Multimedia Tools and Applications (2024) 83:13463–13482
https://doi.org/10.1007/s11042-023-15950-5

Research on visual simulation for complex weapon equipment interoperability based on MBSE

Haigen Yang[1] · Zhun Xia[1] · Yanqing Chen[1] · Linqun Zhu[1] · Luohao Dai[1] · Ruotian Xu[1] · GuiYing Sun[2] · Hongyang Yu[3] · Wenting Xu[3]

Received: 16 December 2021 / Revised: 14 March 2023 / Accepted: 29 May 2023 /
Published online: 6 July 2023

Abstract
As military reforms continue to develop, the battlefield environment is becoming increasingly complex, and traditional single-service combat methods have evolved into integrated joint and collaborative information operations that break down service boundaries on land, sea, and air. The level of weapon system confrontation has also evolved into a system-to-system confrontation. Traditional document-based system architecture design methods can no longer address the complexity and emergent challenges of weapon system construction. In this paper, based on model-driven system engineering, an open, integrated, model-driven weapon equipment interaction system that supports human interaction was constructed using the SysML modeling language and Magicdraw modeling tool. The Unreal Engine 4 landscape building function was used to construct a virtual battlefield environment, and a communication server was developed using C# language to perform visual simulation of interoperability between weapon systems. Based on model-driven weapon equipment interoperability, visual simulation is used to ensure that the function of the weapon equipment system meets the requirements of combat and the combat effectiveness of the system is maximized.

Keywords MBSE · Interoperability simulation · UE4 · SysML · MagicDraw

1 Introduction

With the continuous development of military reforms and the increasingly complex battlefield environment, the traditional warfare pattern has evolved into integrated joint information warfare that breaks the boundaries of various services and arms [25]. Future war is not

✉ Haigen Yang
yhg@njupt.edu.cn

[1] Engineering Research Center of Wider and Wireless Communication Technology of Ministry of Education, Nanjing University of Posts and Telecommunications, Nanjing 21003, People's Republic of China

[2] Chinese People's Liberation Army No. 61416, Beijing 100089, China

[3] Beijing Electro-Mechanical Engineering Institute, Beijing 100074, China

only a confrontation between weapons equipment, which will involve the level of confrontation between weapon systems. Therefore, the systematization, informatization and intelligent construction of weapons equipment is particularly important. The weapon system architecture is a complex system architecture, and system complexity is a difficult challenge to overcome in the construction of weapon equipment systems. System complexity is composed of functional complexity, subsystem complexity, and interoperability complexity [2, 15, 24]. The system of systems is mainly to study the interdependence and collaboration between the hierarchical structure of the system. For the traditional simple weapon equipment system, the relationships between subsystems are not closely connected, and it is easy to produce unpredictable functional coupling and functional conflicts, which is called the emergence of the system. The traditional document-based architecture design method has been unable to solve modern complex weapons equipment. The close association between the subsystems of the complex weapon equipment greatly reduces the design risk of the emergence of the system. However, due to a series of problems such as the very large scale of the weapon equipment system, the complex structure and relationship, high-technology, and high price, etc. The integration between the weapon equipment system and subsystems becomes very difficult, which cannot be guaranteed the interoperability of the system. Therefore, it is necessary to fully demonstrate the system before the architecture design to improve the system performance and maximize the combat effectiveness. In modern warfare pattern, the traditional document-based architecture design method of system of systems is not enough to solve the severe challenges brought by system complexity [3]. In addition, the traditional weapon equipment architecture design based on single platform cannot meet the needs of modern combat environment. It is necessary to establish an integrated platform for demonstration.

As an important factor in warfare, the weapon equipment system is becoming more and more complex, which shows the character of highly complex functions and interaction modes, relatively independent and closely connected sub-systems, functional coupling between systems and unpredictable. Therefore, the construction of complex weapon equipment is a complex system engineering including requirements acquisition and analysis, system structure design, implementation, integration, verification, evaluation, and optimization. In the development early stage of complex weapon equipment, architecture design is an effective means to manage its complexity. In recent years, MBSE has been applied in the field of complex weapon system architecture design. As early as around 2007, European and American began to apply and promote the MBSE method on a large scale in specific industries such as aerospace, military industry, and automobiles. NASA applied the MBSE method to MISSE-X and other projects to solve the problems of confusion in demand management and difficulty in designing complex weapon systems that appeared in the project. The US JPL (Jet Propulsion Laboratory) cooperated with Caltech and successfully developed the basic structure of MBSE, including the basic elements of ontology, reusable modeling methods, integrated comprehensive modeling methods and developed the matching tools are used to generate documents. In China, the China Aerospace Science and Industry Corporation and China Aerospace Science and Technology Corporation are also stepping up efforts to promote the application of MBSE in aerospace field, and have established relevant R&D departments for various research.

In view of the increasingly complex characteristics of equipment architecture design, this paper constructs an open, integrated, model-driven and “human-in-loop” visual simulation and verification system for weapon equipment interoperability. The system uses SysML [5, 9, 19, 22, 23] as the modeling language with MagicDraw [14, 16, 18] to construction the MBSE [6, 13, 17] model and build a virtual reality battlefield

environment with UE4, which developed a plugin of MagicDraw to realize the bidirectional data flow between SysML and UE4 [26]. Through the "human-in-loop" decision confrontation according to the actual combat process of "Observation, Orientation, Decision, Action (OODA)" [4, 8, 11], the model interoperability technology is applied to integrate the operation panel to achieve the interoperability simulation and reproduce the whole process of the model-driven "human-in-loop" weapon equipment closed-loop striking.

2 Related work

2.1 MBSE method

The concept of MBSE is a formal modeling methodology based on model expression and driven proposed by INCOSE in 2007 [20], which includes three parts: modeling language, modeling tools, and modeling methods. MBSE plays an important role in the design field of complex weapon equipment, which mainly used to analyze, design, and verify system requirements, architecture, and functions. As a specification guide for model-driven principles, methods, languages, and tools, MBSE guides the realization of the entire modeling process, and specifies various models and views of each model in the system development life cycle. As the principle of the entire architecture modeling, MBSE also defines the timing relationship between model views, which includes the methodologies of DoDAF [1, 7, 12, 21], OOSEM [10], and Harmony-SE [27]. These methodologies have been accumulated for a long time and summarized in practice under different system modeling backgrounds. Figure 1 shows the development roadmap of MBSE.

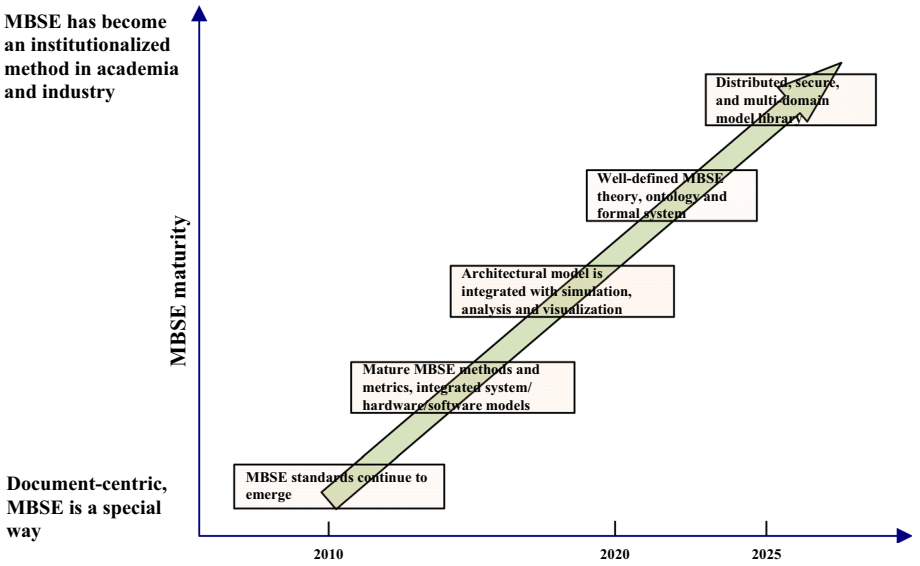

Fig. 1 MBSE development roadmap

2.2 Modeling language

The SysML is one of the most widely used graphical modeling languages in MBSE.At present, SysML language has replaced UML language as the standard modeling language in the field of systems engineering to realize complex weapon systems design. The description analysis and design inspection of the system by using SySML can improve the quality of system design and the interactive ability in system engineering. The SysML language adds demand diagrams and parameter diagrams, with more flexible and powerful functions, which can not only promote communication between system engineers, but also promote communication and coordination across rules and development cycles. At the early stage of complex weapon equipment architecture design, SysML language facilitates the visualization of system engineers' design ideas, facilitates communication among all developers, avoids ambiguity, and improves the efficiency and accuracy of communication with system engineers in the whole of the system development life cycle. The SysML language defines three types of nine basic graphics in total. As shown in Fig. 2 Each basic graphics is a visual representation of the system structure model.

2.3 Modeling method

The MBSE method can be defined as a collection of related processes, methods and tools used to support the disciplines of systems engineering in a "model-based" environment. After MBSE was proposed, many research institutions and scholars began to conduct research on MBSE development the methods, such as DoDAF, Harmony, OOSEM, etc.

1) DoDAF

DODAF2.0 was launched on May 28, 2009 by US Army, which includes 8 viewpoints: All Viewpoint(AV), Data and Information Viewpoint (DIV), Standards Viewpoint(StdV), Capacity Viewpoint(CV), Operational Viewpoint(OV), Services

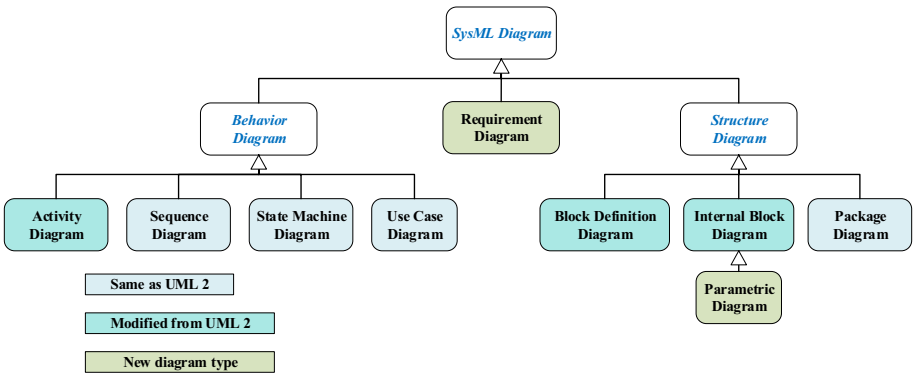

Fig. 2 SysML structure diagram

Viewpoint(SvcV), Systems Viewpoint(SV), Project Viewpoint(PV). The general process of DoDAF modeling, mainly focusing on the structure and functional decomposition of the system, which views describe the system from different sides and different angles.

2) OOSEM

The Object-Oriented Systems Engineering Method (OOSEM) is a top-down, model-based system engineering method developed by INCOSE using the SysML language. It originated from the cooperation with Lockheed Martin Company in the mid-1990s. The OOSEM is mainly used to capture and analyze system requirements, integrate software and hardware, and other professional design methods. In addition, the OOSEM method framework includes software and hardware research and development, which is particularly suitable for the research and development of complex weapon equipment. The development flowchart of the OOSEM methodology is shown in Fig. 3 OOSEM can be divided into simulation modeling settings, analysis of subjective requirements, analysis of system requirements, definition of logical

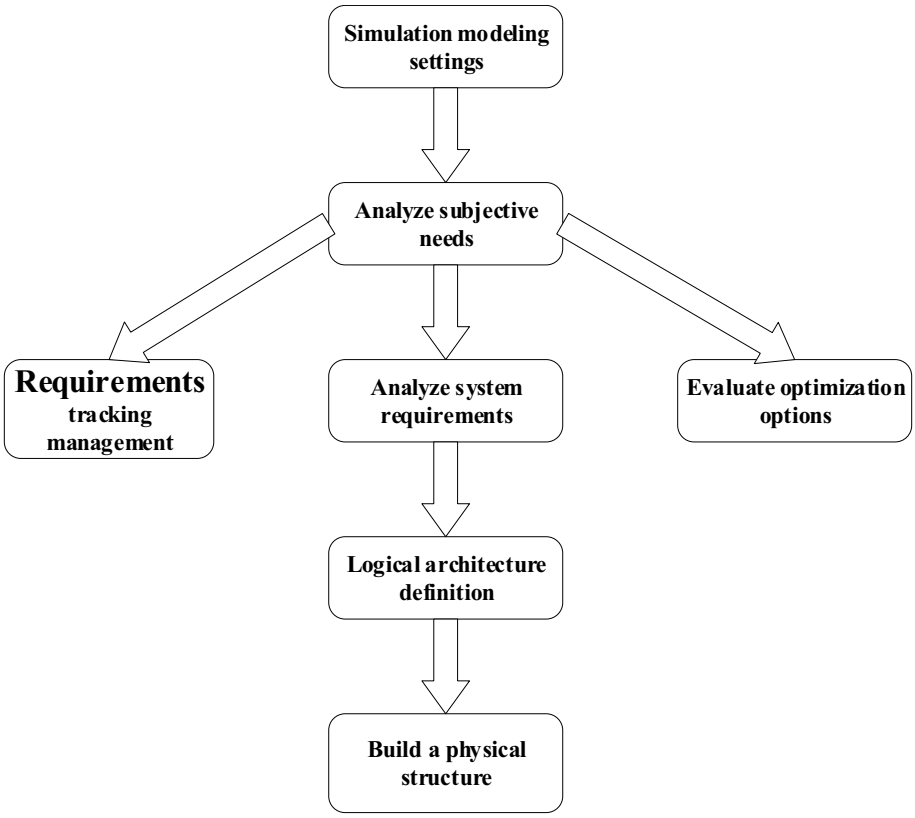

Fig. 3 OOSEM development flow chart

architecture, steps to build physical architecture, tracking management of requirements, and evaluation and optimization of alternatives. The simulation for complex weapon equipment interoperability based on MBSE proposed in this paper adopts the OOSEM method.

3 Visual interoperability simulation platform design

3.1 System schema design

Based on the model-driven visual simulation and verification technology of "human-in-loop" weapon equipment interoperability, the maned/unmanned combat unit under the background of information technology is used as the basic combat unit to construct an almost real virtual battlefield environment. Through flexible configuration of various conditions of "human-in-loop" simulation system conducts virtual combat simulation test to observe the results of the test confrontation for the researchers to analyze and evaluate. The system uses virtual reality technology, real-time battlefield interaction simulation technology, MBSE technology, etc. to implementation of the test consists of combat "human-in-loop" simulation combined with the MagicDraw modeling tool and UE4 engine. Figure 4 shows the system schema.

A typical virtual battlefield simulation confrontation system should be composed of red, blue, and white. All members are built in accordance with the distributed network structure specification design. The "white side" is the designer and manager of the entire simulation system act as a referee, which mainly responsible for the editing of the mission scenario, configuring each unit node in the early stage of the simulation, and the real-time monitoring of the process of whole simulation. The "blue side" consists of several aircraft carriers and several destroyers. The "red side" consists of serval "human-in-loop" simulation

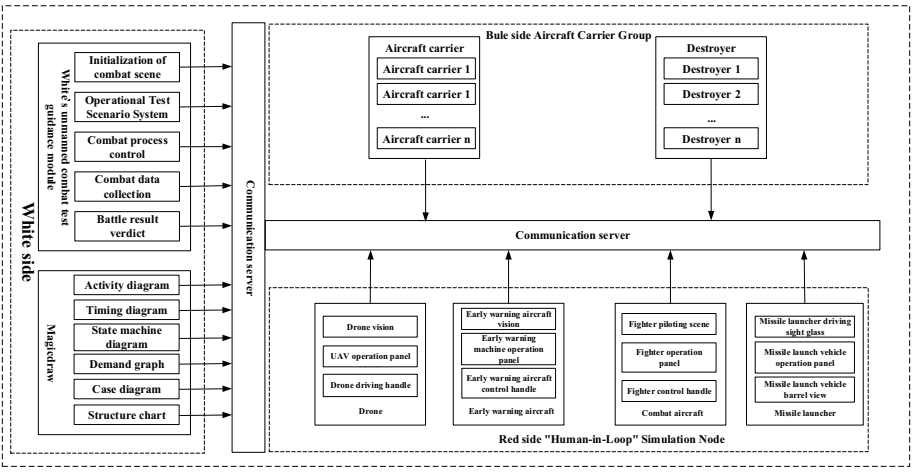

Fig. 4 System Schema Diagram

nodes such as drones, early warning aircrafts, fighters, and missile launch vehicles. In general, the deployment configurations of the red and blue parties are under the unified management and control of the "white side". Based on the combat background and mission, conduct actual combat simulation, and record the simulation result data set by continuously changing the deployment configuration of the red and blue parties to achieve the coordinated combat effect of weapons and equipment in multiple real combat simulation.

3.2 System framework design

This article uses MagicDraw as a weapons equipment architecture modeling tool, and mainly uses activity diagrams and state machine diagrams to construct a dynamic model of the weapons equipment architecture and a discrete event-driven behavior model, combined with the UE4 engine platform as a continuous scene visualization analysis tool. The system is driven by various external response events through the actual equipment architecture model, which will produce continuous complex behavior patterns, avoiding the emergence of equipment, making the entire equipment interoperability visual simulation process and equipment demonstration process are realized under the model interaction mechanism. In addition, the physical and behavioral models of weapons are set up in MagicDraw, and the structural models of weapons and equipment are quantitatively analyzed through the visual analysis tools in the virtual battle scene of UE4. The whole process of simulation software platform with the communication server, enables bi-directional data interoperability mechanisms. According to system requirements and mission scenario analysis, this paper designs a "human-in-loop" visual simulation for weapon equipment interoperability based on MBSE, through the establishment of 3D and behavior models of weapons including UAVs, early warning aircraft, fighters and missile launchers and construction of a virtual simulation platform based on UE4 to realize the weapon collaborative confrontation process and scene simulation, which simulates the interoperability process of the weapons and verify the performance requirements of the system. The system framework diagram shows in Fig. 5.

3.3 Communication server design

For the model-driven "human-in-loop" visual simulation and verification for weapon system interoperability, the communication interface between MagicDraw and UE4 uses UDP protocol to establish the information delivery between operating vision and SysML models such as UAVs, early warning aircraft, fighters, and missile launchers. The combat and status process generated by MagicDraw, through the communication server to drive the UE4 vision to realize the visual simulation of equipment in the virtual battlefield. At the same time, the decision result in the visual scene will be sent to MagicDraw for process judgment through communication server. The simulation system composed of multiple combat unit nodes is connected to each other through the UDP/IP network to realize the "human-in-loop" simulation of weapons operation.

Figure 6 shows the UDP communication process diagram of the communication server. It is developed in C# language to realize socket interface of UDP

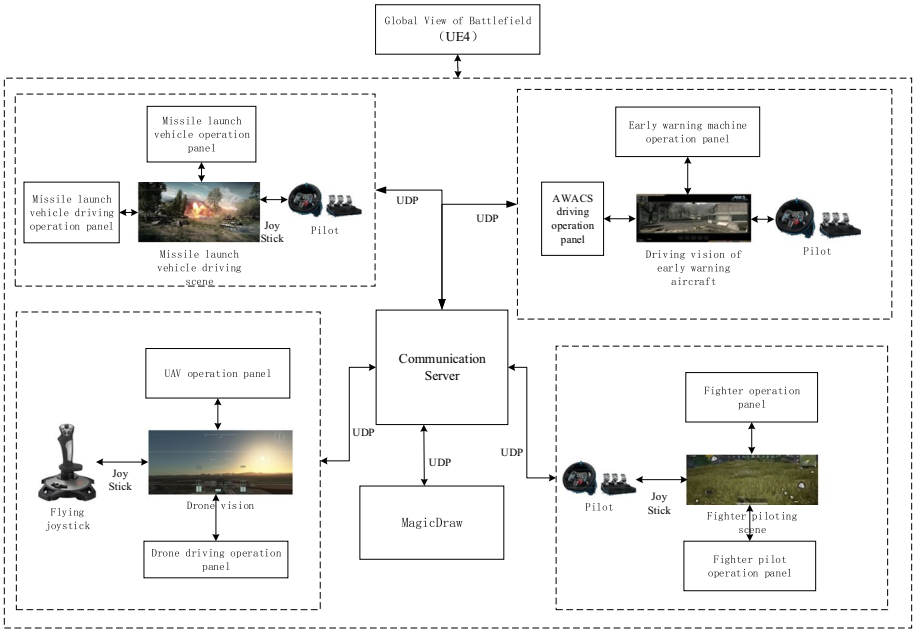

Fig. 5 System framework diagram of the platform

protocol. First, set and bind the IP address and Port number of the communication server. The server starts a new thread of *ReceiveFrom()*, which responsible for receiving and forwarding the UDP data message between clients. The clients can be MagicDraw, hardware, control panel, or UE4 virtual combat environment, which creates a simplified process of establishing a connection between the server.

3.4 Visual simulation environment design

The interoperability visual simulation system for complex system equipment is to place the equipment in a real running environment to achieve simulation missions. It is necessary to construct a realistic operating scene with comprehensive expressiveness, generate a virtual operating environment consistent with the real environment, put the equipment model of the complex system that needs to be simulated into the environment, and perform simulation verification on various tasks of the equipment system. The construction of the virtual natural environment includes terrain, virtual plants, roads, electromagnetics, cloud scenes, etc. The use of virtual reality technology to build a 3D visualization scene of a complex running environment has important practical significance. Compared with other virtual engine platforms, UE4 engine has particularly strong data processing capabilities, and the rendering function is also very powerful. UE4 uses blueprint programming to create scenes that trigger events or actions, without the need

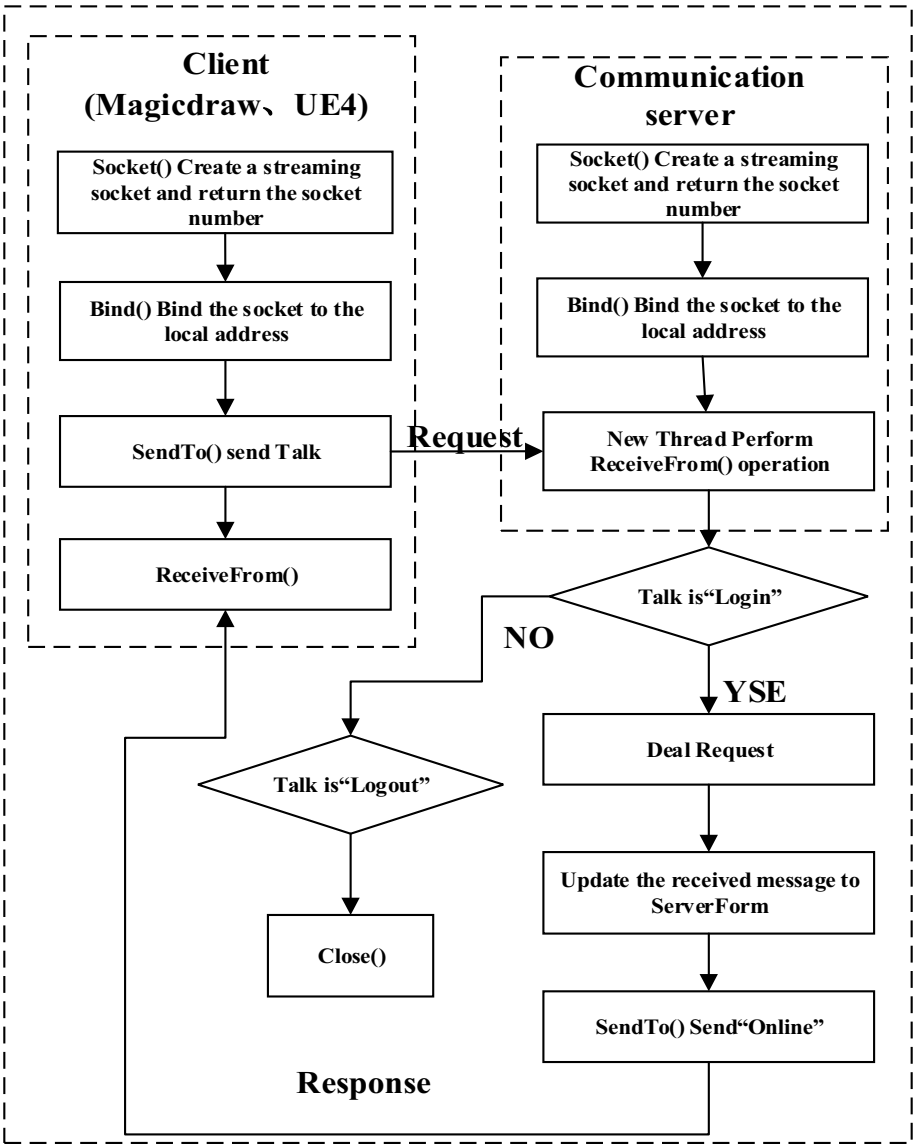

Fig. 6 Process of communication server

to write complex script files, and is compatible with many 3D modeling tools. For example, objects created by modeling software such as 3DS MAX or Maya can be imported into the UE4 engine platform for editing. Users can build an extremely realistic virtual simulation scene, which is super coupled with equipment, characters, entities, and environmental objects. In addition, the UE4 engine has a very powerful terrain and surface description system, which can process the detailed attributes of vegetation through the material editor to make it conform to

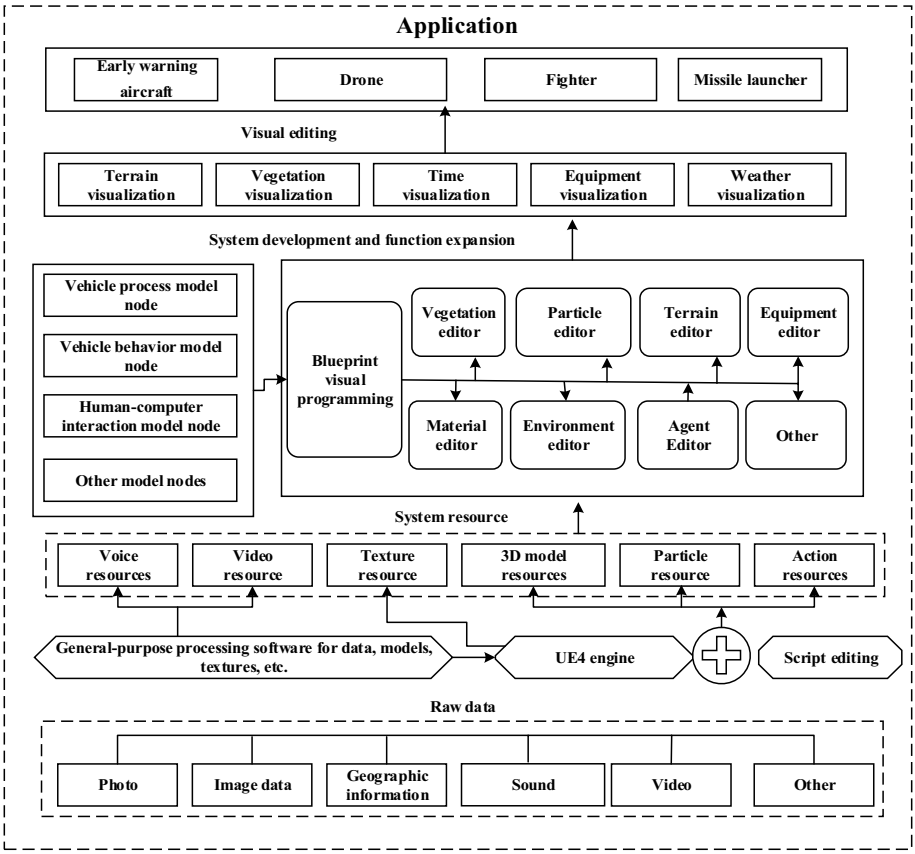

Fig. 7 Virtual running environment framework constructed by UE4

the natural growth law, and quickly generate surface vegetation that conforms to the natural growth law to achieve lifelikeness in accordance with the actual natural environment effect. The 3D virtual running environment not only requires the simulation process to be scientific and reasonable, but also improves the fidelity of visualization and achieves a high degree of coupling between the scene and the simulation model. Figure 7 shows the overall framework of the 3D virtual running environment.

The virtual running simulation environment framework mainly consists of four layers, including the system resource layer, the system development function extension layer, the visualization editing layer, and the application layer. The system resource layer is mainly used for pre-processing some video, image, geographic information, and other data, which converts the data format to make it compatible with the UE4 engine platform interface. The system development function extension layer is mainly for the secondary development of the software platform, adding new function modules and interfaces, etc., to provide various functional interfaces for the visual editing layer. The visual editing layer is mainly using some editors that come with the UE4

platform to process the detailed attributes of the environment objects such as vegetation, terrain, and equipment in the virtual environment to make them more realistic and suitable. The application layer mainly includes geometric models and behavior models of complex system equipment to achieve specific simulations based on task requirements.

3.5 Interoperability plug-in of MagicDraw design

If the modeling tool MagicDraw wants to transmit with UE4 by the communication server, it must develop a plug-in of MagicDraw to realize UDP protocol. The plug-in is a bridge between MagicDraw and UE4, which gets the messages, parameters, signals of the model in MagicDraw and transmits the information to UE4 to drive the weaponry entity model. The data message generated in MagicDraw is transmitted to the communication server through the plug-in, and then the data message is forwarded to UE4 through the communication server driving the weapons equipment entity model in UE4 to achieve the corresponding combat mission tasks. After the end, the results of the task execution are fed back to the MagicDraw by the plug-in in the form of data or messages, and then transmit to the SysML model in MagicDraw until the simulation is finished. The MagicDraw plug-in connects MagicDraw with the outside world to receive/transmit the data as a "bridge". As shown in Fig. 8, the interoperable plug-in design is shown.

The development of MagicDraw plug-in uses Java language and Eclipse as IDE. According to the official MagicDraw secondary development document, it needs to include the corresponding dependent jar packages and add them to the specified classpath. The simulation_api.jar package is mainly used to support the MagicDraw interface package for simulation. The plug-in must contain at least one class derived from the com.nomagic.magicdraw.plugins.Plugin class, the code is as follows:

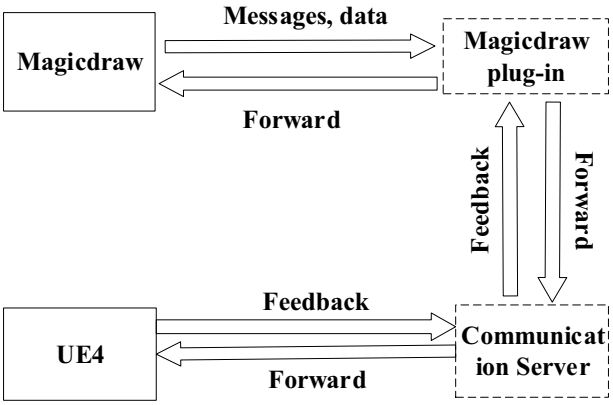

Fig. 8 Interoperability plug-in of the MagicDraw

```
package MagicDraw;
import com.nomagic.magicdraw.plugins.Plugin;
public static String serverAddress="192.168.199.243";
public static int serverPort=1110;
private static UDPClient udpClient;
private MyTransactionListener mTransactionListener;
                    @Override
                        public void init()
                        {

                        initialized = true;
                        udpClient=new UDPClient(serverAddress,serverPort);
                        String login="login|MagicDraw";
                        udpClient.SendMessage(login);
ActionsConfiguratorsManager manager = ActionsConfiguratorsManager.getInstance();
manager.addMainMenuConfigurator(new MainMenuConfigurator(getSubMenuActions()));
mTransactionListener = new MyTransactionListener();
AnyExecutionListener executionlistener = new AnyExecutionListener(udpClient);
SimulationManager.registerExecutionListener(executionlistener);
Application.getInstance().getProjectsManager().addProjectListener(new
ProjectEventListenerAdapter()
{
                        @SuppressWarnings("deprecation")
                        @Override
                        public void projectOpened(Project project){
TransactionManager transactionManager = project.getRepository().getTransactionManager();
                        transactionManager.addTransactionCommitListener(mTransactionListener);
                        registerListenerToSmartEventSupport(project);
AnyPropertyChangeListener listener=new AnyPropertyChangeListener(udpClient);
                        project.addPropertyChangeListener(listener);}
                        @Override
                        public void projectClosed(Project project){
project.getRepository().getTransactionManager().removeTransactionCommitListener(mTransacti
onListener); }});
}
                        @Override
                        public boolean close()
                        {
                                String logout="logout|MagicDraw";
                                udpClient.SendMessage(logout);
                                return true;
                        }
```

During the simulation process, the state and signal of the SysML model need to be transmitted to the Server and sent to UE4. Therefore, in the process of plug-in development, the state or filtering operation is performed, and the required data is "talk|MagicDraw|UE4Client|XXXX "Format to send to UE4, Fig. 9 shows the MagicDraw plug-in developed by Eclipse IDE.

3.6 Communication and blueprint design

As part of the visual simulation, the model of UE4 needs to be driven according to the MagicDraw message forwarded by the communication server. Therefore, UE4 also needs

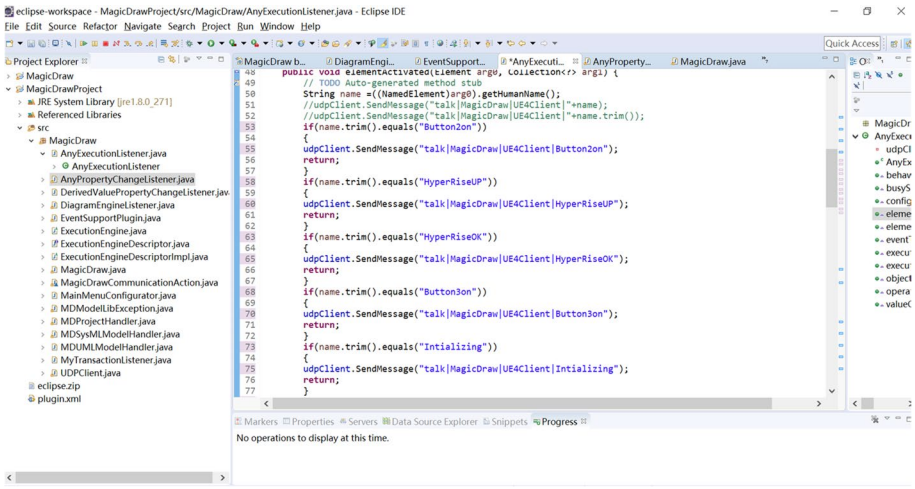

Fig. 9 MagicDraw Plug-in developed by Eclipse IDE

to develop UDP communication module in order to receive or forwarded messages to Communication Server. First, add a C++class named UDPClient in the project file browser of UE4, and select its parent type as Actor. After the addition is completed, the corresponding .h and .cpp file will be automatically added to the project source file. The Sockets module of UE4 encapsulates the socket work flow. First, Sockets.h is included in the UDPClient.h file to call the Sockets module. The UDP communication function required by this project is realized by writing a constructor. The specific constructor's name and function method are as follows:

1) BeginPlay(): Used to obtain the IP and Port number ofthe server and client.
2) InitSocket(): Used to bind the IP and Port of the UE4 client, then set the buffer size to check the buffer interval, and bind the callback function for receiving messages.
3) OnUdpDataReady(): Used for data reception and judgment. Define a data receiver ReceiveData to determine the received data to judge the message whether the well or fault defined communication structure, then receive the data and copy to the receiver.
4) SendToServer(): The message can be sent to Communication Server according to the Ip and Port of the Communication Server.
5) EndClient(): Send the message of "logout|UE4Client" to Communication Server, which means that UE4 client login out of the Server and ends UDP communication.

After completing the development of the UDPClient class, it needs to use the UDP communication function in the scene. Here we create an Actor named MyUDPClient and use the blueprint to inherit the methods in the UDPClient class. At the beginning of the event, call the InitSocket() function to bind the address of the UE4 client, and then send "login|UE4Client" to the Communication Server through SendToServer() waiting for the communication between UE4 and the Server is established. The specific development logic is shown in Fig. 10.

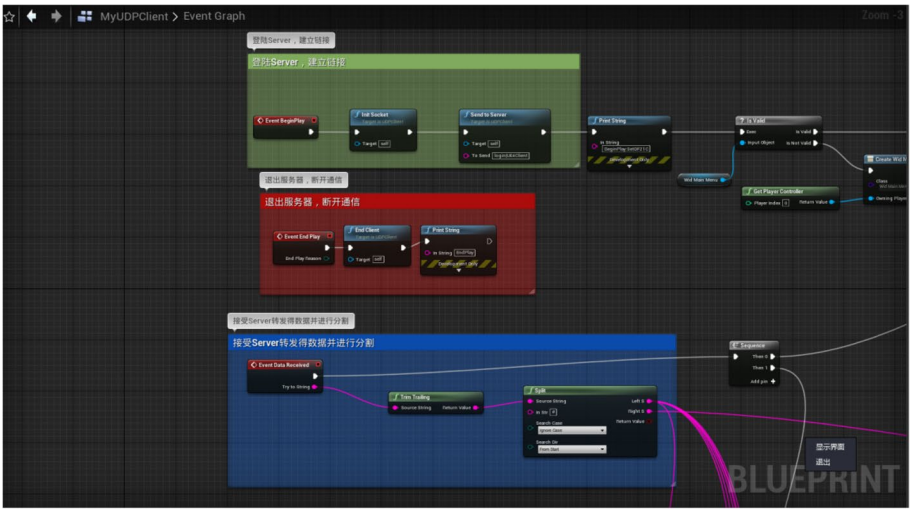

Fig. 10 Blueprint communication of UDPClient

After the blueprint of communication is realized, the received message needs to be processed and useful instructions are extracted. According to the data format of server forwarding to client: "talk|MagicDraw|UE4Client|CommandMessage", the received message is processed. The content of CommandMessage is put into the data receiver and forward to call the corresponding blueprint to act the interactive operations, which shows in Fig. 11.

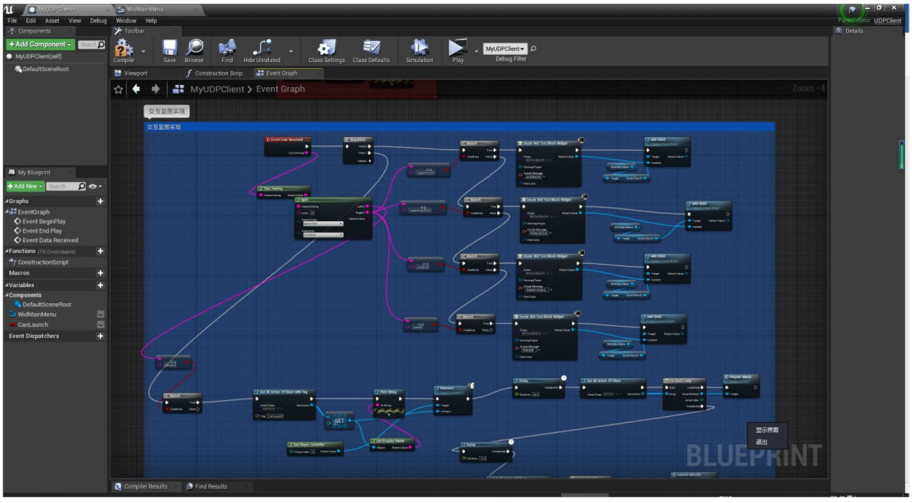

Fig. 11 Blueprint of Interactive call interactive operations

4 Visual simulation for complex weapon equipment interoperability

4.1 Mission scenario

Taking the island chain conflict that may occur in the future as a task background of this paper, weighing various combat plans, and based on the intelligence of all parties. Based on the coordinated operations of reconnaissance satellites, communication satellites, early warning aircraft, fighters, missiles, unmanned aerial vehicles, and joint command centers, the goal is to break through the enemy's aircraft carrier fleet air defense system and destroy the enemy's aircraft carrier formation. The process is decomposed and sampled, and the simplified diagram of the battlefield situation is shown in Fig. 12.

4.2 Model of the system

The system model of the complex weapon equipment includes variety behaviors, states, flows used to describe the equipment's driving, strike, flight, reconnaissance, etc., which can be established by using the activity diagram, state machine diagram, and sequence diagram of the modeling tool MagicDraw. The models usually describe a series of discrete state behaviors of the dynamic system models of complex weapon equipment. For example, activity diagrams describe a series of discrete behavior activities triggered by the complex weapon equipment in order to realize the corresponding functions, which is a simulation result at a critical time point, and the progress of the sequence of activity execution. The state machine diagram describes the state transition process of the key time node in a certain simulation operation process, which is atomic and consistent. The behavior diagram is equivalent to the task scenario logic diagram of the equipment unit to express the behavior of complex weapon equipment. Figure 13 shows the SysML model of the entire combat scene, reflecting the coordination and signal interaction between all levels of weaponry and equipment, which is the most intuitive display of the coordinated operation of the entire weapon equipment and the core part of the weapon equipment architecture design framework.

As shown in Fig. 14, this article established a UI interface diagram of a simulated command center in order to simulate the process of human control in loop and make the

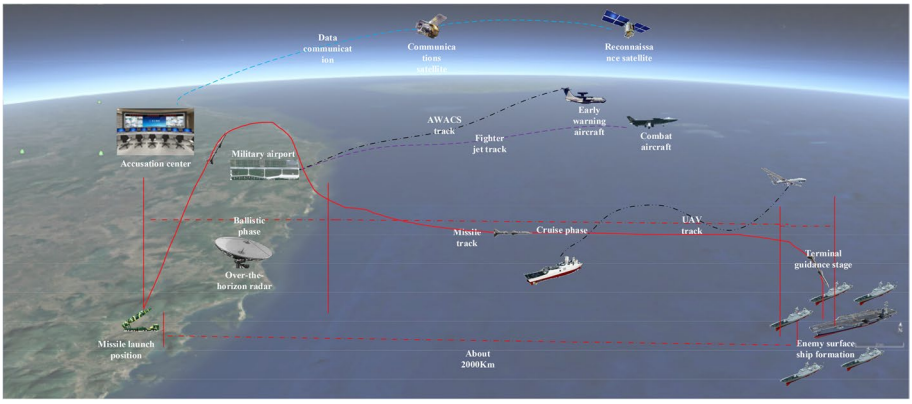

Fig. 12 Schematic diagram of coordinated attack combat situation

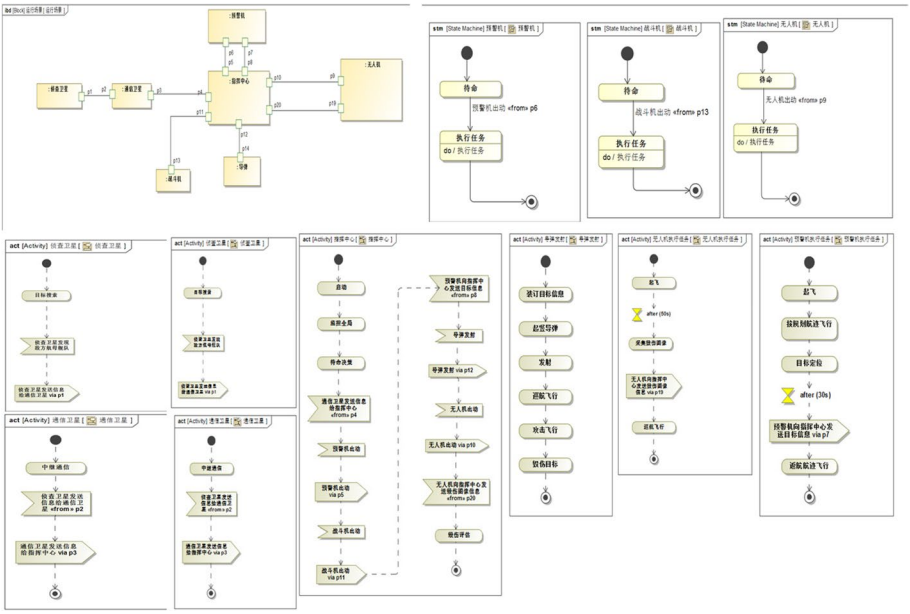

Fig. 13 Framework diagram of SysML model construction

simulation demonstration more in line with the operational process. The operator simulates the control of virtual equipment by clicking on the signal buttons on the UI interface, such as alert aircraft launch, unmanned aircraft launch, fighter aircraft launch, and missile launch, in order to execute specific behaviors and make corresponding decisions based on the results of the simulation screen.

4.3 Visual simulation

Through the SysML model in Magicdraw, data messages are transmitted through the communication server to drive the equipment entity model in UE4 for visual simulation of coordinated weapons operations. The real-time calculation of the position change and attitude data of weapon equipment by UE4, and Magicdraw controls the state parameters of weapon equipment, analyzes the demonstration work of weapon equipment system structure, and further improves the design index of the systems design according to the analysis results. Figure 15 shows the simulation operation diagram of MagicDraw.

Fig. 14 UI interface diagram

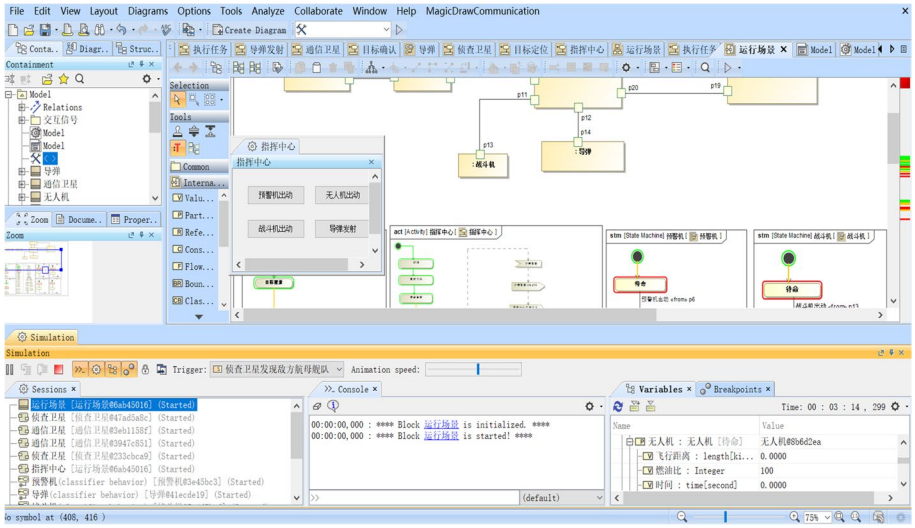

Fig. 15 MagicDraw simulation

As shown in Fig. 16, there is a simulation scene of some weapons equipment in the UE4 environment and related data messages of the communication server. The whole process is that the weapon units at all levels cooperate with each other according to the task background to attack the target at the critical time. By analyzing the simulation results including position information, speed, fuel thrust, etc., and comparing them with the requirements, the design parameters are modified and optimized to achieve the seamless connection between the design and the requirements.

Through co-simulation, the environmental factors in UE4 are constantly changed, including weather, day and night, wind speed, magnetic field, etc., Operators can participate in battlefield decision-making by simulating the battlefield commander using the UI interface diagram drawn by MagicDraw, as shown in Fig. 16. When the warning aircraft, unmanned aircraft, and combat aircraft deployment buttons are pressed, the simulation screen switches to the corresponding aircraft type taking off from the aircraft carrier and going to execute the mission. When the missile launch button is pressed, the screen switches to the coastal battlefield, where multiple missile launch vehicles launch missiles towards the target, to achieve interoperable and cooperative combat simulation of weapons and equipment in different environments. According to the analysis of simulation results at key time points, such as the amount of fuel at a certain moment, flight distance, strike acceleration, strike distance, etc., the goal of model-driven "human-in-loop" weapon equipment system demonstration is realized ultimately.

5 Conclusion

This paper proposes a model-driven human-in-loop architecture design method for weapon equipment based on the challenges and difficulties brought by system complexity and emergence that cannot be solved by traditional document-based architecture design methods. By studying the interoperability mechanism between Magicdraw and UE4, a

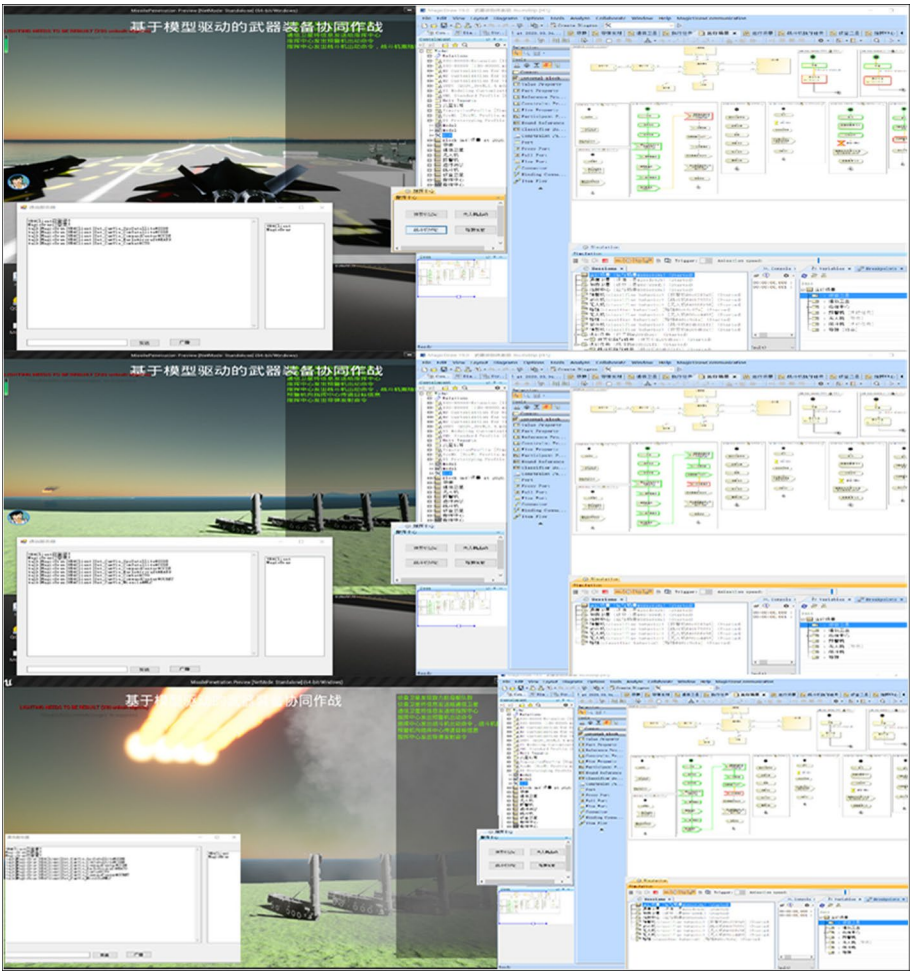

Fig. 16 UE4 simulation scene

new weapon equipment architecture design method is proposed, which drives the SysML model of Magicdraw and the equipment entity model in UE4. The Harmony-SE framework, which is used for complex system structure design, is studied and its model-driven design concept is applied to military architecture design. In order to address the difficulties faced by the Chinese military's weapon equipment verification work and the complexity and emergence of modern weapon equipment architecture design that cannot be solved by traditional document-based architecture design methods, a complete architecture modeling process framework is established. The framework clearly illustrates the workflow and analysis process of weapon equipment architecture modeling. The Magicdraw modeling tool and the Unreal Engine 4 are used for integrated platform simulation to visualize the collaborative combat of weapon equipment, and to optimize and improve the blueprint logic of the SysML parameter model in Magicdraw and the equipment entity model in UE4 based on the analysis results. The goal is to conduct a comprehensive platform demonstration of the model-driven human-in-loop architecture design of weapon equipment systems, to

ensure that the weapon equipment system meets the combat requirements and maximizes the combat effectiveness of the system.

Declarations

Conflict of interest The authors declare that they have no conflicts of interest to report regarding the present study.

References

1. Aghamohammadpour A, Mahdipour E, Attarzadeh I (2022) Architecting threat hunting system based on the DODAF framework[J]. J Supercomputing 79(4):4215–4242
2. Anyanhun A, Adejokun AP, Hause M (2022) An MBSE architectural framework for inter-satellite communication in a multiorbit disaggregated system[J]. INCOSE International Symposium, vol 32, issue 1, pp 665–685
3. DeLaurentis DA, Crossley WA, Mane M (2011) Taxonomy to guide systems-of-systems decision-making in air transportation problems. J Aircr 48(3):760–770
4. Good MR, Sturtevant GH (2020) Technology insertion OODA loop strategy for future flexible surface warship acquisition and sustainment, vol 132, issue 2, pp 59–77
5. Graves H, Bijan Y (2011) Using formal methods with SysML in aerospace design and engineering. Ann Math Artif Intell 63(1):53–102
6. Gregory H, Berthoud L, Tryfonas T, Rossignol A (2020) The long and winding road: MBSE adoption for functional avionics of spacecraft. J Syst Softw 160:110453. https://doi.org/10.1016/j.jss.2019.110453
7. Hongxing Z et al (2021) The on-orbit mission analysis of OTV based on DoDAF[J]. Aircr Eng Aerosp Technol 93(6):937–945
8. Huang YY (2015) Modeling and simulation method of the emergency response systems based on OODA. Knowl Based Syst 89:527–540. https://doi.org/10.1016/j.knosys.2015.08.020
9. Jacobs J, Simpson A (2017) On the formal interpretation and behavioural consistency checking of SysML blocks. Softw Syst Model 16(4):1145–1178
10. Leserf P, De SSP, Hugues J (2019) Trade-off analysis for SysML models using decision points and CSPs. Softw Syst Model 18(6):3265–3281
11. Ling MF, Moon T, Kruzins E (2005) Proposed network centric warfare metrics: from connectivity to the OODA cycle. Mil Oper Res 10(1):5–13
12. Liu B, Wu XY (2012) Mission reliability analysis of missile defense system based on DODAF and Bayesian networks. Information 15(12B):5659–5666
13. Lu JZ, Wang GX, Torngren M (2020) Design ontology in a case study for cosimulation in a model-based systems engineering tool-chain. IEEE Syst J 14(1):1297–1308
14. Mazeika D, Butleris R (2020) Integrating security requirements engineering into MBSE: profile and guidelines. Secur Commun Netw. https://doi.org/10.1155/2020/5137625
15. Pandey M (2022) A generic hierarchical System of Systems Engineering (SOS) approach for model based system Engineering (MBSE) Projects[J]. INCOSE International Symposium, vol 32, issue 1, pp 737–766
16. Planas E, Cabot J (2020) How are UML class diagrams built in practice? A usability study of two UML tools: Magicdraw and Papyrus. Comput Standards Interfaces 67:103363. https://doi.org/10.1016/j.csi.2019.103363

17. Rogers EB, Mitchell SW (2021) MBSE delivers significant return on investment in evolutionary development of complex SoS. Syst Eng. https://doi.org/10.1002/sys.21592
18. Silingas D, Butleris R (2009) Towards implementing a framework for modeling software requirements in MagicDraw UML. Inform Technol Control 38(2):153–164
19. Sprock T, Bock C (2020) SysML models for discrete event logistics systems. J Res Natl Inst Stand Technol 124:125023. https://doi.org/10.6028/jres.125.023
20. Squires A, Cloutier R (2010) Evolving the INCOSE reference curriculum for a graduate program in systems engineering. Syst Eng 13(4):381–388
21. Tao ZG, Luo YF, Chen CX, Wang MZ, Ni F (2017) Enterprise application architecture development based on DoDAF and TOGAF. Enterp Inform Syst 11(5):627–651
22. Wan W, Cheong HM, Li W, Zeng Y, Iorio F (2016) Automated transformation of design text ROM diagram into SysML models. Adv Eng Inform 30(3):585–603
23. Wolny S, Mazak A, Carpella C, Geist V (2020) Thirteen years of SysML: a systematic mapping study. Softw Syst Model 19(1):111–169
24. Yang HG, Fu X, Zhan ZH, Xin WG (2020) Parameterization dynamics visual design platform for missile launching system. Adv Mech Eng 11(2). https://doi.org/10.1177/1687814019827129
25. Yang HG, Li LY, Chen JX et al (2021) Visual dynamics simulation for adapters separation of missile launching. AIP Adv 11(2):025001. https://doi.org/10.1063/5.0043494
26. Zhang L (2019) Application research of automatic generation technology for 3D animation based on UE4 engine in marine animation. J Coast Res 93:652–658. https://doi.org/10.2112/SI93-088.1
27. Zhang TT, Wu JM, Qi L, Xu HY (2012) Architecture analysis and design language & harmony system engineering process. 2012 IEEE/AIAA 31st Digital Avionics Systems Conference (DASC), Williamsburg, VA