the flap sensors. The flap sensors transmit flap status data to the controller, and the flap controller receives data from other system components.

B. FLAP CONTROL SYSTEM MODEL BASED ON SysML MODEL TRANSFORMATION

The established SysML model is transformed into the formal description of NuSMV according to the conversion rules in Section IV. Taking the physical flap as an example to illustrate model transformation, the corresponding NuSMV model is shown in Fig. 15.

To realize normal operation of the IMA flap control system, the corresponding system safety requirements in this case are expressed using a temporal logic formula as follows:

$$CTLSPEC\ AG\ FCS.status\ != fail.$$

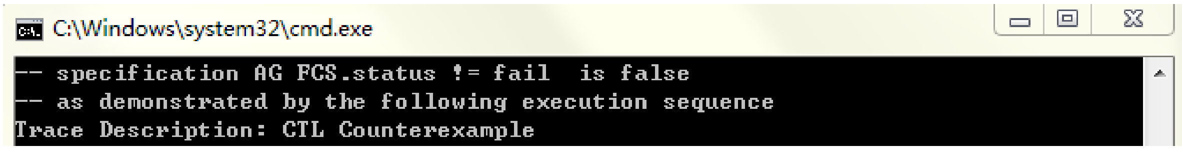
Verification of these properties will help highlight the ability to verify safety properties.

C. SAFETY ANALYSIS

NuSMV was used for automatic model checking, and the safety analysis results are shown in Fig. 16.

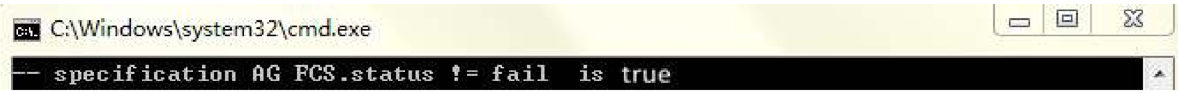
The results show that the system does not meet the safety requirements, which may lead to accidents. In this case, NuSMV successfully identified counter examples of the systems violating safety requirements. Even though the number of attainable states that caused hazard was approximately 50,000, screening was performed to eliminate events that that did not cause hazard. Through analysis of the counter-example path, it was found that the problem lay in the logic design of the controller, which resulted in inconsistent variable parameter values. The verification results obtained after automatic modification of the control logic of flap retraction are shown in Fig. 17.

Based on the above findings, integrating model checking with SysML in the context of complex system safety analysis can help find defects in system design and notify the designer



```
-- specification AG FCS.status != fail  is false
-- as demonstrated by the following execution sequence
Trace Description: CTL Counterexample
```

FIGURE 16. The verification results of NuSMV.



```
-- specification AG FCS.status != fail  is true
```

FIGURE 17. The verification results of NuSMV after modifing.

to make corresponding modifications to improve model correctness. Notably, the example and modification considered herein are hypothetical and simplified. The example was used to illustrate application of the proposed integration application, and its detailed design is not included. The example shows how the proposed method can help design, develop, and execute safety assessment processes to achieve more robust and fault-tolerant designs.

VI. CONCLUSIONS

In this paper, integration of model checking with SysML for complex system safety analysis was proposed. This method extends model checking with system modeling language to ensure model consistency and improve usability in engineering. The use of SysML, which has been employed widely for modeling complex systems to perform system modeling, makes it conducive for designers, analysts, and suppliers to use the unified model. Then, the semi-formal model SysML is transformed into the formal model NuSMV, which is used to perform safety analysis and verification.

The results of this study demonstrate the procedure of transforming SysML into model checking by using a simple IMA flap control system. The unified model was established through SysML, and safety analysis and verification were performed using model checking. Defects in product development and verification, especially in system design, can be found using the proposed method, and the designer can be notified to make corresponding modifications to improve system safety.

The findings of this paper open up a new paradigm in complex system safety analysis and the discovery of potential hazards; this improves the efficiency of safety analysis work. The proposed method must be researched further, for example, transformation of other types of category diagrams in SysML and exploration of transformation tool of the proposed method. In addition, replace NuSMV with other model checkers such as SPIN or UPPAL is considered, this to check out whether the results would vary.

REFERENCES

[1] S. Sharvia and Y. Papadopoulos, "Integrating model checking with HiP-HOPS in model-based safety analysis," *Rel. Eng. Syst. Saf.*, vol. 135, pp. 64–80, Mar. 2015.

[2] C. A. Ericson, *Hazard Analysis Techniques for System Safety*. Hoboken, NJ, USA: Wiley, 2005, pp. 11–34.

[3] *Guidelines for Development of Civil Aircraft and Systems*, Standard ARP4754A, 2010.

[4] *Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment*, Standard ARP4761, 1996.

[5] E. Hollnagel and O. Goteman, "The functional resonance accident model," *Cognit. Syst. Eng. Process Plant*, pp. 155–161, Nov. 2004.

[6] P. V. R. de Carvalho, "The use of functional resonance analysis method (FRAM) in a mid-air collision to understand some characteristics of the air traffic management system resilience," *Rel. Eng. Syst. Saf.*, vol. 96, no. 11, pp. 1482–1498, 2011.

[7] N. G. Leveson, *Engineering a Safer World: Systems Thinking Applied to Safety*. Cambridge, MA, USA: MIT Press, 2011, pp. 1–33.

[8] E. M. Clarke and J. M. Wing, "Formal methods: State of the art and future directions," *ACM Comput. Surv.*, vol. 28, no. 4, pp. 626–643, 1996.

[9] E. M. Clarke, Jr., O. Grumberg, and D. Peled, "Model checking," *Foundations of Software Technology and Theoretical Computer Science*. Berlin, Germny: Springer, 1997.

[10] M. Bozzano *et al.*, "ESACS: An integrated methodology for design and safety analysis of complex systems," in *Proc. 14th Eur. Saf. Rel. Conf.*, 2003, pp. 237–245.

[11] G. Holzmann, *The SPIN Model Checker: Primer and Reference Manual*. Reading, MA, USA: Addison-Wesley, 2004.

[12] K. Y. Koh and P. H. Seong, "SMV model-based safety analysis of software requirements," *Rel. Eng. Syst. Saf.*, vol. 94, no. 2, pp. 320–331, 2009.

[13] A. Cimatti, E. Clarke, F. Giunchiglia, and M. Roveri, "NuSMV: A new symbolic model verifier," *Computer Aided Verification*. Berlin, Germany: Springer, 1999.

[14] K. G. Larsen, P. Pettersson, and W. Yi, "UPPAAL in a nutshell," *Int. J. Softw. Tools Technol. Transfer*, vol. 1, no. 1, pp. 134–152, 1997.

[15] G. Behrmann, A. David, and K. G. Larsen, "A tutorial on UPPAAL," in *Proc. Int. School Formal Methods Design Comput. Commun. Softw. Syst.*, vol. 4, no. 12, 2004, pp. 200–236.

[16] *System Modelling Language (SYSML) Specification*, OMG, document ad/2006.3.1.

[17] S. Friedenthal, "Systems modeling language (SysML)," *Insight*, vol. 7, no. 3, p. 20, 2004.

[18] L. Balmelli, "An overview of the systems modeling language for products and systems development," *J. Object Technol.*, vol. 6, no. 6, pp. 149–177, 2007.

[19] J. Rumbaugh, I. Jacobson, and G. Booch, *The Unified Modeling Language Reference Manual*. 2006.

[20] T. Zhang, F. Jouault, J. Bézivin, and X. Li, "An MDE-based method for bridging different design notations," *Innov. Syst. Softw. Eng.*, vol. 4, no. 3, pp. 203–213, 2008.

[21] Y. Zhu, Z. Q. Huang, Z. Cao, H. Zhou, and M. Yuan, "A method for generating software architecture model based on formal specifications," Tech. Rep., 2010, pp. 2738–2751, vol. 21, no. 11.

[22] K. Larsen *et al.*, "As cheap as possible: Effcient cost-optimal reachability for priced timed automata," in *Computer Aided Verification* (Lecture Notes in Computer Science), vol. 2102. 2001, pp. 493–505.

[23] Y. Y. Song, "Research on formal transformation of SysML model for verification," Nanjing Univ. Aeronaut. Astronaut., Nanjing, China, Tech. Rep., 2012.

[24] H. H. Hansen, J. Ketema, B. Luttik, M. R. Mousavi, and J. van de Pol, "Towards model checking executable UML specifications in mCRL2," *Innov. Syst. Softw. Eng.*, vol. 6, nos. 1–2, pp. 83–90, 2010.

[25] T. Ando, H. Yatsu, W. Kong, K. Hisazumi, and A. Fukuda, "Translation rules of SysML state machine diagrams into CSP# toward formal model checking," *Int. J. Web Inf. Syst.*, vol. 10, no. 2, pp. 151–169, 2014.

[26] E. Andrade, P. Maciel, G. Callou, and B. Nogueira, "A methodology for mapping SysML activity diagram to time Petri Net for requirement validation of embedded real-time systems with energy constraints," in *Proc. 3rd Int. Conf. Digit. Soc.*, Feb. 2009, pp. 266–271.

[27] C. Ermel, "Visual modelling and analysis of model transformations based on graph transformation," *Math. Struct. Comput. Sci.*, vol. 24, no. 99, pp. 135–152, 2012.

[28] R. Oueslati, O. Mosbahi, M. Khalgui, Z. Li, and T. Qu, "Combining semi-formal and formal methods for the development of distributed reconfigurable control systems," *IEEE Access*, vol. 6, pp. 70426–70443, 2018.

[29] R. Alur and M. Yannakakis, "Model checking of hierarchical state machines," *ACM Trans. Program. Lang. Syst.*, vol. 23, no. 3, pp. 175–188, 2001.

[30] P. Bhaduri and S. Ramesh. (2004). "Model checking of state-chart models: Survey and research directions." [Online]. Available: https://arxiv.org/abs/cs/0407038

[31] P. J. Prisaznuk, "Integrated modular avionics," in *Proc. Nat. Aerosp. Electron. Conf.*, May 1992, pp. 39–45.

[32] C. B. Watkins and R. Walter, "Transitioning from federated avionics architectures to integrated modular avionics," in *Proc. IEEE/AIAA 26th Digit. Avionics Syst. Conf.*, Oct. 2007, pp. 2.A.1-1–2.A.1-10.

[33] C. B. Watkins, "Integrated modular avionics: Managing the allocation of shared intersystem resources," in *Proc. IEEE/AIAA 25th Digit. Avionics Syst. Conf.*, Oct. 2006, pp. 1–12.

[34] T. Zhou, H. Xionq, and Z. Zhang, "Hierarchical resource allocation for integrated modular avionics systems," *J. Syst. Eng. Electron.*, vol. 22, no. 5, pp. 780–787, Oct. 2011.

Authors' photographs and biographies not available at the time of publication.