THE MODULAR SIMULATION LANGUAGE (MODSIM) -A POWERFUL TOOL FOR COMPUTER SIMULATION

Juliana KARAKANEVA

Introduction

One of the important directions in enhancing the processes of planning and management of defense and armed forces is the use of contemporary methods to support decision making. The application of these methods requires analysis of functional processes, technology, methods and means for their realization. The development of models, the research and assessment aimed at optimizing the process on different levels are achieved on the basis of comprehensive and in-depth analysis.

We develop a concept for using modeling and simulation in studying problems of command and control systems in conditions maximally close to the expected reality. So, the considered approach is applied to the warfare control system and in particular to the analysis of different aspects of these systems. In many cases the purpose of such models is to explore and analyze various sides of the real systems or subsystems, or on a lower level - of their elements.

In this article the discussion object is one aspect of modeling - the use of specialized software for simulation experiments with discrete models and, in particular, with models including random processes. These are systems in which random processes are evolving, i.e. random events changing the state of the system are appearing. Approaches and methods exist that allow to present and process complex sequence of events, such that may develop accidentally and asynchronously.

Computer simulation

Specialized programming languages are created for simulation experiments.[1,2] These languages include opportunities for representation of models (objects, attributes,

actions), algorithms for events control, mechanisms of objects interoperability, servicing libraries of subroutines for statistics and for preparation of final results, etc.

There are two general categories of computer simulation: continuous simulation and discrete-event simulation. A continuous simulation describes a system using sets of equations which are solved numerically with respect to time. The continuous simulation program then steps forward by the increment of time chosen for the time step and recalculates all equations which describe the model.

In this case the attention is directed towards the discrete-events-based computer simulation which describes a system through logical relationships between its elements and where changes of state are caused in discrete points in time.

In a discrete-event simulation varying amounts of time elapse between events, but the state of system changes when one of its component parts changes state. The Modular Simulation Language MODSIM [3] takes the capabilities of discrete systems modeling languages like Simula and SIMSCRIPT II.5 and adds object-oriented programming capability and the modular construct of Modula-2. These properties make it an advanced tool for simulation, as well as for warfare models development.

The classical approach to discrete-event simulation involves processing of event sequences. For instance, in a simple "fire engagement" model the event routines might be:

- *Aircraft takes off*
- *Aircraft enters queue*
- *Aircraft attacks*
- *Air-defense detects the target*
- *Air-defense fires on the target*
- *Target is engaged (or target leaves), etc.*

The simple model consists of two simulations - simulation A1, representing an air force object (*AirForceObj*) and simulation A2, representing an air-defense object (*AirDefObj*)**.** The routines describe discrete events and passage of time is handled by scheduling the next event for the object currently being manipulated.

This approach is adequate for smaller models, but in larger models it is difficult to follow or modify the logical scheme which describes the behavior of an object. In this case the process approach simplifies the larger model design by allowing many aspects of an object's behavior in a model to be described in one method. It creates classes of objects and each class describes the corresponding functional object's behavior and the simulation program can generate multiple, concurrent instances of this object. For example, the simulation program would create a new instance

AirForceObj each time an aircraft takes off. While there would be multiple, distinct copies of the object operating simultaneously, each one could have different values of its fields to describe particular properties.

The considered process approach is supported in MODSIM. It combines object-oriented features and discrete-event simulation for development and maintenance of large models. A MODSIM simulation model defines a system through processes because this technique provides a powerful structure for describing multiple simulation problems, and provides many advantages over the direct use of discrete events.

Project creation

MODSIM program is created as a project that includes several types of modules. A mechanism is proposed for creation, modification and work with the project by sequence of options.

MODSIM supports a possibility for importing definitions and declarations between different modules. There are two major types of modules in MODSIM: *Main* modules and *Library* modules, and they are compiled separately to ease the task of development and maintenance. There are two parts of a Library Module - *definition* module, which contains declarations of types, constants, variables, procedures and methods, and which can be imported to other modules, and *implementation* module containing the implementation code of procedures and methods for objects, but which does not have to be visible outside the library module itself.

Data structures. Objects

The data is organized in object structures, manipulated and managed as compact information units. The object type contains a list of definition fields, describing the object properties. For instance, the information about an air force object (*AirForceObj)* includes:

Name (type)
Time between two takeoffs (random distribution)
Speed (AirSpeed)
Flight height (H)
Flight corridor (X1, Y1), (X2, Y2), etc.,

and respectively for an air-defense object (*AirDefObj*):

Name (type)
Number of Ammunitions

Fire range (R, r = f(H))

Number of Ammunitions in a shot (ShotForHit)

Time between shots (TimeForHit)

Probability for target detection (ProbForDet)

Probability for target engagement in a single shot (ProbForHit), etc.

During the experiment a possibility is created for interactive mode change of values in the objects characteristics realized through a service function in a special menu-module.

The modular structure of MODSIM program helps for flexible approach to the data base design. The developer may make changes to the methods realization and add new object types - inheritors (for example- helicopter, balloon, air-landing, etc.) to data structure and later will be able to recompile only those modules which have been changed. Through the language technology the data base is created as an open structure, where the new object types are added without problems.

Processes and events management

Besides possessing object-oriented programming features MODSIM is capable of building process-based discrete-event simulation models.

Time management

The use of time management is a key element of interoperability between processes and objects in simulations. MODSIM provides powerful and flexible tools for managing objects' behavior as model elements. A MODSIM object is capable of carrying on multiple, concurrent activities each of which elapses simulation time. An *activity* is what occurs in the model as time elapses. An *event* is a point in time at which the state of the model changes.

An object has methods associated with it to describe actions it can perform. ASK method is similar to the conventional procedure call, but when the ASK statement is executed, a message is sent to the object requesting it to invoke the method for an action. ASK methods are not allowed to pass any simulation time. For instance:

ASK Squad TO Add(Enemy),

ASK Enemy TO SetSpeed(EnemySpeed), etc.

Unique MODSIM capabilities for models realization and management are the TELL methods, which provide elapse of simulation time. The TELL call starts the simulation under the MODSIM simulation engine control. During an *activity*, time is elapsing and no MODSIM code is executed to cause this happen. During an *event*,

MODSIM code is executing to change the model's state, but no simulation time is elapsing. So,

TELL Enemy TO GoTo(X,Y

TELL (ASK Tank Gun) TO Blink(CX,CY)

are statements, which elapse simulation time using a WAIT statement.

Interaction

One MODSIM object can have multiple TELL methods carrying on activities simultaneously with respect to simulation time. A method can perform a sequence of related concurrent actions. These actions may be punctuated by intervals during which simulation time elapses, i.e. they perform a WAIT. When a WAIT statement is encountered, MODSIM engine saves the state of the time-elapsing method, then suspends its execution until the WAIT is completed or interrupted. MODSIM then resumes execution of the time-elapsing method at the appropriate simulation time. During the WAIT, other activities may take place.

Several distinct object instances can be carrying out activities at the same point of simulation time. One object instance can have two different TELL methods carrying out activities at the same point of simulation time. One particular TELL method of an object instance can be invoked multiple times so that distinct instances of that method are each carrying out activities at the same point of simulation time.

Spatial management

MODSIM is provided with an object-oriented graphic system SIMGRAPHICS. The features distinguishing SIMGRAPHICS from other graphic system are its portability and its integration with MODSIM simulation engine. SIMGRAPHICS employs vector graphics to draw icons, charts and backgrounds, and allows simple approach to the use of interface, graphic and animation features of window systems such as Microsoft Windows and X Windows. The system supports the following functions:

- Data visualization, charting, plotting
- Animation and interactive graphics
- User interface through dialog boxes, menus, etc
- Editing graphical and user interface objects
- Recording and playback of animated graphics
- Three dimensional animation

SIMGRAPHICS objects (icons, graphics, forms, images) can be created through graphic editor SIMDRAW, integrated in MODSIM environment or through a MODSIM program and can be manipulated in the same way.

Experimental model

A simple computer simulation program models the "fire engagement" operation in conditions close to the reality. The goal for the model is to get test results about fire engagement effectiveness for various object types - military objects. The rules governing the objects' behavior and the logical objects' interaction are presented. Objects involved in the simulation are:

AirForceGenerator	*AirForceObj*	*Explosion Obj*
AirDefenceGenerator	*AirDefenceObj*	*TypeObj*
GunGenerator	*GunObj*	
MenuGenerator	*MenuObj*	

The following steps are realized by the MODSIM project:

- Updating objects' parameters through menu-options (*MenuGenerator*) and air-force and air-defense objects initialization
- Presetting the experiment conditions - battlespace, terrain, air-force units flight corridors, air-defense units positions, etc.
- Generation and graphic presentation of objects and terrain
- Starting the simulation experiment

The scenario includes:

- Target detection in detection range with probability *ProbForDisc*
- One shot fire engagement in fire range with probability *ProbForHit*
- Attack and destroy target, while given conditions are available

The main conditions to consider are:

- Targets' speed and height
- Air-defense detection range
- Air-defense fire range
- Available ammunitions
- Terrain

The fire engagement effectiveness is calculated as

$$E = \text{Number of destroyed targets} \ / \ \text{Number of arrived targets.}$$

For a more precise model development other conditions should be taken in mind.

Conclusion

MODSIM possesses all the capabilities of object-oriented programming languages for the development of an information base and object classes, a mechanism for process and events control, tools for experimental data statistics and graphic user interface. Input/Output, simulation, animated graphics are available through standard modules. A unique property of the language is the *powerful and flexible engine for building process-based discrete-event simulation models.* The object-oriented graphic system SIMGRAPHICS contains means for data visualization, user interface, three dimensional graphics and animation.

Beyond the scope of this study remains the object-oriented framework for developing commercial quality discrete-event simulation applications SIMOBJECT (based on MODSIM). COMNET III (Communications Network Planning Tool) and SIMPROCESS III (Business Process Reengineering Tool) are complete applications, which are built on the SIMOBJECT foundation.

MODSIM with SIMGRAPHICS and SIMOBJECT is a powerful software tool, which enhances the ability to rapidly prototype real-world system. By the means of the program system there can also be created mathematical models based on Game Theory, Applied Queuing Theory, Graphs Theory, Morphological Analysis, Linear Programming, etc. The problems of planning, activity networks, project management, analysis and management decisions are solved.

The MODSIM-based models are mobile structures, which can be used together in different simulations, on different levels of compatibility. The programming language provides real-world process simulations building, in the military field, respectively, aimed at obtaining experimental and statistical data which would support the development of command and control systems, as well as the development of systems with education and training purposes.

References

1. T.J. Schriber, *Simulation Using GPSS* (New York: Wiley, 1974).

2. *MODSIM II. The Language for Object-Oriented Programming* (CACI Product Company, 1994).
3. Slavcho I. Slavchev, Juliana Karakaneva, V.L. Radeva, "Implementing the Programming Language MODSIM II for Development of Imitation Models," In *Proceedings of the 1997 Conference of the Bulgarian Army Academy* (Veliko Tarnovo, 1997).

JULIANA KARAKANEVA is born in 1953. She holds a M.Sc. degree in Automatics and Telemechanics from the Technical University of Sofia (1976), Ph.D. degree from the Technical University of Sofia (1983) with dissertation on "Methods for Logical Design of Fail-Safe Automatic Devices for Railway Transport." Research fellow at the Military Research Development Institute of General Staff of Bulgarian Armed Forces, currently - Institute for Advanced Defense Research. Main research interests and activities: design of command and control systems, computer assisted exercises, combat modeling. *E- mail*: gpavlov@md.government.bg.