SimAIT:An Automatic Initialization Tool for Complex Simulation Systems

Sirui Bao
College of Information System and Management,National University of Defense Technology
137 Yanwachi, Changsha
China
baosirui15@nudt.edu.cn

Feng Zhu
College of Information System and Management, National University of Defense Technology
137 Yanwachi, Changsha
China
zhufeng@nudt.edu.cn

Yuliang Zhao
AllSim Technology Inc.
635LuGu Road, YueLu District, Changsha
China
zhaoyuliangyx@163.com

Laibin Yan
College of Information System and Management, National University of Defense Technology
137 Yanwachi, Changsha
China
Lbyan2000@163.com

Yiping Yao
College of Information System and Management, National University of Defense Technology
137 Yanwachi, Changsha
China
ypyao@nudt.edu.cn

ABSTRACT

A simulation execution essentially depends on the initial condition, necessitating correct and efficient initialization. However, at present most of the initialization work is done manually, which leads to high costs and error-prone code. To solve this problems, this paper designed and implemented an automatic initialization tool-SimAITconsisting of three modules. Firstly, scenario files are parsed and stored in a NestedHashMap Storage (NHMS). And then the adapter interfacefunction library is established for assigning data in NHMS to variables. Finally based upon the above modules, the initialization program is generatedbyautomatic code generationmodule. As the case study shows,SimAIT can initialize the simulation systems automatically and efficiently,and the time consumption is obviously reduced compared to that ofwriting setup code manually.

CCS CONCEPTS

•**Computing methodologies→Simulationtools**; *Massively parallel and high-performance simulations* •**Software andits engineering→** *Automatic programming*

KEYWORDS

Simulation systems, simulation scenario, automatic initialization, automatic code generation, nested hash map, function library.

SIMUTOOLS '17, September 11–13, 2017, Hong Kong, China

ACM ISBN 978-1-4503-6388-4/17/09…$15.00
https://doi.org/10.1145/3173519.3173537

1 INTRODUCTION

Since the 1990s, Europe and other developed countries began to research on parallel discrete event simulation (PDES) which has been widely used in large-scale ecological environment simulation, computational system biology simulation, large-scale combat simulation in a complex war environment, crisis prediction and decision making and so on [1]. This PDES support environments provides important experimental methods and tools for complex systems researches. And combat simulation technology is now widely used in military field. It is an effective and low-cost method to analyze modern warfare by establishing the vivid combat simulation system [2].

The first step of carrying out a simulation is to initialize the simulation system. And it is very important for the simulation system to be carried out smoothly and get the valid simulation data. The combat simulation system is a kind of complex simulation system, which is usually established and runs based on a predesigned scenario that provides detailed initialization data. A correct and efficient system initialization from the scenario is of great importance [3, 4].

Complexsimulation systems based on PDES usually consist of multiple simulation objects, and each object consists of multiple models [5, 6]. With the increasing scale of simulation, the initialization process becomes more and more complex. At present, the traditional modelling and developing method is to write code by engineers manually, which is inefficient and tedious. First the involved engineers need to understand the initialization requirement and data organization in scenarioand then find the corresponding initialization data in scenario to code the initialization program manually. However, once the scenario files are modified, the engineers need to manually modify the code and recompile. This handwork handling may lead to low efficiency,

mistakes and heavy workload problem. To sum up, the current system initialization has a high degree of manual-work dependence, tight coupling between processes, low efficiency and low degree of automation.

To solve these problems, this paper designed and implementedan automatic initialization tool–SimAITfor complex simulation systems. We first designed a nested HashMap storage structure to save the parsing scenario files, which can solve the heterogeneity problem between the scenario data structure and the variables needed to be initialized in the simulation system. Also, we provide aninterface library of assignment function to establish the bridge between the cached scenario data in HashMap and the variables of the simulation system, so as to establish the mapping relationship between the data and attributes. And then, the simulation system initialization codeis generated inthe automaticcode generation module, which can improve the efficiency of code writing and the correct rate. And the generated initialization code could assign the variable of simulation object (SimObj) instance to the initial value in simulation system initial phase.

The case study shows that the designed initialization tool SimAITcan initialize the simulation system automatically and efficiently. This tool reduces heavy manual-work dependence and improves the system development efficiency.

The paper is organized as follows. Section 2 gives the related works on the system initialization. Section 3 introduced the architecture of automatic initialization tool which includes three modules. Section 4 presents a case study using this tool for system initialization. At last, the conclusion and future work is presented in Section 5.

2 RELATED WORKS

About the issue of system initialization, there are some studies both on Data Preparation and Data Initialization. Concerning Data Preparation, a main direction is to establish the standard data exchanging format. The representative work is Military Scenario Description Language (MSDL) [7]. Being Standardized by SISO, MSDL provides a standard format for data description in scenario [8]. Scenario based on MSDL can be shared among C4I system and military simulation system [9]. As for Data Initialization, [3, 4] [10] studied the initialization issue of different military simulation system. The focus of these studies, however, still lies in the data exchanging format. The concrete process of variable assignment is ignored. Reference [11] studied a MDA based transformation between the universal simulation scenario and simulation script, based on which the system runs. There still exists the issue of how to initialize the system from the simulation script. Reference [12] realized a similar transformation, which is between simulation scenario and database. In the field of non-military simulation, reference [13] studied the initialization of power system simulation. The initialization is a two-stage process: finding out the variables to be initialized, and data matching for each variable. A method for parallel initialization of distributed simulation systems is proposed in [14]. In this method, object instantiation and setup are split in Config and Post Bind phases allowing distribution of initialization task among different nodes and removing the ordering requirement between the interdependent objects. But as for variable assignment, it is loadedmanuallyrather than automatically. However, as forcomplex simulation systems based on PDES, the process of variable assignment is quite complicated with a heavy workload. To sum up, it is of great significance to study on the automatic initialization.

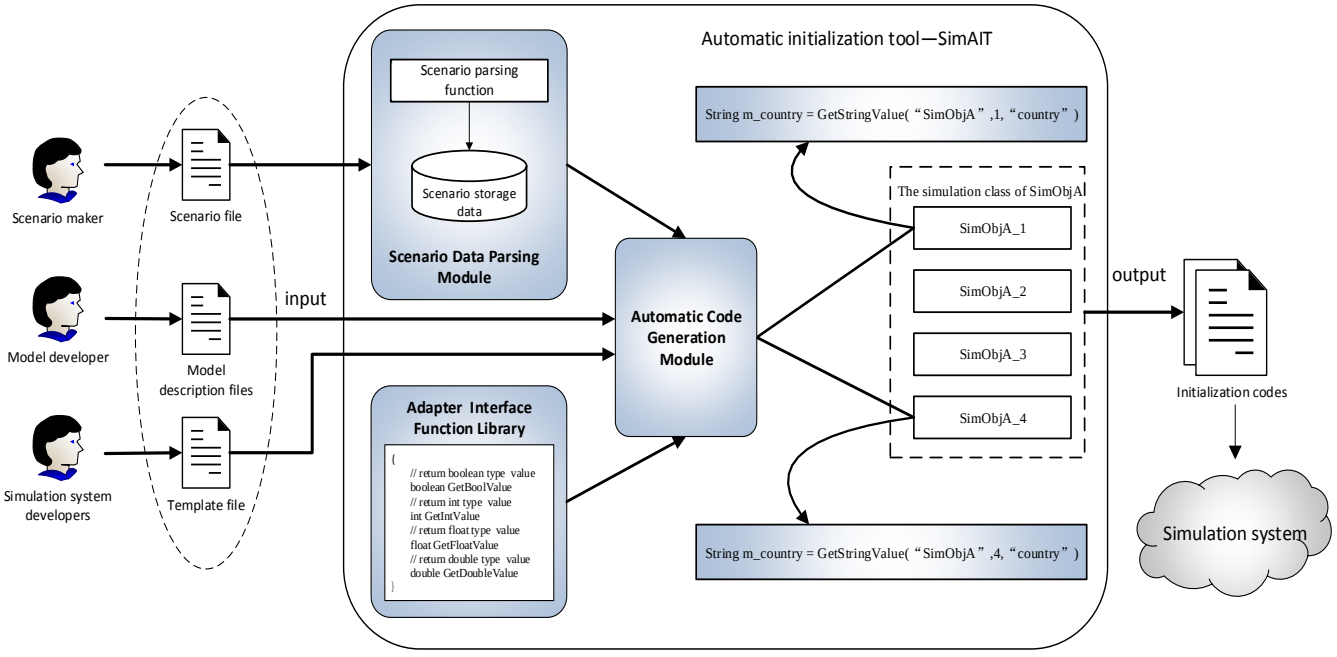

Figure.1:The architecture of SimAIT.

3 ARCHITECTURE OF THE AUTOMATIC INITIALIZATION TOOL

To obtain a more efficient and flexible initialization for complex simulation systems, we designed an automatic initializationtool–SimAIT. Its three modules and the interaction of each module are shown in Fig.1.In the scenario data parsing module, the scenario files are parsed and stored in a proper structure. This approach avoids scanning scenario files for each simulation object initialization. The adapter interface function library moduleprovides interfaces for automatic generation of each simulation object initialization code. According tothe nameofsimulation object classand other information,these interface functions can automatically search the data storage to establish the mappingrelationship between scenario data and properties name. And then, the simulation object matches its value. In the automatic code generation module, its function is to generate code statements for assigning values to simulation object variables.

3.1Scenario Data Parsing Module

The traditional method to parse the XML files is to use a tree structure to store the all file into system memory [15,16]. However, as the scale of simulation grows larger, the scenario files become more complex, and XML data become larger. So using a tree structure to store the scenario files will increase the complexity of the algorithm space and cause excessive memory consumption and low query efficiency. In computing, a hash map (hash table)structure uses a hash function to compute an index into an array of buckets or slots, from which the desired value can be quickly found.

In military simulation systems, the scenario is correspondingly organized under the military hierarchical structure, which is heterogeneous with the structure of system initialization requirement. So that initializing systems from the scenario needs massive artificial participation. In order to search the desired value quickly in data storage, we proposed a Nested HashMap Storage (NHMS) method according to the structure of a simulation application system. The first layer is SimObjclass, and in this layer of HashMap, the key is the name of SimObjclass, the value is the second layer of HashMap. In the second layer of SimObjinstance, the key is the id or name of SimObj instance and the value is the third layer of HashMap. In the third layer of SimObj parameter, the key is the name of SimObj parameter and the value is the value of the parameter that we desired. The structure of NHMS method is shown in Fig.2. With the NHMS method, the scenario data information is reorganized. The new structure of scenario data is matched to the structure of a simulation application system, which makes it easy to search the desired initial variable and improves the search efficiency.And this module usestingXML [15] to parse this scenario file and storesthe parsed data intoNHMS structure.

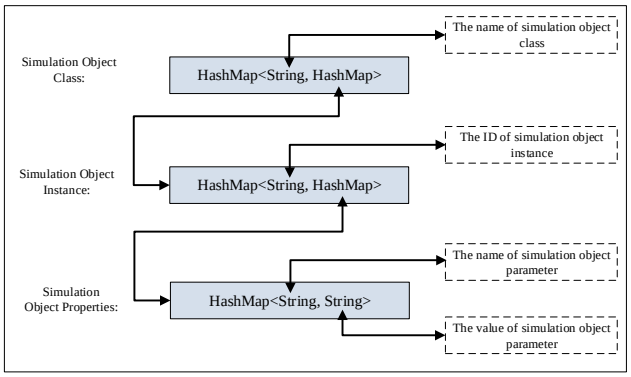

Figure 2:The structure of NHMS method.

3.2Assignment Function Library based on Adapter Mode

In order to extract the value of simulation object parameter in the NHMS, we design assignment function library based on adapter mode. The basic form of interface assignment function in the library is as follows:

*Get/Type/Value(string simobjName, intinstanceNumber, string parameterName)*or

Get/Type/Value(string simobjName, string instanceName, string parameterName)

The type of returned value is Type, such as intGetIntValue() , double GetDoubleValue() etc.simobjName is the name of simulation object class, instanceNumber is the id and instanceName is the name of simulation object instance, parameterName is the name of simulation object parameter. This function can automatically search the value of the parameter from the scenario data storage according to the name of simulation object class and the id of simulation object instance and the name of simulation object parameter. Based on adapter mode, the interface functions establish the mapping relationship between scenario data and properties name.

An example of calling a function from assignment function library to initialize variables for simulation object is shown as Fig.3. RadarInitData is the data structure of model radar, and RadarName is the variable of data structure. Because the type of this variable is string, we use GetStringValue() function for assignment.

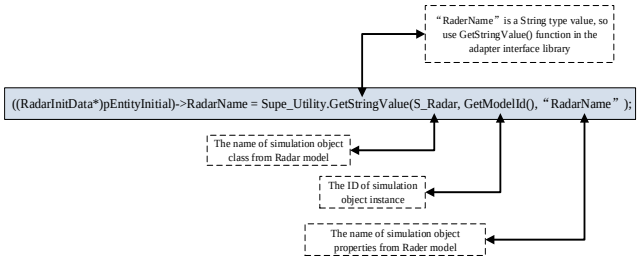

Figure3: An example of calling a function to initialize variables.

3.3 Automatic code generation (ACG)

At present, when the data of scenario files are modified, the engineers need to manually modify the code and recompile. This handwork handling may lead to low efficiency, error and heavy workload problem. So we propose the automatic code generation module. When the inputs of scenario files are modified, it just needs to run the program again and the generated code is changed. The function of the automatic code generation module is to generate simulation object initialization variable assignment code statement. This code gives the initial value of the state variable of the simulation object by calling the corresponding interface of the interface function library. And the corresponding value in the cache structure is found with the name of property. At last, the initial value of simulation object property is assigned.

```
void S_Ship::Init()
{
    int num = Supe_Utility.GetObjNumber(S_Ship);  //Get the number of SimObj Instance
    for(int i = 0; i < num; i++){                 //Initialize the properties by assignment function
      ((ShipInitData*)pEntityInitial)->longtitude = Supe_Utility.GetDoubleValue(S_Ship, Supe_Utility.GetId(i), "longtitude" );
      ((ShipInitData*)pEntityInitial)->latitude = Supe_Utility.GetIntValue(S_Ship, Supe_Utility.GetId(i), "latitude " );
      ((ShipInitData*)pEntityInitial)->altitude = Supe_Utility.GetIntValue(S_Ship, Supe_Utility.GetId(i), "altitude " );
      ...
    }
}
```

Figure 4: An example of automatically generated initialization codes.

In the generating procedure, we use the Velocity template engine based on Java to generate the initializationcodes with the scenario storage data and defined template files. Velocity-based code generation technology is a commonly used method now. Velocity engine put data into context objects through Java program and using velocity template language (VTL) in template files to call the data in Context, and then the required output is generated with template files [17]. An example of automatically generated initialization codes are shown as Fig.4.

As a whole, the generating procedurecan be summarized as Algorithm 1.First, the scenario file is parsed and the scenario data are stored into NHMS. For each class of SimObj, the SimObj instance layer HashMap is obtained through the name of SimObj Class name from NHMS. And for each instance of SimObj, the SimObj property layer HashMap is obtained. For each property of SimObj, the initialization codes are generated, and these codes call the interface function in library to assign the value until all the variables to be initialized.

Algorithm 1： Generating Procedure of Automatic Initialization

Input: *scenario file, model description file, template file*
Output:*Init() function of eachSimObj*
parse *scenario file and* store *the data into* NHMS
calculate *the number of SimObj Class:*N_C
for i=0 to N_C **do**
calculate *the number ofSimObj Instance*:N_I
getHashMap*of Instance layer*
for i=0 to N_I **do**
parse *model description file*
calculate *the number of SimObj Property:*N_P
get HashMap*ofProperty layer*
for i=0 to N_P **do**
generateInit()*code statement with template file*
call *the interface function in library to assign the value*
end for
end for
end for

4 ACASE STUDY

This section takes a simple scenario file as an example for input to verify SimAITautomatic initialization function and its effectiveness. The generated codesare used to initialize the complex simulation system based on YHSUPE(YH Simulation Utilities for Parallel Environment) [17].YHSUPE is a parallel discrete-event simulation environmentaiming at the simulation of large-scale system.First, we verify the function of automatic initialization.The input scenario file is shown in Fig.5. Due to space limitations only a portion of the files is shown here. All data of simulation system that needs to initialize the variables can be found in the file.With the input files, the code generator then generates the corresponding initialization program automatically.

```
<ScenarioDescription>
      <ScenarioSummaryfileName="CaseStudy"createTime="2017-01-20-09-54-
26"version="Ver1.0"instruction="this is a case study"verification="No pass"/>
      <ScenarioTimestartTime=" 2017-01-20-09-54-26
"endTime="2400.000000"aheadTime="1200.000000"/>
      <SimObjSet>
          <SimObjistanceID="35"className="BlueContinent"
country="2"longitude="10.011289"latitude="3.283614"altitude="0.000000"modelGUID=
"10001006"/>
          <SimObjistanceID="24"className="EarlyWaringRadar"country="1"longitude
="108.011289"latitude="31.283614"altitude="0.000000"modelGUID="10000064">
            <AttributeSet>
<Attribute Name="RadarName" Value="6027" Type="std::string "instruction=""/>
              <Attribute Name="AntennaCount" Value="1" Type="std::int
"instruction=""/>
            <Attribute Name="RadarMinAzimuth " Value="30" Type="std::double
"instruction=""/>
<Attribute Name="RadarMaxAzimuth" Value="150" Type="std:: double
"instruction=""/>
<Attribute Name="RadarElevation" Value="0 " Type="std:: double "instruction=""/>
              <Attribute Name="RadarAzimuth " Value="90 " Type="std:: double
"instruction=""/>
              <Attribute Name="PowerOnOffID " Value="1 " Type="std:: double
"instruction=""/>
              <AttributeName="FlighRoute " Type="std:: Position "instruction="">
                <AttributeDetailName="RadarX" Value="91.150670"
Type="std::double "instruction=""/>
                <AttributeDetailName="RadarY" Value="14.581067"
Type="std::double "instruction=""/>
                <AttributeDetailName="RadarZ" Value="10.000000"
Type="std::double "instruction=""/>
              </Attribute>
</AttributeSet>
          </SimObj>
        <SimObjistanceID="40"className="FlyMissile"country="2"longitude="-
103.060394"latitude="38.599220"altitude="0.000000"modelGUID="10001006">
            <AttributeSet>
<AttributeName="TargetID" Value="11" Type="std::int "instruction=""/>
              <Attribute Name="TargetType" Value="1051" Type="std::int
"instruction=""/>
<Attribute Name="IdentifyFlag" Value="5" Type="std::int "instruction=""/>
<Attribute Name="MissileX0" Value="101.245167" Type="std:: double "instruction=""/>
<Attribute Name="MissileY0" Value="21.379652 " Type="std:: double
"instruction=""/>
                <Attribute Name="Altitude0" Value="0.000000
```

Figure 5:The sample of input scenario file.

The automaticcode generation project is shown in Fig.6.The *template.vm* is template file for Velocity template engine andthe *Radarcode.cpp*and *Missilecode.cpp*aregenerated code files. And in the cpp files, the initialization sentences for these variables are generated. As shown in Fig.6, the initialization of a variable is actually an invocation to a certain initialization function in the library. With the generated files and function library and scenario data storage, we construct the executable automatic initialization project.In order to verify that the initialization codes can be initialized correctly, we run the initialization codes in the simulation system YHSUPE. The result of initialization is shown in Fig.7. By comparing with the scenario data in Fig.5, it is found that each variable of simulation object is initialized correctly.

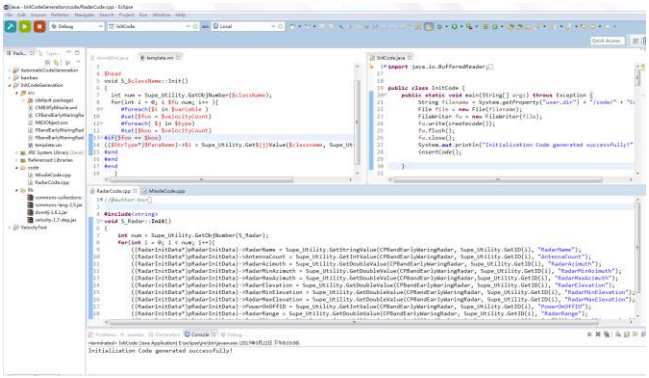

Figure 6:Initialization projectof automaticcode generationmodule

```
Problems  Tasks  Console  Properties  Call Graph
<terminated> 20170313 [C/C++ Application] /root/Baosirui/workspace/201
Loading Scenario...
Start initializing...
Initialization Done!

class name : BlueContinent
instance ID: 35
class name : EarlyWaringRadar
instance ID: 24
name: AntennaCount value: 1
name: PowerOnOffID value: 1
name: RadarAzimuth value: 90
name: RadarElevation value: 0
name: RadarMaxAzimuth value: 150
name: RadarMinAzimuth value: 30
name: RadarName value: 6027
name: RadarX value: 91.150670
name: RadarY value: 14.581067
name: RadarZ value: 10.000000
class name : FlyMissile
instance ID: 40
name: AttackID value: 167
name: IdentifyFlag value: 5
name: MissileX0 value: 101.245167
name: MissileX1 value: 107.244167
name: MissileY0 value: 21.379652
name: MissileY1 value: 31.379742
name: MissileZ0 value: 0.000000
name: MissileZ1 value: 0.000000
name: TargetID value: 11
```

Figure 7:The result of automatic initialization.

In order to verify the effectiveness of SimAIT, we designed three simulation systems. And the system parameters are shown in Table 1. Compared with more than one hour cost throughwriting setup code manually, it takes less than 15 minutes to finish initialization using SimAITshown in Fig.8. It is only needed to click the mouse and the initialization codes will be generated automatically in SimAIT, thus significantly reducing development time.

The case study shows that the automatic initialization tool SimAIT can generate the initialization program correctly and automatically with the input files, and that the constructed initialization program under the architecture can also initialize the system automatically. From the initialization program automatic generation to the automatic data fetching, data parsing and Variable assignment, the only manual work just lies in the construction of the initialization project for a simulation system. Compared with the initialization done manually, the efficiency improvement of the automatic initialization tool is obvious.

Table 1:Parameterinformation of each simulation system

Simulation system	Num of SimObj entities	Num of SimObj parameters	Timeconsumption(min)	
			manual	Automatic
System 1	10	30	32.5	5.2(84.00%)
System 2	20	60	67.6	10.8(84.02%)
System 3	30	90	98.2	13.5(86.25%)

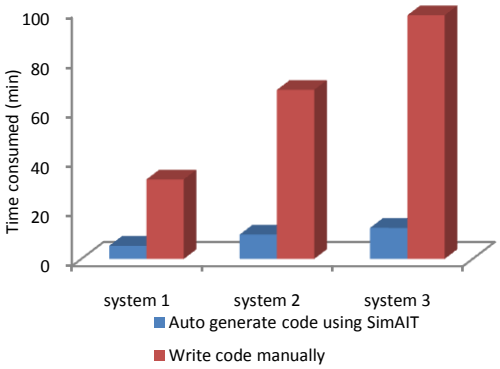

Figure 8:Comparison of time consumption with SimAIT and manual initialization

5 CONCLUSIONS

This paper proposed an automatic initialization tool–SimAIT for complex combat simulation system. In order to realize the function for automatic initialization, we designed three modules. First, the scenario files were parsed and stored in NHMS structure for easier access of scenario data in scenario data parsing module. Also, we designed the function library in adapter interface function library module for initialization sentences to assign data in storage to variables. And in the automatically generated code modules, the initialization sentences for these variables are generated. The case study shows thatSimAIT can initialize the simulation system automatically and efficiently. Automation reduces the heavy manual-work dependence and the artificial error rate.And the time consumption of simulation system initializationis reduced obvious compared to writing setup code

manually.As for future work, the standard scenario format and efficient Data Preparation are worth studying.

REFERENCES

[1] Bo-Hu, L. I., Chai, X. D., Tan, L. I., Hou, B. C., Lin, T. Y., & Xing, C., et al. 2012. *Research on high-efficiency simulation technology of complex system*. Journal of China Academy of Electronics & Information Technology.

[2] Parsons, M. D., Ltc, J., &Surdu. 2006. *The U.S. army's next generation simulation modelling the response to the world's future threat.* Chest, 118(5), 1412-1418.

[3] Carlton, B., Scrudder, R., Black, C., & Hopkins, M. 2004. *Initialization of C4I Systems and Simulation Federations Today and in the Future.* In Paper 04FSIW-089, Fall Simulation Interoperability Workshop.

[4] Lanman, J. T., & Proctor, M. D. 2009. *Governance of data initialization for service oriented architecture-based military simulation and command and control federations.* Journal of Defense Modeling & Simulation, 6(6), 5-16.

[5] Li, J., Yao, Y., Tang, W., &Zhu, F. 2015. *Research on the development approach for reusable model in parallel discrete event simulation.*Discrete Dynamics in Nature & Society, 2015, 1-11.

[6] Zhu, F., Yao, Y.,Tang, W.,& Tang, J. 2017. *A hierarchical compositeframework of parallel discrete event simulation for modelling complex adaptive systems.*Simulation Modelling Practice and Theory , 77, 141–156.

[7] Blais, C. 2012. *Strategies for application of the coalition battle management language (c-bml) with the military scenario definition language (msdl).* Approved for public release; distribution is unlimited..

[8] Hsu, I., Yuan, K. T., Cheng, Y. J., & Huang, D. C. 2011. *Semantic-based military scenario generation for mission planning.* Journal of Convergence Information Technology, 6(4), 123-134.

[9] Cai, Y., Peng, Y., Huang, K., & Huang, Q. 2012. *Msdl based lifecycle support of cloud simulation.* Procedia Engineering, 29, 1281-1285.

[10] Sprinkle, R., Black, C., 2006. "*Joint BC & Simulation Systems Initialization Process.*" Fall 2006 SIW.

[11] Yan, L., & Chong, S. 2012. *A simulation scenario matching study based on the MDA model transformation.* International Conference on Fuzzy Systems and Knowledge Discovery (pp.2866-2869). IEEE.

[12] Yao, S., Hu, P., &Guo, X.2012. *XML-based Exchange Methods of Heterogeneous Simulation Scenario Data.* International Conference on Industrial Control and Electronics Engineering (pp.885-888).

[13] Molzahn, D. K., &Lesieutre, B. C.2013. *Initializing dynamic power system simulations using eigenvalue formulations of the induction machine and power flow models.* Circuits & Systems I Regular Papers IEEE Transactions on, 60(60), 690-702.

[14] Menezes, N. N. C. 2014.*Parallel Initialization on Distributed Simulation Systems.* In Proceedings of the 2014 IEEE/ACM 18th International Symposium on Distributed Simulation and Real Time Applications (pp. 18-24). IEEE Computer Society.

[15] Chong Z.2014.*XML data parsing method based on Tingxml.* Science and technology economic market, 2014(8) : 76-77

[16] Bitonti, T., Price, D. L., &Bouknight, W. J. 2016. *Adaptive parsing of sparse xml data.*Patent No. 9,396,171. Washington, DC: U.S. Patent and Trademark Office.

[17] Surhone, L. M., Tennoe, M. T., Henssonow, S. F., Foundation, A. S., Platform, J., & Edition, M. 2010.*Apache Velocity*. Betascript Publishing.

[18] Y.-P. Yao and Y.-X. Zhang.2008. *Solution for analytic simulationbased on parallel processing.* Journal of System Simulation, 20(24), 6617–6621,.