35th European Modeling & Simulation Symposium
20th International Multidisciplinary Modeling & Simulation Multiconference

2724-0029 © 2023 The Authors.
doi: 10.46354/i3m.2023.emss.013

An integrated methodology to system design and simulation for electric vehicles based on X language

Pengfei Gu[1,2], Yuteng Zhang[1,2], Zhen Chen[1,2] and Lin Zhang[1,2,*]

[1]School of Automation Science and Electrical Engineering, Beihang University, Xueyuan Road No.37, Haidian, Beijing, 100191, China
[2]Complex System Engineering Center of the Ministry of Education, Xueyuan Road No.37, Haidian, Beijing, 100191, China

[*]Corresponding author. Email address: zhanglin@buaa.edu.cn

Abstract

To design and develop electric vehicles, traditional model-based system engineering methodologies need to combine system-oriented SysML and physics-oriented Modelica or Simulink. However, cross-language and cross-platform integration design can lead to model inconsistencies and increase the learning cost for modelers, while also reducing system development efficiency. To address these issues, this paper proposes the use of X language for model-based systems engineering (MBSE) to realize integrated modeling and simulation of functional and physical characteristics in the linear driving scenarios of electric vehicles. Based on the analysis of the requirements for this scenario, an integrated model is established using X language, including a logical model for system architecture based on functional requirements and a physical model based on performance requirements, and verified in XLab. This X language-based integrated methodology to system design and simulation can quickly verify the coordination of functional and performance requirements for electric vehicles. It can also help designers better identify system design problems holistically and enable them to make timely corrections. This X language-based integrated methodology to system design and simulation provides a new theoretical and methodological reference for designers in different fields.

Keywords: X language; Model-based system engineering(MBSE); Electric vehicles; Modeling & Simulation

1. Introduction

Complex electromechanical products involve various fields and disciplines, such as mechanics, electronics, hydraulics, and control, which pose significant challenges in their design due to the tight couple between different fields and the increasing scale of the products (Lin et al., 2010). Model-based systems engineering (MBSE) provides an effective solution for designing and developing complex systems by delivering information through a unified and standardized model throughout the design and development process (Ramos et al., 2011). The mainstream modeling language for MBSE is currently SysML, which supports graphical modeling in three dimensions based on nine diagrams (Friedenthal et al., 2008).SysML is currently supported by Rhapsody, a software developed by IBM, and Cameo Systems Modeler (CSM), a commercial software from Dassault. In terms of modeling methods, Harmony-SE (Douglass, 2005) for Rhapsody and MagicGrid for Cameo systems modeler are widely used (Morkevicius et al., 2017). However, SysML lacks precise semantic support to describe the physical mechanisms of complex systems. It primarily focuses on logical verification of system architecture, limiting its ability to address deeper design issues. Therefore, simulation techniques need to be incorporated into MBSE to verify system designs and provide feedback for model improvements. This integration of system design and simulation is crucial for enhancing the efficiency of developing complex products.

This paper proposes a methodology for system design and simulation integration in X language, a new generation of integrated modeling and simulation language, to achieve the integration of system design and simulation. The methodology provides a framework for unifying requirements, functional, logical, and physical modeling, and simulation, with integrated system design and simulation capabilities. The methodology is applied to designing and verifying pure electric vehicles in linear driving scenarioss.

This paper's subsequent work summarizes and analyzes the status of existing system design and simulation integration methods and their problems in Section II. Section III gives a general overview of X language and proposes an integrated design and simulation methodology based on X language. Section IV describes the integrated design and simulation methodology based on X language for designing and verifying pure electric vehicles under the linear driving scenario in detail. The paper concludes by summarizing and validating the study.

2. Related Works

The current research work for system-level design and simulation integration is divided into two main types: one is co-simulation. Based on the model integration standard interface FMI realizes the joint simulation of the system logical behavior model constructed by SysML and the system multidisciplinary physical behavior model constructed by a specific modeling simulation language. Secondly, model conversion. Based on the SysML extension mechanism, the extension package for specific modeling simulation language is constructed to realize the automatic conversion from system design to simulation. Regarding co-simulation, (Feldman et al., 2014) achieved the combined simulation of the system's logical design model constructed using SysML and the physical model constructed using Modelica through FMI, offering guidance for co-simulating continuous and discrete hybrid models. (Jieshi et al.) compiled the SysML model into .fmu by FMI (functional mock-up interface) to realize the integration with Simulink and Modelica, which helps the unit design to be co-simulated in advance and realize the testing process in advance. However, the integration method based on FMI only integrates the system-level design model and physical-level simulation model from the data level, but the system-level design model and physical-level simulation model are still based on different languages and software implementations, which makes it difficult to ensure the consistency of the model. In terms of model conversion, W. Schamai (Schamai, 2009; Schamai et al., 2010) proposed the first more detailed method for system design and simulation integration, i.e., simulation mapping to Modelica by extending UML, but its method does not support the full Modelica syntax standard. In 2012, OMG (integration working group et al.) proposed a SysML and Modelica mapping method based on QVT and provided the definition of the SysML4Modelica extension package for metamodel conversion, which provides a new idea for mapping between SysML and Modelica, but the definition of this extension is still not perfect and the QVT method used does not support bidirectional conversion between models. Shuhua Zhou (Shuhua et al., 2014) et al. implemented the mapping from SysML to Modelica based on ATL and gave the definition of the M-Design extension package for converting metamodels. The M-Design extension package defines a relatively complete Modelica syntax compared to the SysML4Modelica extension package, but there are still relevant metamodels that are not defined, such as state machine, etc. Wu (Xinquan et al., 2023) proposed a model conversion method for simulating hybrid SysML models in the framework of the Discrete Event System Specification (DEVS), constructed a DEVS-oriented SysML outline file and a DEVS metamodel including discrete, hybrid, and coupled models, and implemented the ATL-based The automatic conversion of SysML to DEVS is implemented based on ATL, and the SysML model with the continuous and discrete hybrid system is verified using the simulator of DEVS. However, it only achieves one-way conversion from SysML to DEVS. In conclusion, the model conversion approach can achieve seamless integration of system design and simulation from the theoretical level, but it is limited by the significant gap in syntax and semantics between system modeling languages (UML, SysML, etc.) and simulation languages (Modelica, Simulink, etc.), which cannot be fully bridged by the metamodel extension mechanism. Also, it needs to be implemented through different languages and software integration. This also poses a huge challenge to the consistent maintenance of the models. To truly realize the integration of design and simulation, an integrated MBSE methodology with complete support for requirements analysis, functional analysis, system design, and verification needs to be fundamentally constructed.

3. X language and integrated methodology to system design and simulation

3.1. X language

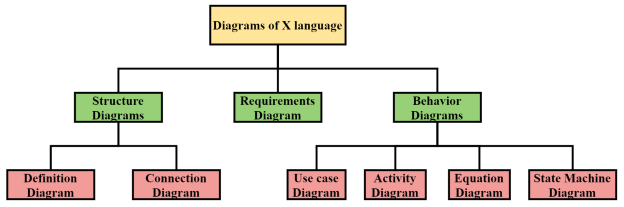


Figure 1. Diagrams of the X language

The X language is a new generation of integrated modeling and simulation language for complex systems sup-

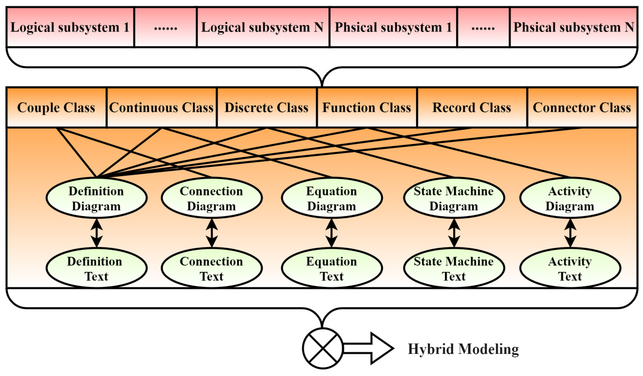


Figure 2. A hybrid modeling framework for the X language.

porting MBSE based on the hybrid system modeling specification XDEVS (Xie et al., 2022) and borrowed from SysML and Modelica modeling concepts (Zhang et al., 2021, 2022a,b; Pengfei et al., 2022). The X language is an object-oriented language that defines six specific classes (continuous class, discrete class, couple class, etc.) and two forms (graphical modeling and textual modeling) to realize integrated modeling and simulation of system architecture and physical mechanism, which provides the basis for fully supporting the integration of MBSE.

As shown in Figure 1, in order to fully support MBSE, X language defines seven types of diagrams, which are requirement diagram, use case diagram, definition diagram, connection diagram, equation diagram, activity diagram, and state machine diagram. Among them, requirement diagrams enable the analysis and management of requirements; use case diagrams and activity diagrams enable the analysis and establishment of system functions; definition diagrams and connection diagrams achieve the structural description of the model; equation diagrams, state machine diagrams, and activity diagrams achieve the description of continuous and discrete behaviors of the model. As shown in Figure 2, in order to support the hybrid modeling of system-level design models with discrete characteristics and physical-level models with continuous characteristics, continuous classes to support the modeling of systems with continuous behavior, discrete classes to model systems with discrete behavior, and couple classes to realize multi-granularity hybrid modeling are defined on the basis of XDEVS theory. In addition, function classes, connector classes, and record classes are defined to support the modeling of specific functions, data structures, and ports in continuous and discrete classes. The different classes are modeled in both graphical and textual forms. For example, the couple class defines the system architecture and the interaction between the components based on the definition diagram/text and the connection diagram/text. Among other things, the components can consist of discrete classes defined by a combination of definition diagram/text and state machine diagram/text and continuous classes defined by a combination of definition diagram/text and equation diagram/text. Ultimately, the resulting textual model has executable capabilities. In conclusion, the XDEVS theory, the graphical combinatorial modeling approach, and the modeling paradigm of bi-directional conversion of graphical and textual models provide the possibility to implement integrated modeling in X language.

3.2. System design and simulation integration methodology based on X language

Currently, there are two core methodologies in MBSE: Harmony-SE and MagicGrid. The core of Harmony-SE is divided into three parts: requirement analysis, functional analysis, and design synthesis. In terms of implementation details, Harmony-SE relies on activity diagrams, sequence diagrams, and state machine diagrams for the behavioral description of complex system, from the external behavioral identification in the black box to the internal functional logic in the white box, essentially downplaying the importance of parametric diagrams. The MagicGrid implements the design process of complex systems in the form of a matrix. Vertically, it is based on the four core perspectives of requirements, behavior, structure, and parameters as the pillars of system engineering research; horizontally, it is mainly divided into two levels: problem domain and solution domain to analyze the system solution step by step. The core implementation process also starts with requirements analysis, defines the structure and behavior of the black box, and then defines the structure and behavior of the white box and the parameter constraints of the system. In terms of implementation details, MagicGrid relies on activity diagrams and state machine diagrams for behavior descriptions but places more emphasis on modeling parametric diagrams. However, although Rhapsody and Cameo Systems Modeler provides methods for describing complex system objects at a finer granularity (Harmony-SE, MagicGrid), in the end, these SysML-based models are not inherently executable. In essence, these two MBSE methodologies only implement the design process of complex systems and it is difficult

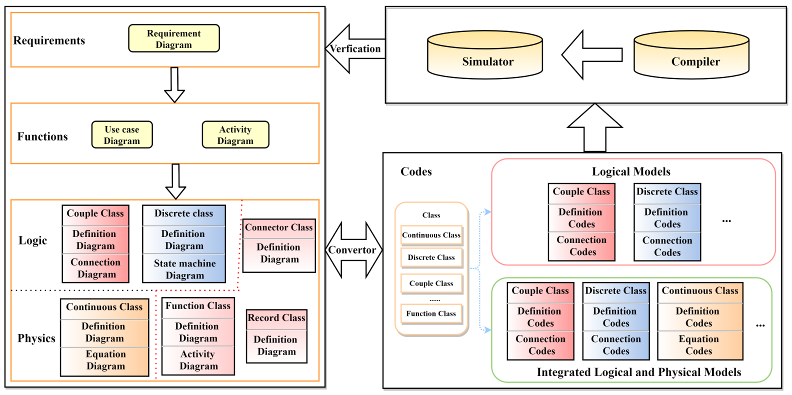


Figure 3. System design and simulation integration methodology based on X language.

to verify the system design in depth. Therefore, a true MBSE methodology must introduce simulation, and only by realizing the unification of complex system design and simulation can the closed-loop system design be truly realized.

Currently, there are two traditional MBSE modeling and simulation approaches to achieve the unification of complex system design and simulation: the FMI-based co-simulation approach and the model conversion-based approach. Although model conversion and co-simulation have bridged the barriers between design and simulation to a certain extent, both methods require cross-language and cross-platform, which poses a huge challenge to the consistency maintenance of models when the R&D system is complex enough.

In order to truly realize the integration of design and simulation. In this paper, an integrated design and simulation methodology based on X language is proposed. The methodology can completely support the design and verification process of MBSE based on X language. As shown in the figure3, combining the four levels of MBSE development (requirements, functional, logical, and physical), the diagram based on X language realizes the modeling process of each level. Specifically, at the requirements level, stakeholder needs are analyzed and managed based on requirements diagrams; at the functional level, the decomposition of functions is realized based on use case diagrams and activity diagrams; at the system logical architecture level, the composition of logical subsystems, interaction logic, and functional behavior of each logical subsystem is realized based on graphical couple classes (definition diagram, connection diagram), discrete classes (definition diagram, state machine diagram), etc. At the physical architecture level, the physical constraints of each physical subsystem are modeled based on graphical continuous classes (definition diagrams, equation diagrams), etc. From the simulation point of view, the graphical logical architecture models and physical models constructed above can be automatically generated into corresponding textual simulatable models by the developed converter. Specifically, for the logical architecture model of the system, it consists mainly of couple classes and discrete classes. The definition text of the couple class defines the components of the logical system, the connection text defines the interaction between all logical subsystems constructed by the discrete class, and the definition text of the discrete class defines the parameters, ports, etc. of each logical subsystem, and the state machine text defines the states and state transfer conditions. The final logic simulable textual model is simulated by the developed compiler and simulator to verify the functional requirements of the system. The mixed logical and physical model consists of coupled, continuous, and discrete classes. The definition text of the couple class defines the components of the hybrid system, the connection text defines the events and data interaction between the logical and physical subsystems, the definition text of the discrete class defines the parameters, event inputs, and output ports of each logical subsystem, the state machine text defines the states and state transfer conditions, the definition text of the continuous class defines the parameters, variables, and ports of each physical subsystem, and the equation diagram defines the physical constraints based on the equations. The resulting mixed logical and physical simulable textual model can be simulated by the developed compiler and simulator to integrate and verify the functional requirements and non-functional requirements related to the physical performance of the system. When the simulation results show that the requirements are not satisfied, the system's functional architecture and key performance parameters can be verified and optimized through continuous iterative design. In summary, the X language-based integrated design and simulation methodology unifies the integrated modeling and simulation of the logical and physical levels of the system. The methodlgy avoids the problem that the physical layer is based on requirements only and cannot be verified jointly with the system logical architecture layer, which leads to differences in the system integration process and cannot guarantee the consistency of the system design.

4. Case Study for electric vehicles(EV)

Currently, MBSE has become the first method for the development and design of various complex systems. Research institutes and developers in different fields at home and abroad are gradually adopting MBSE to try to build complete virtual digital prototypes containing system design and simulation. Therefore, this paper adopts the integrated design and simulation methodology based on X language to realize the integrated modeling and simulation of the linear driving scenarios of EV from requirements and functional analysis, system architecture, and physical units to achieve the purpose of rapid verification and optimization of EV design solutions and key performance parameters.

4.1. Requirements analysis

Electric vehicles are the main trend of future automotive development. In this paper, an electric vehicle linear driving scenario is studied and an integrated design and simulation approach based on X language is used to realize the design and verification process for electric vehicles. The driver is the main stakeholder involved in this scenario. The specific requirements involved in the linear driving scenarios process are summarized as follows by investigating the relevant drivers' driving requirements for electric vehicles:

1. EV can release the handbrake normally.
2. EV can accelerate and decelerate normally.
3. Under WLTC conditions, EV can automatically complete driving tests with an average error of less than 0.5.

The above requirements are the basis for realizing the functional analysis and design of EV. As shown in Figure 4, this paper documents and manages requirements based on the requirements diagram.

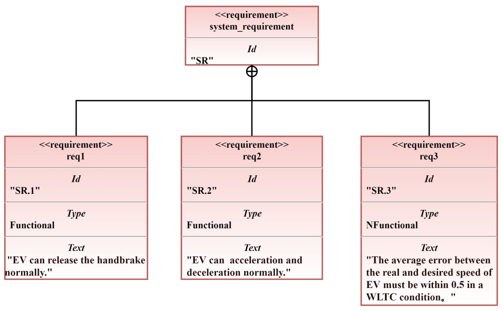

<<requirement>>
system_requirement
Id
"SR"
<<requirement>>
req1
Id
"SR.1"
Type
Functional
Text
"EV can release the handbrake normally."
<<requirement>>
req2
Id
"SR.2"
Type
Functional
Text
"EV can acceleration and deceleration normally."
<<requirement>>
req3
Id
"SR.3"
Type
NFunctional
Text
"The average error between the real and desired speed of EV must be within 0.5 in a WLTC condition。"

Figure 4. Requirements Diagram for EV

4.2. Functional analysis

In order to refine the requirements of the linear driving scenarios, this paper is based on a use case diagram to clarify the functions that should be implemented in an electric vehicle. As shown in Figure 5, the use case diagram depicts the linear driving use case and its four sub-use cases (handbrake brake release, acceleration, deceleration, and operating state testing) in the linear driving scenarios of EV .

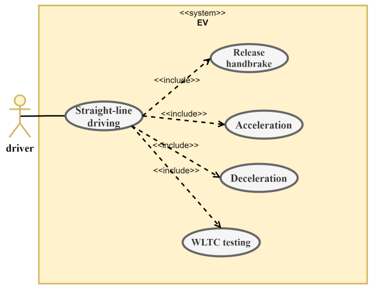


Figure 5. Use case Diagram for EV

After the requirements and use case model of the linear driving scenarios are built, this paper establishes the functional activity flow of the linear driving scenarios based on the activity diagram. As shown in Figure 6, firstly, the driver releases the handbrake and then sends a speed change request according to the road condition, and when it is necessary to accelerate or drive at a constant speed, the throttle opening size is adjusted, and the car body realizes the speed response and feeds back to the central control system; similarly, when it is necessary to decelerate, the brake opening size is adjusted, and the car body realizes the speed response and feeds back to the central control system; so on and so forth, and when the driving task is completed, the speed control is stopped. By assigning the functional activity flow to the corresponding participants in the form of swim lanes, the logical or physical subsystems involved in the linear driving scenario and the interaction between them can be clearly analyzed, based on which the top-level model of the electric vehicle can be constructed.

Based on the above functional architecture analysis, the linear driving scenario will include several subsystems, such as the central control system, throttle system, braking system, etc. In this paper, we establish the composition of each logical or physical subsystem inside the electric vehicle and the interaction between them based on couple classes. The couple class is described at the graphical level by a definition diagram and a connection diagram, as shown in Figures 7a and 7b. Among them, the definition diagram describes the subsystems involved in the electric vehicle during linear driving; the connection diagram describes the interaction of logical signals or physical variables between the subsystems. The subsystems in yellow in the connection diagram have discrete characteristics, and the subsystems in red have continuous characteristics. Meanwhile, the corresponding textual model can be automatically generated after the graphical modeling is completed, as shown in Figure 7c.

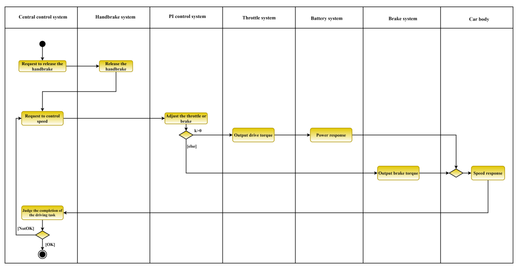
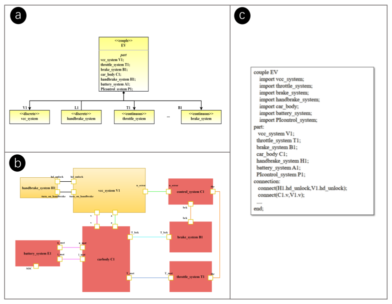


Figure 6. Activity Diagram for EV

4.3. Architecture Design



```
couple EV
   import vcc_system;
   import throttle_system;
   import brake_system;
   import handbrake_system;
   import car_body;
   import battery_system;
   import PIcontrol_system;
part:
  vcc_system V1;
 throttle_system T1;
 brake_system B1;
 car_body C1;
 handbrake_system H1;
 battery_system A1;
 PIcontrol_system P1;
connection:
  connect(H1.hd_unlock,V1.hd_unlock);
  connect(C1.v,V1.v);
 ....
end;
```

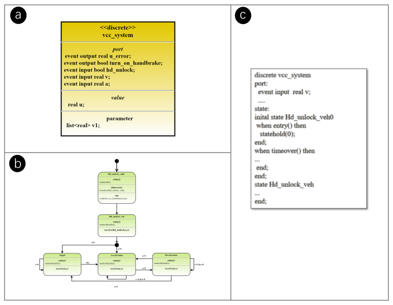
Figure 7. Graphical and textual models for system architecture

4.4. Design of logical and physical subsystems

Based on the above analysis, the logical or physical behavior of each subsystem is established based on continuous and discrete classes. Once all the subsystems are modeled, simulations can be executed to verify that the system design meets the requirements. In this paper, we introduce the typical subsystems of the central control system with discrete characteristics and the dynamics system with continuous characteristics as examples.

4.4.1. *The central control system*

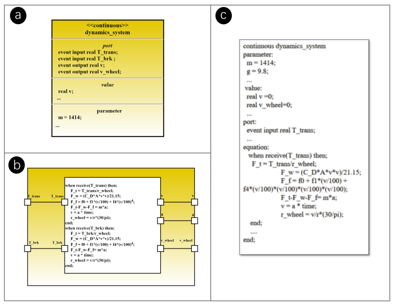
The central control system(CCS) acts as the master control system in the linear driving scenario of the EV and coordinates and controls the behavior of other subsystems in the linear driving scenario. After sending the handbrake release command, the central control system enters the handbrake pending release state. After the handbrake is released, it enters the state of the driving situation to be determined. In this state, the difference signal between the desired speed and real speed is sent to the automatic control system, which gets the opening degree of throttle or brake through PI control and outputs the desired driving or braking torque through the throttle system or brake system, and then calculates the current car speed through the motor system, transmission system, power system, and dynamics system, and returns it to CCS, which judges the current driving status of the EV based on the car speed and acceleration. The central control system judges the current driving state (acceleration, deceleration, and standstill) of the EV based on the speed and acceleration of the EV, and so on until the task of linear driving is completed. Here, the structure and behavior of CCS are described based on discrete classes. As shown in Figures 8a and 8b, where the definition diagram describes the input and output signals of CCS and the related state variables; the state machine diagram describes the state and behavior logic of CCS. Meanwhile, the corresponding textual model can be automatically generated after the graphical modeling is completed, as shown in Figure 8c.



```
discrete vcc_system
port:
  event input  real v;
  ....
state:
inital state Hd_unlock_veh0
 when entry() then
   statehold(0);
end;
when timeover() then
...
 end;
end;
state Hd_unlock_veh
...
end;
```

Figure 8. Graphical and textual models for CCS

4.4.2. *The dynamics system*



```
continuous dynamics_system
parameter:
  m = 1414;
  g = 9.8;
  ...
 value:
  real v =0;
  real v_wheel=0;
  ...
port:
  event input real T_trans;
  ...
equation:
   when receive(T_trans) then;
     F_t = T_trans/r_wheel;
                    F_w = (C_D*A*v*v)/21.15;
                    F_f = f0 + f1*(v/100) +
f4*(v/100)*(v/100)*(v/100)*(v/100);
                    F_t-F_w-F_f= m*a;
                    v = a * time;
                    r_wheel = v/r*(30/pi);
    end;
    ....
end;
```

Figure 9. Graphical and textual models for the dynamics system

In the linear driving scenarios, the dynamics system calculates the current acceleration and speed by calculating the combined force on the body at each moment and sends it to CCS to determine the current driving status. The dynamics system, upon receiving the actual driving torque or braking torque calculated by the transmission system, can calculate the driving force or braking force F_t using equation (1).

$$F_t = \frac{T_0}{r} \tag{1}$$

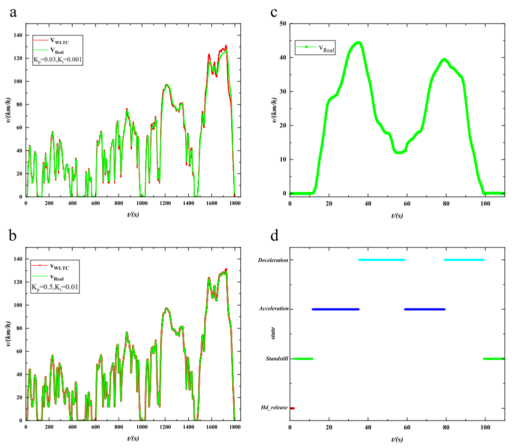


Figure 10. Simulation results for WLTC.

where T_0 represents the actual driving or braking torque and r represents the wheel radius. The resistances encountered during linear driving scenarios include rolling resistance and air resistance. Their calculation formulas are as follows:

- Rolling resistance:

$$f = f_0 + f_1(\frac{u}{100}) + f_4(\frac{u}{100})^4 \tag{2}$$

$$F_f = f * mgcos(\alpha) \tag{3}$$

where f is the rolling friction coefficient, u represents the vehicle speed, and f_0, f_1, and f_4 are constants.

- Air resistance:

$$F_w = \frac{C_D A u^2}{21.15} \tag{4}$$

where C_D represents the air drag coefficient, A represents the windward area, and u represents the vehicle speed.

Based on the principle of dynamics, the acceleration, speed, and wheel speed of EV can be calculated with the following formulas:

$$F_t - F_w - F_f = ma \tag{5}$$

$$u = u_0 + at \tag{6}$$

$$u_w = (30/pi)(\frac{u}{r}) \tag{7}$$

where a represents the acceleration, u represents the vehicle speed, u_0 represents the initial vehicle speed, and u_w represents the wheel speed. Based on the above analysis, the construction of the structure and behavior of the dynamics system is realized based on continuous classes. As shown in Figures 9a and 9b, the definition diagram defines the input and output ports, related parameters, and variables and the equation diagram define the physical mechanism . Meanwhile, the corresponding textual model can be automatically generated after the graphical modeling is completed, as shown in Figure 9c.

4.5. Simulation

The textual model of the electric vehicle formed after all subsystems are built can be combined to form a complete executable simulation model. Based on the simulation results obtained, it is possible to verify whether the currently designed EV meets all the system design requirements. At the beginning of the simulation, the values of the relevant parameters of the EV are shown in Table 1.

Table 1. Key parameters of EV

parameters	value
Transmission ratio	13.5
Efficiency of transmission	0.95
Wheel radius	0.284
K_p	0.03
K_i	0.001

Finally, the results of the simulation are used to verify that the requirements are met. As can be seen from Figure 10a, when the parameters Kp=0.03 and Ki=0.001 of the PI controller, the difference between the EV speed and the desired speed is relatively large. In addition, the average error between the car speed and the desired speed was calculated to be 3.014. The req3 was not satisfied. Therefore, the relevant parameters need to be adjusted and re-simulated to verify that the requirements are met. Here, we ensure that the other parameters remain unchanged and adjust the values of Kp and Ki to meet the relevant requirements. After several rounds of parameter adjustment, Kp=0.5 and Ki=0.01 were finally selected, and as can be seen from Figure 10b, the difference between the car speed and the desired speed is relatively small. In addition, the average error between the speed of the EV and the desired speed was calculated to be 0.252. The req3 was satisfied. in addition, under this set of parameters, combined with Figures 16c and 16d, it can be seen that the EV completed the handbrake release, acceleration, and deceleration normally in the first 110s, satisfying the req1 and req2. In summary, the integrated design and simulation methodology based on X language can realize the full closed-loop verification process of requirements and functional analysis, system architecture design, and simulation for linear driving scenarios of EV, effectively improving the efficiency of system development.

5. Conclusions

This paper proposes an integrated design and simulation methodology based on the X language, which provides a framework to unify requirements, functions, logic and physics. Based on this methodology, combining X language and XLab, the full closed-loop verification process from requirement and functional analysis, system architecture design to simulation is realized for the linear driving scenarios of EV, which can quickly realize the early verification of the design solution of EV and improve the efficiency of system design and development.

6. Funding

This work is supported by the National Key R&D Program of China (GrantNo.2018YFB1701601).

References

Douglass, B. (2005). The harmony process. i-logix white paper, i-logix. *Inc.: Burlington, MA, USA*.

Feldman, Y. A., Greenberg, L., and Palachi, E. (2014). Simulating rhapsody sysml blocks in hybrid models with fmi. In *Proceedings of the 10 th International Modelica Conference; March 10-12; 2014; Lund; Sweden*, number 096, pages 43–52. Linköping University Electronic Press.

Friedenthal, S., Moore, A., and Steiner, R. (2008). Omg systems modeling language (omg sysml™) tutorial. In *INCOSE international symposium*, volume 18, pages 1731–1862. Wiley Online Library.

integration working group, S.-M. et al. Sysml modelica transformation specification version 1.0, 2010.

Jieshi, S., Qing, Z., Bingfei, L., and Cong, C. Co-simulation of sysml and simulink/modelica using fmi.

Lin, Z., Lei, R., and Fei, T. (2010). Complex product manufacturing digital integration platform technology. *Defense Manufacturing Technology*, 4:4–10.

Morkevicius, A., Aleksandraviciene, A., Mazeika, D., Bisikirskiene, L., and Strolia, Z. (2017). Mbse grid: A simplified sysml-based approach for modeling complex systems. In *INCOSE International Symposium*, volume 27, pages 136–150. Wiley Online Library.

Pengfei, G., Lin, Z., Zhen, C., and Junjie, Y. (2022). Collaborative design and simulation integrated method of civil aircraft take-off scenarios based on x language. *Journal of System Simulation*, 34(5):929.

Ramos, A. L., Ferreira, J. V., and Barceló, J. (2011). Model-based systems engineering: An emerging approach for modern systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(1):101–111.

Schamai, W. (2009). *Modelica modeling language (ModelicaML): A UML profile for Modelica*. Linköping University Electronic Press.

Schamai, W., Pohlmann, U., Fritzson, P., Paredis, C. J., Helle, P., and Strobel, C. (2010). Execution of umlstate machines using modelica. In *3rd International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools; Oslo; Norway; October 3*, number 047, pages 1–10. Citeseer.

Shuhua, Z., Yue, C., Zheng, Z., and Yusheng, L. (2014). System design and simulation integration for complex mechatronic products based on sysml and modelica. *Journal ofComputer-Aided Design & Computer Graphics. Beijing*, 30:728–738.

Xie, K., Zhang, L., Laili, Y., and Wang, X. (2022). Xdevs: A hybrid system modeling framework. *International Journal of Modeling, Simulation, and Scientific Computing*, 13(02):2243001.

Xinquan, W., Xuefeng, Y., Xingchan, L., and Yongzhen, W. (2023). Simulating hybrid sysml models: a model transformation approach under the devs framework. *The Journal of Supercomputing*, 79(2):2010–2030.

Zhang, L., Ye, F., Laili, Y., Xie, K., Gu, P., Wang, X., Zhao, C., Zhang, X., and Chen, M. (2021). X language: an integrated intelligent modeling and simulation language for complex products. In *2021 Annual Modeling and Simulation Conference (ANNSIM)*, pages 1–11. IEEE.

Zhang, L., Ye, F., Xie, K., Gu, P., Wang, X., Laili, Y., Zhao, C., Zhang, X., Chen, M., Lin, T., et al. (2022a). An integrated intelligent modeling and simulation language for model-based systems engineering. *Journal of Industrial Information Integration*, 28:100347.

Zhang, Y., Gu, P., Chen, Z., and Zhang, L. (2022b). A method and implementation of automatic requirement tracking and verification for complex products based on x language. In *China Intelligent Networked Things Conference*, pages 443–455. Springer.