Research on Semantics implementation method of domain specific modeling language of air-combat decision behavior based on Python

Lei He

China Aerodynamics Research and Development Center, Mianyang Sichuan 621000,China

leiluodelei@nudt.edu.cn

Keywords: air-combat simulation; decision behavior; domain specific modeling language; code generation; python script

Abstract. The application of domain specific modeling language (DSML) in the decision behavior modeling of the air-combat simulation brings higher level of abstraction and friendly environment. However, the decision behavior model must be translated to executed code before it be used in air-combat simulation system. So, how to realize auto generation of code and improving the efficiency has become an important problem. A code generation framework and mechanism based on python script is designed, which provide the semantic algorithm of air-combat simulation decision behavior modeling language and model explain framework. Otherwise, based on the BON technology provided by generic modeling environment (GME), a code generator is built in VS2010 using C++, which really realize the auto code generation and avoid the error and time waste by manual coding.

Introduction

Facing the modeling problem of air-combat simulation decision behavior, traditional methods generally follow the paradigm: modeling—coding. Conceptual modeler constructs the conceptual model (CM) according to the requirement of air-combat simulation system and the process of actual air-combat. The CM is independent of simulation platform, its abstract level is higher and it can't be executed by computer. The CM must be translated into platform specific model (PSM) in order to be executed. In general, this work was performed by software engineers by manual coding. It not only increase the workload and errors but also inconvenient, and modification and validation of the tactics is extremely difficult [1]. Lastly, the model was applied into simulation system. One day, when the model must be modified to fit the new requirement, it would be painful for software engineers. Sometimes, they have to choose to throw away the old model and build again. Domain specific modeling (DSM) method provide an approach to solve this problem: build the domain model based on domain specific modeling language, then run the code generator to obtain executable code. A very important goal of DSM is one hundred percent code auto-generation [2].

One key factor of implementing the code auto-generation is describing the air-combat decision behavior logically. There are several approaches. It is a common scheme using the rule library to construct the decision behavior model directly, based on programing language such as C++. However, it lack agility, and only suitable to the case that the situation space is simple [3]. There is another approach which adopts script mechanism, it is available when the situation space is complicated [3]. Otherwise, considering the control character between the decision behavior model and physical model, the script can describe interaction behavior between them.

Python is an object-oriented script language. It supports dynamically input, and can be applied into numerous domain, especially computation science domain [4]. So design the air-combat simulation decision behavior modeling mechanism based on python script can effectively meet the requirement.

Framework and Meta-model

As illustrated in Fig.1, the design and implementation of DSM include three aspects: DSM language, code generator, domain framework. The code generator is the implementation of interpretation algorithm of DSML and it generates code with the information provided by domain model.

Application of DSM method is usually based on meta-modeling platforms, and mature platforms include MetaEdit+, GME, Microsoft DSL Tools, Dome, etc. In all of those, GME is popular. It is open source and provides a framework for meta-modeling and interpretation [5]. The implementation framework of decision behavior model based on GME is shown in Fig.2. MetaModel is the core of DSML, and user can build the conceptual model (Domain Model) by using DSML. The framework code of BON (Builder Object Network) model can be generated by MetaModel, and then build its DLL, which can interact with conceptual model to generate the code in GME environment.

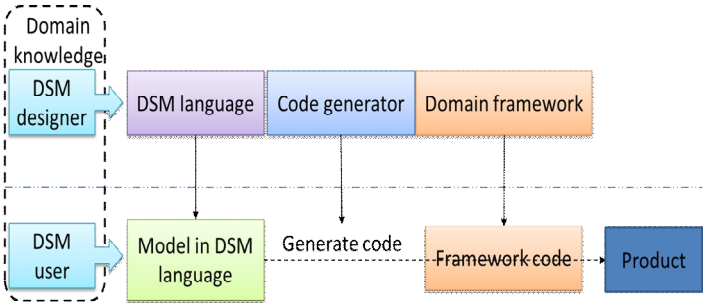

Fig. 1 The design and implementation of DSM

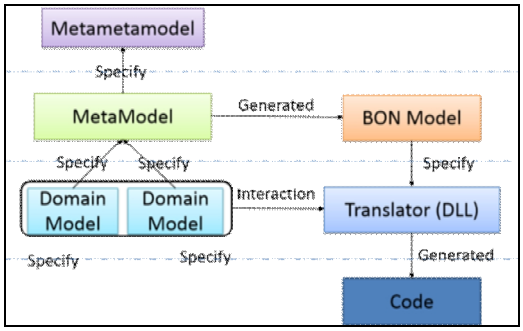

Fig.2 The Implementation framework of decision behavior model based on GME

The meta-model of air-combat simulation decision behavior was designed based on GME, as shown in Fig.3. The meta-model define the abstract concepts and relationship that used to describe the air-combat simulation decision behavior. The meta-model has referred “State Machine”, which include state and transition. The AbstractPhase in the meta-model is just like the “state”. For example, typical Phase types contain radar warning, form fly, lock warning, etc.

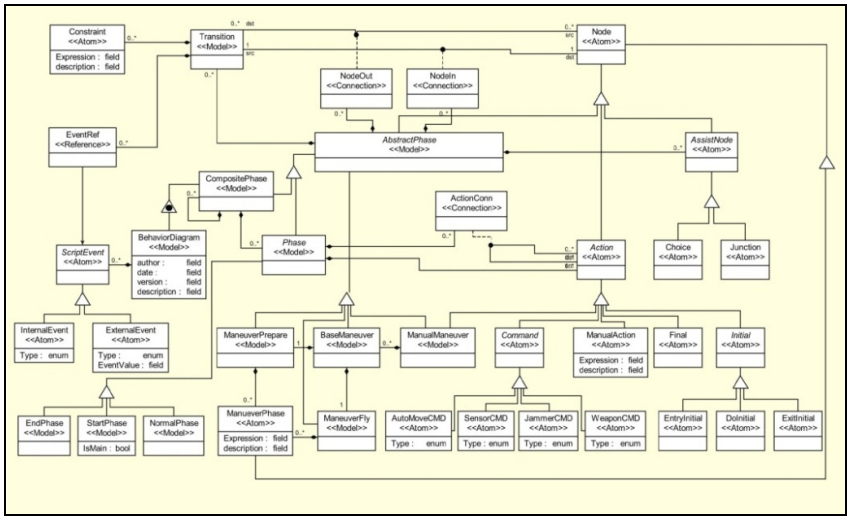

Fig.3 The meta-model of air-combat simulation decision behavior

Semantics implementation algorithm of DSML based on python script

The final purpose of the decision behavior model is embedded into air-combat simulation system with the models in physical aspect to support simulation. So the DSML of decision behavior should have explicit execution semantics that can be carried out by computer. The code generator traverses the CM to python script based on BON framework using the following Algorithm of generator:

1. Begin with the root model (BehaviorModel) and create a python script file.
2. Write the definition function of Initial Behavior in the python script, then in the function area write the execution code of StartPhase and find and traverse the ScriptEvent node list, write python code to bind each node with a response function.
3. Find the Phase node list, call the StateMachine algorithm to generate StateMachine function.

4. Traverse Phase node list, call the Phase Generation algorithm to generate python function for each Phase node, and remove the generated Phase from the list. Go to Step 5, when the list is empty.

5. Find and traverse the Transition node list, call the Transition Generation algorithm to generate python function for each Transition node, and remove the generated Transition from the list. Go to Step 6, when the list is empty.

6. Traverse the ScriptEvent node list, call the ScriptEventHandler Generation algorithm to generate python response function for each ScriptEvent node, and remove the generated node from the list. Go to Step 7, when the list is empty.

7. Generate other specific code to complete the script file.

The StateMachine Generation Algorithm:

1. Write the definition function of StepBehavior in the python script, then in the function area write the execution code to acquire the current phase of platform.

2. Traverse Phase node list, write the execution code to judge if the phase node is the current phase, when yes, write the execution code of the Phase function.

The Phase Generation Algorithm: the Phase node is a state, it may include EntryInitial、DoInitial and ExitInitial nodes, so it generate three execution function.

About EntryInitial node:

1. Write python node to set the current phase with the Phase. Begin with the EntryInitial, follow the output line(s) to next node(s).

2. Write a node execution code line for current node(s), if the node is a Transition, the next node execution line should be in the function area of the Transition node execution line.

3. if the output line(s) is exists, go to step 2; Otherwise, stop the EntryInitial node generation.

About DoInitial node:

1. Find the Transition nodes list which connected to the Phase node by NodeOut, if the list is empty, go to step 3, else go to step 2.

2. Traverse the Transition list, if current node contains ScriptEvent, then remove it from list, else, write an execution line for it, and remove the node from the list. Go to step 3, when the list is empty.

3. Begin with the DoInitial node, follow the output line(s) to next node(s).

4. Write a node execution code line for current node(s), if the node is a Transition, the next node execution line should be in the function area of the Transition node execution line.

5. If the output line(s) is exists, find the next node(s), go to step 4; Otherwise, stop generation.

About ExitInitial node: see the EntryInitial.

The Transition Generation Algorithm: Find all the Constraint contained by the Transition, if the Constraint exist, then write the execution code of all Constraint nodes in python script in order, otherwise, return true value.

The ScriptEventHandler Generation Algorithm:

1. Write python code line to acquire the current Phase node, then find the Transition nodes list that connected to the Phase node by NodeOut. And find Transition node which contains current ScriptEvent from all the Transition nodes list to form a new list.

2. Traverse the new Transition nodes list, write an execution line for current node, and remove the node from the list. Stop this generation, when the list is empty.

Script event response function must be bind to according ScriptEvent. There are two categories of event: Internal Event and External Event. The Internal Event is defined by user in the domain model, for example “WeaponFailed”, the response function is" WeaponFailedHandler", and the python implementation code for binding is as follows:

PlatformInfo.SubscribeEvent("WeaponFailed "," WeaponFailedHandler").

The Internal Event includes Timer Event and Simulation Event, for example, the python implementation code for binding is as follows:

PlatformInfo.ScheduleTimerEvent("TimerEvent","TimerEventHandler",4)

PlatformInfo.ScheduleSimulationEvent("SimulationEvent","SimulationEventHandler",11.5)

Other execution code line is easy, for example ,if it an order to open all radars, the code generated is: PlatformInfo.OpenAllRadar().

The technical framework of decision model based on python script

Although a Semantics implementation algorithm based on python script is discussed above, the air-combat simulation system must solve the follow problem: how does the python script based decision model interact with physical models. Thus, a technical framework is designed, as illustrated in Fig.4. Each platform model can bind a decision behavior script, the script describe the decision behavior of platform in detail. In running, at each decision time step, platform model will call own python script, the platform will perform the behavior given by the python script, and python script can also visit the API provided by tcPlatform. The tcPlatformTarget object provide target information for python script, and Timer provide time detecting function, the MemoryVariable support the python script to record state information. Timer, MemoryVariable and ScriptEvent are managed by tcPlatform.

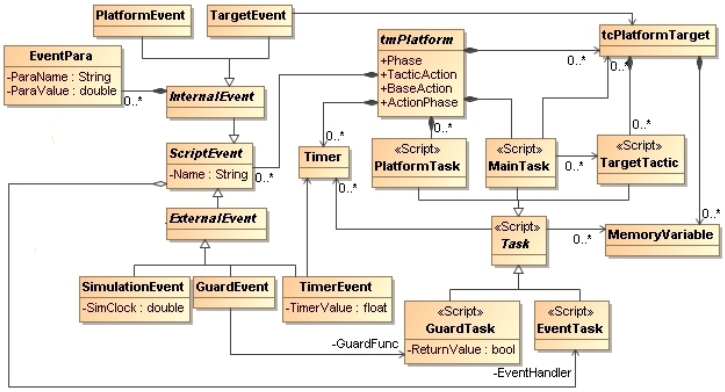

Fig.4 The technical framework of decision model based on python script

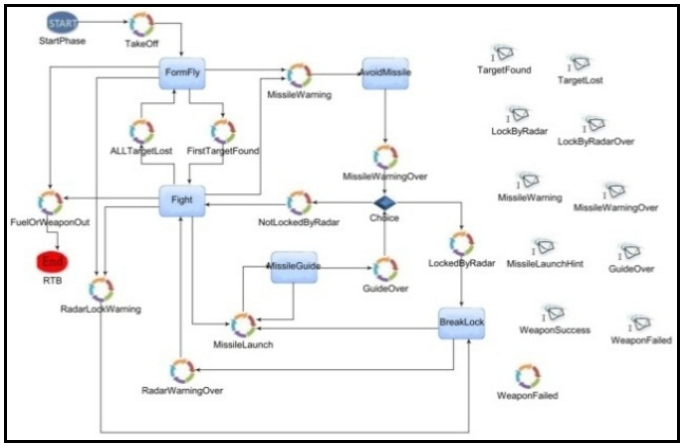

Fig.5 air-combat decision behavior conceptual model

Simulation result

As discussed above, the BON provided by GME has a powerful network of C++ objects. The generator is build based on BON framework using Visual Studio. The generator is not standalone software, it is components (DLLs) that is loaded and executed by GME.

Before running the generator, the user should construct air-combat decision behavior model, which is shown in Fig.5. It is a conceptual model. As illustrated in Fig.6, the python script which describes the decision behavior of domain model in Fig.5 is generated. The InitialBehavior function is the entry of model, and it call the entry function of start phase which is corresponds of the node StartPhase of model in Fig.5. Also, script events are bind in the function domain. The InitialBehavior function will be

called by platform in the simulation system at the start of decision. The StepBehavior is the implementation of state machine, and it will be called by the platform at every simulation decision step.

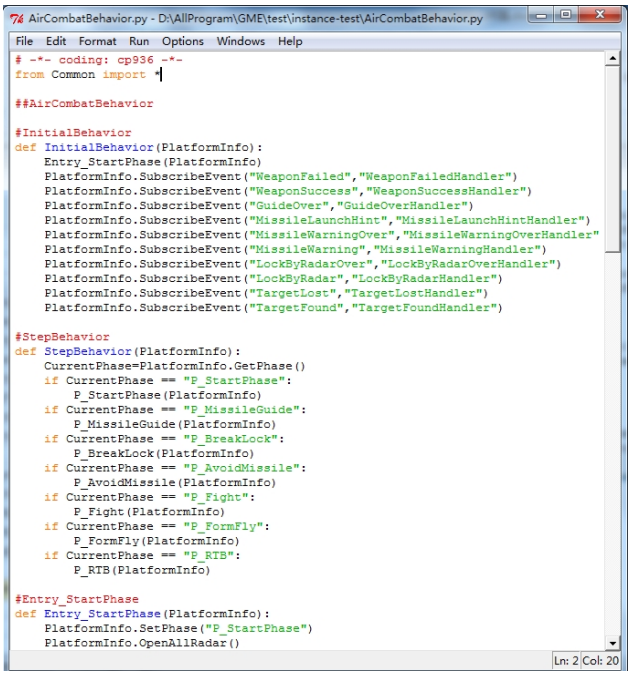

Fig.6 Automatically generated python script of decision model

Summary

A Semantics implementation method of DSML of air-combat decision behavior based on python script is proposed, it aims at absolutely code generation. The method make the computer to be able to process data from CM automatically, and also reduces the errors relative to manual coding. The method supports users build different decision behavior model according their preference. Otherwise, python script model can be easily added to the air-combat simulation system based on the technical framework.

References

[1] He Lei, Yao Jian, Lei Yonglin. Air-combat Decision Modeling Method Based On DSM. Applied Mechanics and Materials, 2014, 536: 416-402.

[2] Yang xintao, Su guiping, Wang rui, Wang xiaofang. Research and Implementation of DSM and Code Generation. Computer System Application, 2009, 18 (4): 100-103.

[3] Zhang wen. Research on model framework of combat behavior decision for fighter aircraft based on script. National University of Defense Technology, 2011.

[4] Lei Yonglin, Zhao Xin, Li Wei, Li Qun. Design and Implementation of Exploratory Analysis Software EASim. Journal of National University of Defense Technology, 2011: 99-104.

[5] Li X, Lei Y, et al. Domain-specific decision modeling and statistical analysis for combat system effectiveness simulation. Journal of Statistical Computation and Simulation, 2013: 1-19.