Journal of Highway and Transportation Research and Development

Home page: www.sciopen.com/journal/2095-6215

Multi-entity co-simulation of intelligent vehicle based on distributed message-oriented middleware

Liping Peng*, Rongrong Deng**, Xiaoyan Li, Aiyan Yang, Lei Li, Yanpeng Wei

Intelligent Transportation Research Center, Research Institute of Highway Ministry of Transport, Beijing 100088, China

Abstract: With the integration of cutting-edge technologies such as information communication, the internet, big data, cloud computing, and artificial intelligence into intelligent vehicles, the scale and complexity of their cyber-physical subsystems have been steadily increasing. In response to the growing demands for high concurrent access and efficient operation within the multiple entities of information system modeling and simulation, this study introduces a novel method for multi-entity co-simulation of intelligent vehicles, grounded in distributed message-oriented middleware. Tailored for the unique challenges of multi-entity co-simulation scenarios, the proposed method involves classifying heterogeneous simulation platforms and developing distinct implementation modes for message middleware interface units corresponding to each platform type. This approach significantly enhances the efficiency of multi-entity co-simulation in intelligent vehicle systems, facilitating more effective integration and operation of diverse subsystems.

Keywords: Automotive engineering; Intelligent vehicle; Cyber-physical system; Heterogeneous simulation platforms; Multi-entity co-simulation; Distributed message-oriented middleware

1 Introduction

The cyber-physical system of an intelligent vehicle integrates the physical, information, and application layers, encompassing the interplay among people, vehicles, roads, and clouds as a unified entity [1]. Leveraging the latest advancements in information and communication technology, it performs vehicle-road fusion perception, decision-making, and control to enhance the safety and efficiency of vehicle travel and traffic operations [2]. The system comprises an intelligent vehicle, other traffic participants, roadside infrastructure, a cloud control basic platform, a cloud control application platform, and relevant support platforms to ensure seamless operation. A communication network connects all components of the system, facilitating data flow and integration [3].

2 Experimental

2.1 The requirement of multi-entity co-simulation

To ensure the speed and accuracy of simulations for the cyber-physical system of intelligent vehicles, different sectors within the automotive industry—such as vehicle manufacturing, cooperative infrastructure, and communication technology—often utilize their own specialized tools and platforms for high-precision modeling. However, the integration of these high-precision models from diverse fields into a single platform for system-level simulation presents challenges. This integration issue complicates the analysis of how individual component characteristics affect the entire system, potentially resulting in extensive design revisions.

To overcome these challenges and enable thorough debugging, system integrators must coordinate with multiple suppliers across various simulation environments. Large-scale integrated simulations are divided into domain-specific models, with the large system broken down into several subsystems. Each subsystem is then modeled and simulated using the most appropriate high-precision tools and platforms available for the respective fields, disciplines, and entities.

As depicted in Fig. 1, large-scale heterogeneous systems often implement parallel processing solutions during operation. The simulation platform incorporates software like Adams and Carsim for automotive simulations, CANoe for bus system emulation and vehicle-to-vehicle communication, and tools such as SystemVue, Veins, and CANoe.Car2X for urban public transport simulations. Vissim is used for microscopic modeling, and the platform is also capable of integrating independent model modules developed in different programming languages.

The traditional co-simulation mode relies on the cooperation between the simulation software manufacturers, so it comes to a workload of secondary development, which will lead to high coupling problems that will be bad for segmentation and parallel of large system. Meanwhile, it lacks of supporting for parallel computing when the system is performing data interaction at discrete communication points, the performance of co-simulation can not be improved in efficiency.

To solve the high coupling problem of the simulation platform, it is necessary to establish a design of independent Interface subsys-

Received: 15 May 2024; **Revised:** 30 July 2024; **Accepted:** 18 August 2024
* *E-mail address*: plp@itsc.cn; ** *Corresponding author*: drr@itsc.cn
DOI: 10.26599/HTRD.2025.9480043

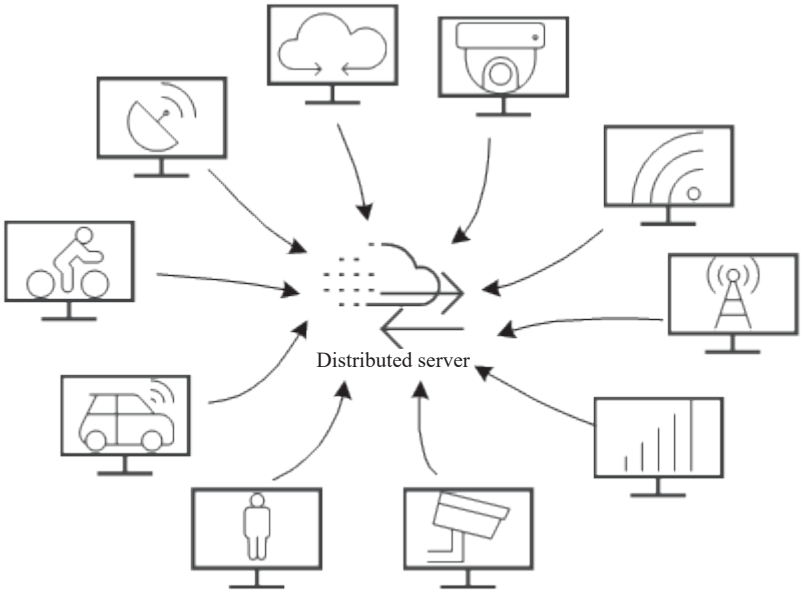

Fig. 1 Multi-entity co-simulation of intelligent vehicle.

tem, which is not depend on simulation platform, the subsystem has the characteristics of multi-domain, multi-disciplinary and multi-entity co-simulation model. It also need to break away from the access restriction of simulation platform, the simulation software is distributed on different computing platforms through the distributed solution, the data parallel exchanging and computing runs at the main controlling node. Thus it can be seen, distributed technology can support the improvement of simulation efficiency of the cyber-physical system of intelligent vehicle.

2.2 Application of distributed co-simulation technology

The existing distributed co-simulation technology mainly has two types. The first one is HLA-based co-simulation, which is a general specification for distributed simulation system. To provide integrated and standardized interfaces for the co-simulation software, the co-simulation infrastructure RTI, which is defined by the HLA interface specification, it specifies the corresponding application interface API. As a sub-node of co-simulation software, RTI is separating the simulation application from the underlying support environment [4], That is, the specific implementation of simulation function, simulation operation management and low layer communication are separated, and their implementation details are hidden [5]. However, HLA can solve the problem of model data interaction caused by the inconsistency of subsystem interfaces in multi-domain co-simulation, and it is so complex to build a multi-domain co-simulation system by HLA, that requires high cost and rich experience [6]. The other one is the co-simulation based on FMI standard, which was originally proposed by DaimlerAG for the purpose of supporting the test of collaborative simulation environment on model, software and hardware in automotive field [7]. At present, many simulation software have added to support FMI standards, including common simulation software such as Matlab, AMESim, Dymola, SimulationX, Carsim, etc., the universality of simulation models have been better expanded, the scope of co-simulation has been expanded [8].

2.3 Application of traditional distributed co-simulation

Traditional distributed co-simulation (such as FMI distributed co-simulation technology), the communication completes data transmission and calculation by TCP/IP, Deploy the communication module independent of the simulation platform at the communication nodes of each simulation platform, to complete the acceptance and forwarding of model data, to support synchronization for distributed integrated simulation and clock of heterogeneous simulation platforms, thus to improve the efficiency of co-simulation. This method is generally used for model mechanism merging in various heterogeneous simulation platforms of different disciplines in a single entity. All kinds of simulation platforms need to negotiate with the communication step, to meet the clock synchronization and complete a single entity simulation step.

As shown in Fig. 2, there is an example, with traditional TCP/IP, the dynamics and kinematics of intelligent vehicle realize distributed co-simulation. Assuming that the simulation step is t_S, the simulation times is n, the time needed for distributed co-simulation is:

$$T^{\text{old}} = n \times t_S. \tag{1}$$

For the mechanism merging relationship between vehicle dynamics and kinematics model, these two simulation need to be carried out simultaneously. Assuming that one calculation of dynamics model takes t_E, one calculation of kinematics model takes t_D, the communication takes t_C, these two simulation steps must be consistent, ideally:

$$t_S = \max(t_E, t_D) + t_C. \tag{2}$$

The time for distributed co-simulation has the following constraints:

$$T^{\text{old}} = n \times [\max(t_E, t_D) + t_C]. \tag{3}$$

Suppose there are $m$discipline models participating in traditional distributed co-simulation, and the time for each discipline model has the following constraints:

$$T_i^{\text{old}} = n \times (t_{\max} + t_C), \quad i \in [0, m). \tag{4}$$

2.4 Application of distributed message middleware in co-simulation

The simulation of the cyber-physical system of intelligent vehicle refers to multi-entity co-simulation, a single entity can enter the next simulation calculation step after it completes its own simulation step, it no need to wait for the feedback from the cloud control system. For example, the intelligent driving roadside unit sends traffic sign information to OBU through radio waves. RSU and OBU realize modeling and simulation on different simulation platforms. RSU completes vehicle-road communication through broadcast radio waves. There is no mechanism merging of physical architecture between RSU and OBU. RSU can enter the next

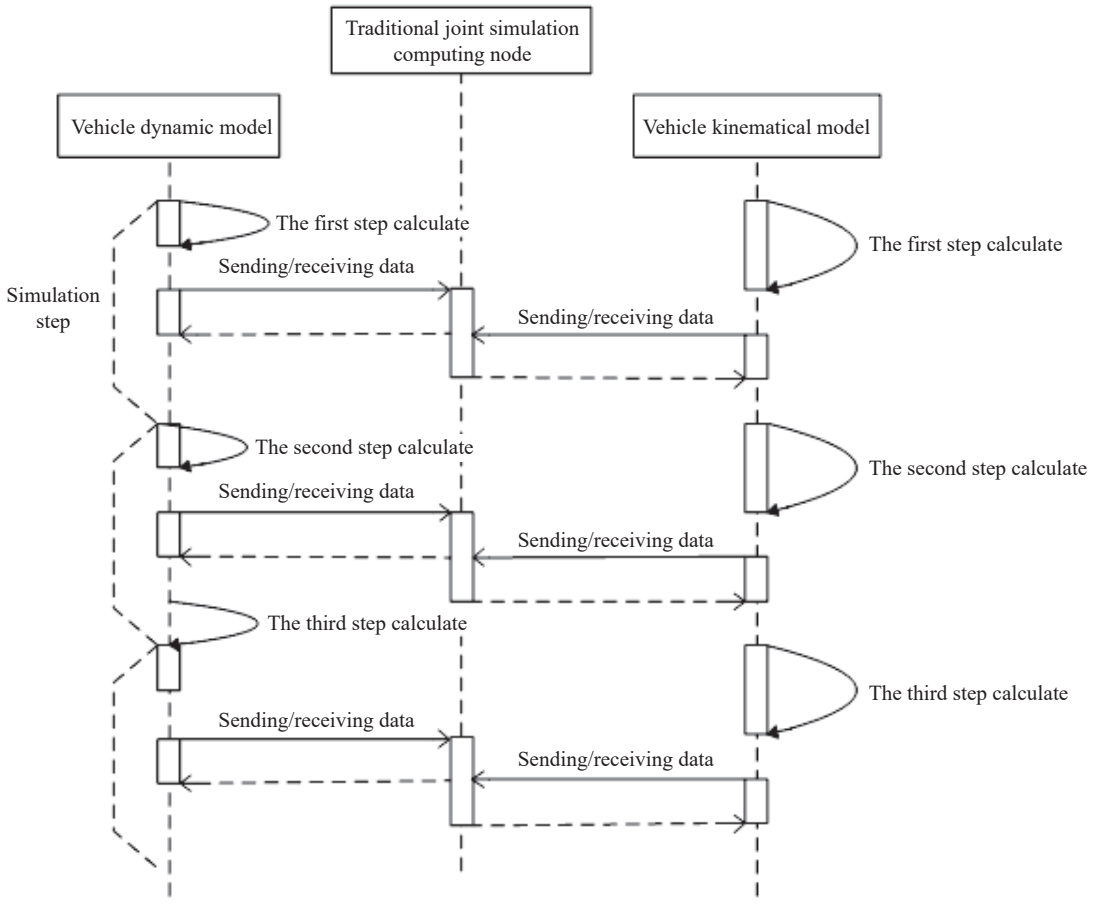

Fig. 2 Sequence diagram of traditional distributed co-simulation of two disciplines.

broadcast transmission without receiving feedback from OBU. On the contrary, OBU will not wait for the feedback when sending data to the sensing device. Therefore, the multi-entity co-simulation of the cyber-physical system of intelligent vehicle does not need to negotiate the communication step on multiple simulation platforms, the data will be sent in a stacked manner. The clock synchronization execution mode of traditional distributed co-simulation will affect the efficiency of multi-entity co-simulation.

Distributed message middleware is the basic software to support sending and receiving messages in distributed systems, it is used to solve the problem of message transmission among distributed systems and to realize the cooperation of multiple systems [9]. Distributed message middleware provides data exchange of unrelated platform, by providing message delivery and message queuing model, it extends the communication among all processes under the distributed architecture, it lowers the coupling degree of multiple systems [10].

As shown in Fig. 3, there is an example of two vehicle models participating in co-simulation in an cyber-physical system of intelligent vehicle, it uses message-based middleware technology to realize distributed co-simulation. Suppose that the simulation step of vehicle A is t_{SA}, the simulation step of vehicle B is t_{SB}, the simulation times is n, the time needed for the distributed co-simulation of vehicle A is:

$$T_{A}^{new} = n \times t_{SA}. \tag{5}$$

The time needed for the distributed co-simulation of vehicle B is:

$$T_{B}^{new} = n \times t_{SB}. \tag{6}$$

Suppose there are m entity models participating in the co-simulation based on message middleware. Ideally, the running time of each entity model has the following constraints on :

$$T_{i}^{new} = n \times (t_i + t_C), \quad i \in [0, m). \tag{7}$$

After superposition of each node, the ratio of the time of distributed co-simulation based on message middleware to the time of traditional distributed co-simulation is as follows:

$$R_{ts} = \left(1 - \frac{\sum_{0}^{m-1} T_{i}^{new}}{\sum_{0}^{m-1} T_{i}^{old}}\right) = \frac{t_{max} - t_{avg}}{t_{max} + t_C}. \tag{8}$$

2.5 Implementation of distributed message middleware in co-simulation

According to support or not support the FMI standard, the high-precision simulation platform of cyber-physical system of intelligent vehicle can be divided into three categories:

(1) Simulation software with FMU module that support the FMI standard: Dymola, SimulationX, OpenModelica, MATLAB/simulink, AMESim, MWorks, Adams, Carsim, CANoe.

(2) Simulation software with FMU module that does not support the FMI standard: SystemVue, Veins.

(3) Other independent model modules written by various codes that do not support the FMI standard.

Based on message-oriented middleware, through message middleware, the model interface unit needs to send the input and output data of the three type models in Fig. 4 to the distributed co-simulation tool server, it relies on the simulation software of their respective platforms for parallel solution when the model simulation runs, so as to realize the efficient integration for cross-domain, cross-platform, cross-discipline and cross-scale model “white box”.

To each kind of heterogeneous platform client simulation software, different interface implementation methods need to be used to achieve the purpose of message middleware communication. The message middleware interface unit must be universal and min-

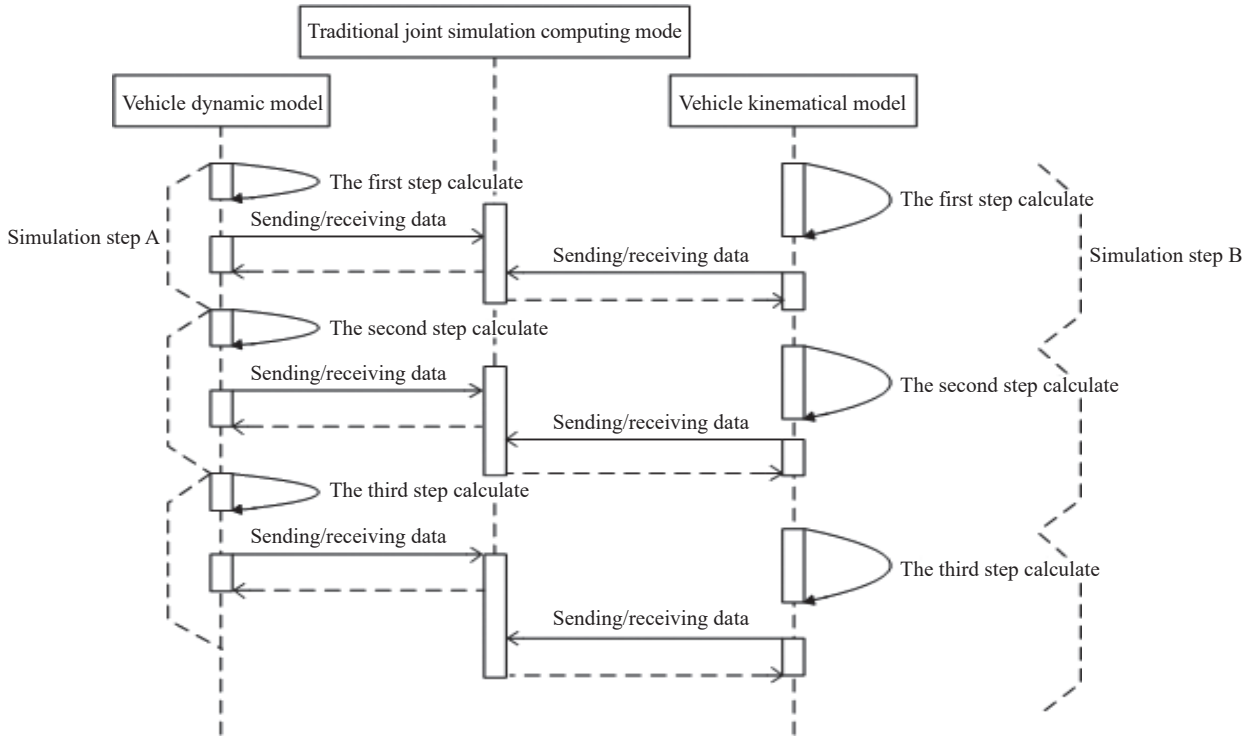

Fig. 3 Sequence diagram of distributed co-simulation based on message middleware between two vehicle entity models.

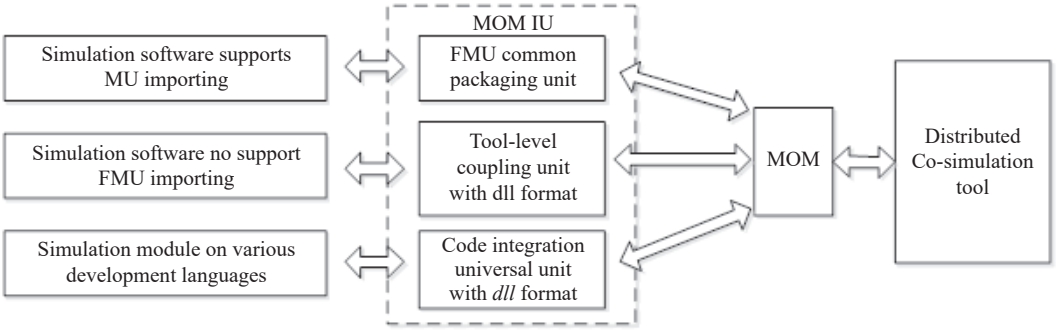

Fig. 4 Distributed co-simulation of three types of heterogeneous platform models based on message middleware.

imize the interface adaptation work on different simulation platforms with the same type. The standardized plug-and-play effect can be achieved through simple configuration on heterogeneous simulation platforms.

The first one is simulation software with FMU module that support the FMI standard, which need to implement the FMU message middleware interface unit with high compatibility. In the process of importing the simulation software, it does not include any simulation model and not participate in any calculation, only as the communication bridge between the simulation software and the message middleware. Most modeling languages can support the message middleware sdk files of the calling and development languages (such as C++, Java, etc.), to realize the function of the message middleware interface unit, they use the simulation software that supporting the FMI standard. The modeling languages mainly include model connection, TCP/IP data communication (data transmission and data reception), parameter dynamic configuration and model white-box integration. The functional unit can be packaged into FMU format and imported into the other application for simulation software that support the FMI standard, so as to realize rapid data interaction between simulation software and message middleware. The manufacturing process of the message middleware interface unit in FMU format is shown as Fig. 5.

The second one is simulation software with FMU module that does not support the FMI standard, considering that, some tools (such as SystemVue and Veins) can not support the FMI protocol very well, the message middleware interface unit with *. dll format need to be designed, the implementation principle is still based on the idea of the FMI agreement. Through calling the message middleware sdk file, it can realize fast data interaction between simulation software and message middleware, the manufacturing process of tool-level coupling message middleware interface unit is shown as Fig. 6.

The thrid one is independent model modules written by various codes. In order to connect different kinds of program codes (or functions) into the technical architecture, and consider different source code as much as possible, the heterogeneous code integration method under the integration architecture is need to be design, that is, to build a code integration middleware in *. dll format. On one hand, the *. dll middleware is a message middleware interface unit, which can realize rapid data interaction between simulation software and message middleware by calling the message- middleware sdk file. On the other hand, the middleware *. dll provides a fixed-format callable interface, which ensures that all programming languages (C/C++, Python, Java, C #, R, Fortran, etc.) can support the dll files and conduct integrated simulation. The imple-

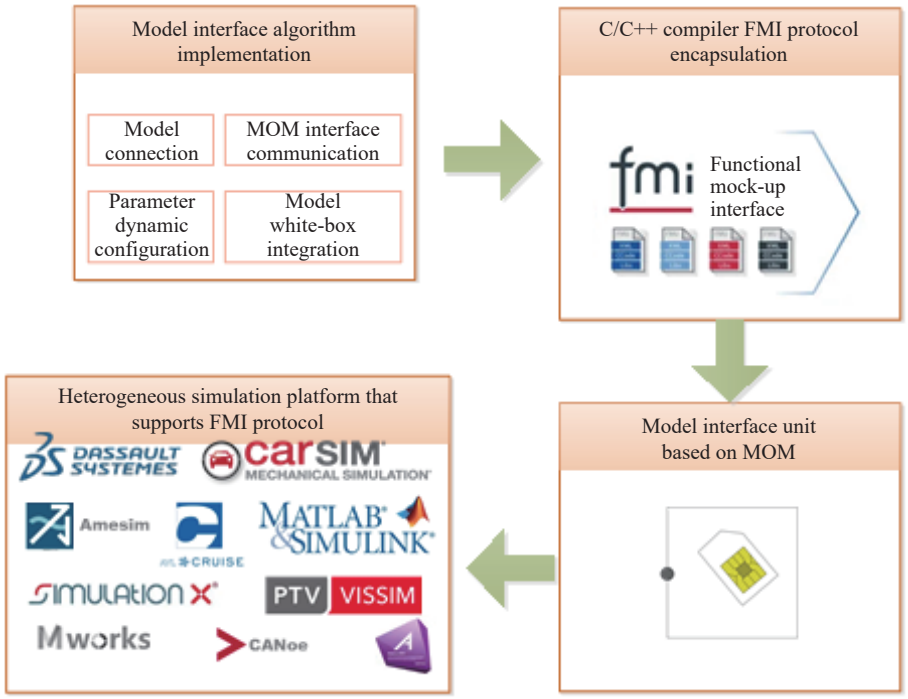

Fig. 5 Manufacturing process of message middleware interface unit in FMU format.

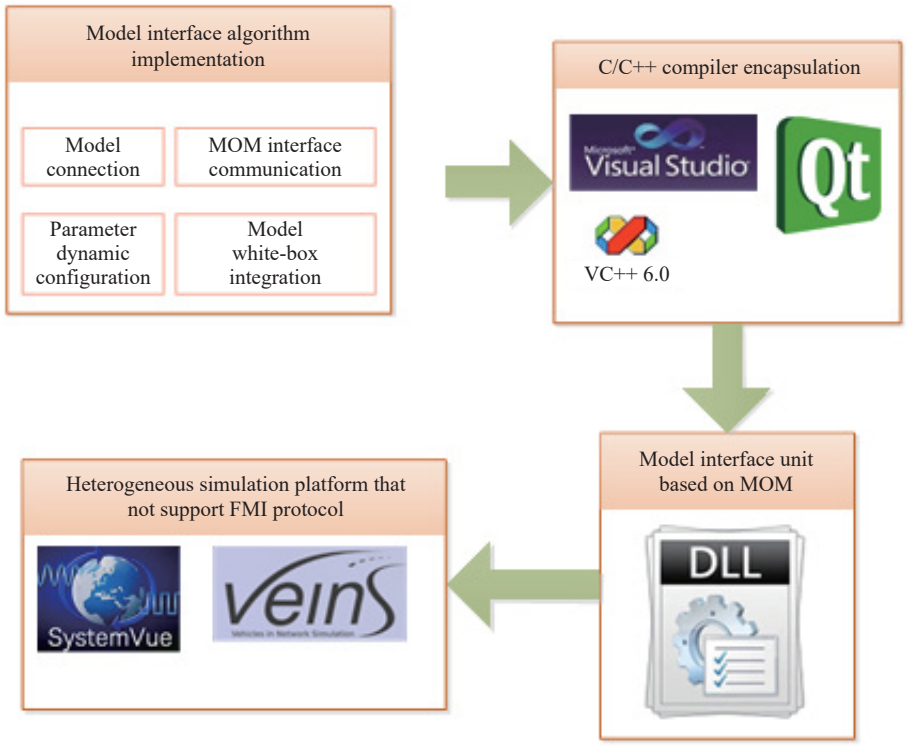

Fig. 6 Manufacturing process of tool-level coupling message middleware interface unit.

mentation principle of code integration is shown as Fig. 7.

3 Conclusion

The application of message middleware in the multi-entity co-simulation of intelligent vehicles aims to establish a transfer hub that facilitates model synchronization, high-efficient interaction, and data distribution. This approach enables the integration and simulation of large-scale complex systems. Research findings indicate that the use of message middleware has significantly enhanced the operational efficiency of multi-entity models.

By leveraging distributed parallel computing technology, a method for achieving rapid data interaction between simulation software and message middleware has been designed. This has allowed for the support of distributed integrated simulation across heterogeneous simulation platforms, thereby improving the efficiency of multi-entity co-simulation for intelligent vehicles.

This work provides a solid theoretical foundation for the modeling and simulation of the cyber-physical system of intelligent vehicles and promotes the advancement of related technologies in

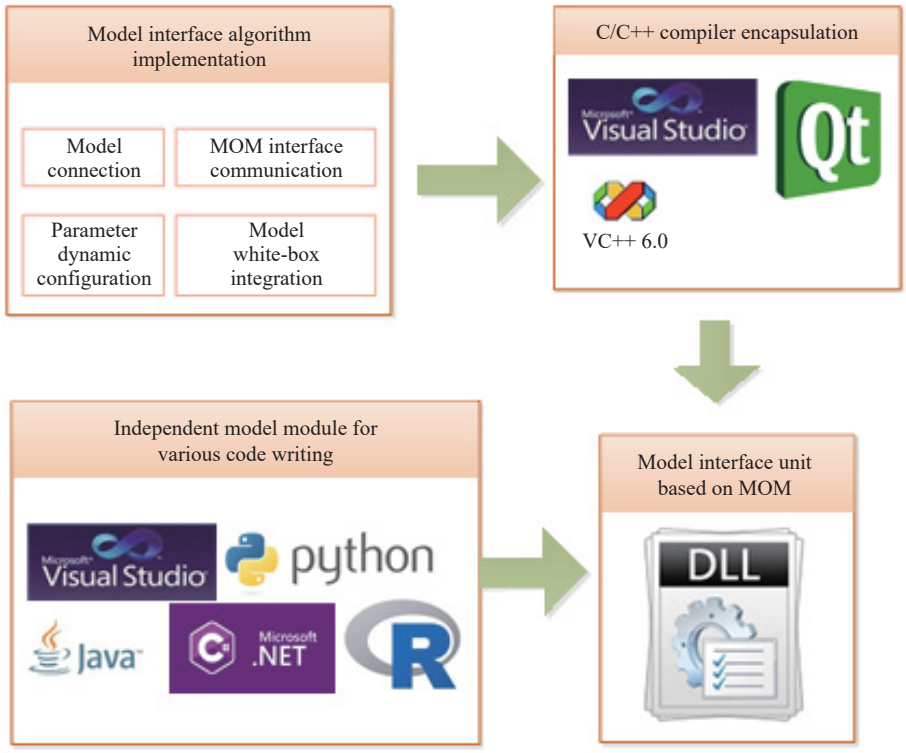

Fig. 7 Production process of code integration message middleware interface unit.

the field of intelligent vehicle simulation.

Declaration of competing interest

The authors have no competing interests to declare that are relevant to the content of this article.

Data Availability Statement

Some or all Data, models, or code that support the findings of available from the corresponding author upon reasonable request.

Acknowledgements

This study was supported by the National Key R&D Program of China (2021YFB2501000), which is greatly appreciated.

References

[1] Lee EA. Cyber physical systems: design challenges. 2008 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC). Orlando: IEEE, 2008: 363−369.

[2] Li KQ, Li JW, Chang XY, et al. Principles and typical applications of cloud control system for intelligent and connected vehicles Journal of Automotive Safety and Energy, 2020, 11(3): 261−275.

[3] Gomes C, Thule C, Broman D, et al. Co-simulation: state of the art. arXiv preprint arXiv: 1702.00686, 2017. https://doi.org/10.48550/arxiv.1702.00686.

[4] Lu JZ. Research on co-simulation technology of heterogeneous simulation system and application in aviation field. Huazhong University of Science and Technology, 2013, 2008(04).

[5] Jin C, Wang G, Gao P. Review of supply chain simulation based on HLA. Journal of System Simulation, 2008, 20(4): 817−822.

[6] Ji XF, Jin Y, Yuan G. Research on the framework of weapon combat simulation system based on HLA. Journal of Changchun University of Technology (Natural Science Edition), 2008, 31(1): 63−67.

[7] Blochwitz T, Otter M, Arnold M, et al. The functional mockup interface for tool independent exchange of simulation models. Linköping Electronic Conference Proceedings. Linköping University Electronic Press, 2011: 105−114.

[8] Junghanns A, Gomes C, Schulze C, et al. The functional mock-up interface 3.0 - new features enabling new applications. Proceedings of 14th Modelica Conference 2021. Sweden: Modelica Association, 2021. https://api.semanticscholar.org/CorpusID:238224838202l.

[9] Kubler S, Schier M, Främling K. A distributed middleware solution for integrating IoT devices with cloud computing. 2017 IEEE International Conference on Cloud Computing Technology and Science (CloudCom). Hong Kong: IEEE, 2017. doi: 10.1109/CloudCom.2017.58.

[10] Wu C, Wang X, Xiao HL. Overview of distributed messaging system. computer science, 2019, 46(S1): 1−5, 34.