SIMSCRIPT: A Programming Language for Military Simulation

Xudong Fang

1 National Defence University, Beijing, 100091, China
*Corresponding author’s e-mail: fangxudong@nudt.edu.cn

Abstract. SIMSCRIPT is high-level simulation language with a long history and world-wide acceptance, especially in the field of military simulation. The paper first introduces the SIMSCRIPT world-view and its simulation mechanism, which lay the basis for its success, and next points out the language’s prominent advantages from the viewpoint of a military modeler, and also lists its weaknesses which are yet to be improved, and then proposes suggestions to better utilize the capability of the simulation language, and last discusses the development trend of SIMSCRIPT.

1. Introduction

Computer simulation for analysis and planning has proved to be valuable in many fields such as manufacturing, logistics, economics, transportation, and military operations. However, one well known obstacle has been the amount of time and energy needed to program simulation of even moderate complexity, which suggests that a wise choice of simulation language really matters.

SIMSCRIPT was developed in 1962 by Nobel Prize Laureate Harry Markowitz and his team to support an Air Force RAND project in efficient preparation of simulation models[1]. The language’s outstanding ability in combat model description enables it to simulate combat at Theatre, Corps, Division, and Battalion levels, which makes it possible its world-wide acceptance and applications in military command & staff training, operation plan analysis, decision supporting and so on. In fact, SIMSCRIPT has provided solutions since it was designed for many world-famous military simulation programs, such as Joint Theatre Level Simulation (JTLS), Tactical Air-Land Operations Model (COMMANDER), Corps Battle Simulation (CBS)[2].

Among those solutions and applications, Joint Theatre Level Simulation (JTLS) is the most well-known and perhaps the best example demonstrating SIMSCRIPT’s modelling capacity. JTLS is a multi-sided, interactive wargaming system that models the operations of army, air force, navy, and special forces, and allows users to gain insight into the comparative advantages of alternative operation plan[3]. The authors of JTLS admitted that the key to the success of JTLS is the SIMSCRIPT language, and professed that “the only reason JTLS was up and running long before other models that started around the same time was because of the SIMSCRIPT language[4].”

Therefore, we should ask the question why SIMSCRIPT makes these things possible. In the following sections, we will introduce the philosophy of SIMSCRIPT simulation, and present the pros and cons of the language, and give our suggestions on building military models using the language, and show the development trend of SIMSCRIPT. In the following discussions, when referring to SIMSCRIPT, unless specifically pointed out, we mean SIMSCRIPT II.5, rather than other versions,
since SIMSCRIPT II.5 is the most typical version of SIMSCRIPT and many successful military simulation systems have been developed using SIMSCRIPT II.5.

2. SIMSCRIPT philosophy
According to Harry Markowitz, the inventor of SIMSCRIPT, the most important aspects of SIMSCRIPT as a competent simulation language are its world-view of model to be simulated and its method of communicating to the computer the world to be simulated[5].

![SIMSCRIPT's world-view](image)

Figure 1. SIMSCRIPT’s world-view.

2.1. SIMSCRIPT’s world-view
The standpoint from which SIMSCRIPT views the simulated world is status and event, processes and resources[6]. Status of a simulated system consists of entities, attributes, and sets. More specifically, SIMSCRIPT characterizes a system status at a specified time point in terms of entities of various types with various attributes, set ownerships and set memberships. Entities can be either permanent or temporary.

An event is an instantaneous occurrence, which may change the state of a system, as shown in figure 1. Every event has an event routine associated with it, which updates the state of the system when a particular type of event occurs. During the simulation process, the status of the system is changed by an arriving outside event or by an inside event which is caused by prior events. SIMSCRIPT organizes all the events in the future event list, and provides the time routine that schedules events in the list in the manner that guarantees that the most imminent event occurs next. Simulated time is advanced from event to event rather than at fixed intervals of time.

In SIMSCRIPT, an activity within a system is bounded by two instantaneous events: when the activity starts, and when it stops. Therefore, if the events that delimit an activity can be identified, the activity can be modelled by these events. Furthermore, such a collection of related events may be represented by a process. In other words, a process may be viewed as a collection or sequence of related events separated in time. A resource is a kind of permanent entity that provides a means of modelling competition among processes for objects which are in short supply. The relationship among events, activities and processes described above is reflected in figure 1.

2.2. SIMSCRIPT simulation mechanism
Although originally produced for simulation, SIMSCRIPT is designed to have five different abstract levels and be programmed accordingly[7].

- Level 1-3 features general-purpose programming language statements.
  SIMSCRIPT uses Level 1 to Level 3 features to handle basic programming elements such as variable assignment, subroutine definition and computation flow control.

- Level 4 supports the entity-attribute-set features.
  SIMSCRIPT uses Level 4 features to deal with entities, attributes and sets definitions, and to define event notices and event routines to describe the effects various types of events have on the system.

- Level 5 focuses on the simulation-oriented part of SIMSCRIPT II.5.
SIMSCRIPT uses Level 5 features to realize time advance, event-processing, and accumulation and analysis of simulation-generated data. Modelers can choose different simulation policies, realized by SIMSCRIPT as Event Scheduling and Process Interaction. It is worth noting that SIMSCRIPT does not support the Activity Scanning simulation policy.

3. Advantages of SIMSCRIPT
As an old but vibrant programming language, there is no doubt at all the SIMSCRIPT features many advantages in military simulation. Here we just point out three prominent advantages from the viewpoint of a military modeler.

3.1. Simple and clear simulation concepts
Follow the SIMSCRIPT’s world-view described in the introduction, when building combat models, modelers first need to identify different types of entities, define their attributes, and set ownerships, and then find the events that change the system status. Since SIMSCRIPT is designed for a discrete event simulation language, events that change the system status are not difficult to identify. It is the event routines, which incorporate the algorithms describing the intricate relationship between entities, that are most energy-consuming. SIMSCRIPT’s simple and clear simulation concepts let modelers quickly focus their time and energy on the most important part of the simulation task. Moreover, SIMSCRIPT supports multiple simulation policies, including Event Scheduling, Activity Scanning, and Process Interaction. Users can choose the suitable simulation policy according to the nature of the simulated system.

3.2. Easy-to-read/write programming style
SIMSCRIPT is an English-like language that allows for modular implementation. This feature has a great significance in military simulation, since most of the time military specialists are not familiar with programming. They always feel repugnant to read arcane programs written in C or Fortran. Simulation programs written in SIMSCRIPT can be understood by those who may not be familiar with programming. Moreover, SIMSCRIPT has many inherent programming statements greatly improving coding efficiency[6]. For example, SIMSCRIPT supports many search statements over sets, such as the FIND THE FIRST CASE statement combined with the IF FOUND statement, the meaning of which are straightforward. However, it takes more code lines if the logic is implemented in a generally purpose programming language.

3.3. Nice self-documenting capacity
Last but not least, SIMSCRIPT programs feature nice self-documenting capacity. It is well known that good documentation is of vital importance to the maintenance and sustainable development of a software system. SIMSCRIPT’s English-like programming makes the source program the best documentation for itself, thus perfectly avoiding the documentation pitfall which plagues even the once well-documented system, since the system is always subjected to increased understanding of the system, changing goals, availability of data, which make the program comments obsolete or incorrect[8].

Besides the above advantages, SIMSCRIPT also has other features, such as powerful statistical measurement and good portability. These advantages altogether render the analyst and programmer great convenience in the design and implementation of simulation models, making SIMSCRIPT one of the best simulation languages.

4. Disadvantages of SIMSCRIPT
Nothing is perfect. The SIMSCRIPT is fairly good for military simulation, but it still has some weaknesses. We list some of them gained from development experience. However, the judgment here is somewhat subjective because different users look at the issue from different angles. In a different requirement scenario, the shortcomings listed below may not be shortcomings at all.
4.1. Not supporting parallel execution
As far as we know from public channels, SIMSCRIPT programs are executed in serial mode. Discrete event simulation is rather difficult to run in parallel mode, due to the strict execution sequence of events according to their scheduling time. SIMSCRIPT is not equipped with synchronization primitives and the program is eventually run as a single thread process. With the ever-growing CPU frequency, this is not a problem for either combat simulation of moderate scale or for large scale simulation demanding no real-time performance. However, when a military simulation system has too many entities and events to process, or needs to run in the analysis mode for evaluating the relative merits of alternative operation plans, the serial execution mode exposes the problem. In some cases, such problem is fatal for the simulation, where the real-time response is of first concern.

4.2. Overly flexible programming style
SIMSCRIPT benefits a lot from its English-like, flexible programming style. Since the programming language is similar to a natural language, there are always more than one way to express one meaning. For instance, when using the CREATE statement to create an entity of the temporary entity class FLIGHT, one can use CREATE FLIGHT, or CREATE a/the/this FLIGHT, all of the statements serving the same effect with subtle nuances [6]. The use of articles is intended to add readability, but is rather confusing for a novice. Besides, permanent entities in SIMSCRIPT are accessed by numbers rather than by names. For example, if five entities of the permanent entity class HOME with attributes ADDRESS and AREA are created, we have to refer the attributes of each HOME as ADDRESS(1), ADDRESS(2), ..., ADDRESS(5), and AREA(1), AREA(2), ..., AREA(5), which is very strange at first sight and elusive for a C/C++ programmer.

5. Suggestions for SIMSCRIPT simulation
With the pros and cons of the language in mind, we give some suggestions for building military simulation models using SIMSCRIPT.

5.1. Prefer event to process approach
Simulation usually chooses one of two policies, Event Scheduling or Process Interaction. These two policies differ primarily in the way that they perceive the simulation world, and result in totally different simulation code version. Generally speaking, any system that can be modelled in the Event Scheduling manner can be modelled in the Process Interaction manner, and vice versa [4]. Unlike some languages, SIMSCRIPT permits either policy to be adopted. However, Event Scheduling is more advisable for those programs that need checkpoints. The ability to take checkpoints during a simulation is always important for large-scale and complex simulation systems that may require replays or After Action Reviews. However, the overhead of taking a checkpoint of the event approach is much higher than that of the process approach. For example, JTLS chooses the Event Scheduling approach rather than the Process Interaction approach, because the modelers found that the overhead of taking checkpoints of the process approach is unacceptable [9].

5.2. Prefer temporary to permanent entities
Permanent entities in SIMSCRIPT is elusive to both readers and programmer, since they are accessed by index numbers rather than by names, as demonstrated in Subsection 3.2. Besides, permanent entities are difficult to create or destroy. However, the role of a permanent entity can be replaced by a temporary entity through rearrangement of the program data structure. Temporary entities are referred by name, and easy to create or destroy, thus facilitating development and maintenance of simulation programs. Such replacement is taking place in the JTLS-GO system [9]. Therefore, we suggest the use of temporary entities as much as possible.
5.3. Be cautious of upgrades to object-oriented versions

SIMSCRIPT III supports object-oriented programming, and CACI Products Company claims that
SIMSCRIPT III is the only simulation package on the market which is specifically designed to support
modular object-oriented simulation development, and provides backward compatibility and seamless
transition of the existing SIMSCRIPT II.5 models to the new technological level[1]. It may be an
inevitable trend to move to SIMSCRIPT III to adopt object-oriented programming since SIMSCRIPT
is a commercial software product and CACI has been promoting the upgrades of SIMSCRIPT III. No
one wants to wake up one day to find out that its simulation program no longer works with new Linux
releases.

However, it may not be so easy for modellers to upgrade the existing simulation systems developed
in SIMSCRIPT II.5 to SIMSCRIPT III. According to ROLANDS & ASSOCIATES Corporation, the
author of the JTLS system, SIMSCRIPT III is not truly object-oriented, and some features workable in
SIMSCRIPT II.5 become deprecated in SIMSCRIPT III. R&A Corp’s solution is to adapt JTLS to
SIMSCRIPT III to take advantage of the new features, such as improved debugging capability, and to
remove deprecated SIMSCRIPT II.5 features, such as bit packing, equivalenced variables, and
permanent entities. However, R&A Corp has refused the object-oriented features of SIMSCRIPT III,
since it only supports the Process Interaction modelling policy which incurs unacceptable overhead for
JTLS when taking a checkpoint[9]. Therefore, think twice before you leap, and upgrade existing systems
to object-oriented versions with due caution.

6. Development trend of SIMSCRIPT

There have been five versions of SIMSCRIPT which can be casted into three categories, namely,
SIMSCRIPT I&I.5, SIMSCRIPT II&II.5, and SIMSCRIPT III, as shown in table 1. Starting with text-
based pre-processors for FORTRAN in SIMSCRIPT I&I.5, the language now supports modular object-
oriented simulation development interfaced with the compiler and the 2-D Java simulation graphics[1].
New features and functions have been added to the capacity of the language over the years. The stimulus
for the development of the language comes from users. For example, SIMSCRIPT III has improved its
debugging capability according to JTLS demand[9]. In light of the ever-growing requirement for fast
simulation speed, CACI may be working toward a more efficient compiler of SIMSCRIPT, and may
provide a parallel version of the language one day.

However, SIMSCRIPT is developing slowly compared to other general-purpose programming
languages, such as JAVA and PYTHON. It can be seen from table 1 that SIMSCRIPT II.5 had remained
as the only available version for more than forty years before SIMSCRIPT III came out in 2007. This is
mainly because SIMSCRIPT is a domain-specific language. Even in its own domain, it is mostly widely
used in military simulation rather than other fields.

| Version | Year | Platform | Compiler | IDE | Design Philosophy | Simulation Policy |
|---------|------|----------|----------|-----|-------------------|-------------------|
| I&I.5   | 1962 | N/A      | Text pre-processor | N/A | Process-oriented   | Event             |
| II&II.5 | 1966 | 32-bit Win/Unix | Independent | SimStudio | Process-oriented   | Event & Process   |
| III     | 2007 | 32/64-bit Win/Unix | Enhanced | SimEclipse | Object-oriented   | Event & Process   |

7. Conclusion

SIMSCRIPT has been improving ever since its debut in 1962 and has gained world-wide acceptance as
a successful simulation language. There are many reasons contributing to the success of the language,
especially in military simulation areas, among which the primary three factors are simple and clear
simulation concepts, easy-to-read/write programming style, and nice self-documenting capacity.
Nevertheless, SIMSCRIPT has its own weaknesses, such as not supporting parallel execution and overly
flexible programming style. Based on existing experience, several suggestions are proposed to better
utilize the capacity of SIMSCRIPT in military simulation, including preferring event to process
approach, preferring temporary to permanent entities, and cautious of upgrades to object-oriented versions. These suggestions are scenario-dependent choices rather than judgements of right or wrong. It is expected that SIMSCRIPT will be equipped with a more efficient compiler and support parallel simulation one day.

References
[1] CACI Products Co. (2020) History. http://SIMSCRIPT.net/partners/partners.html.
[2] CACI Products Co. (2020) SIMSCRIPT Solutions. http://SIMSCRIPT.net/solutions/military/military_operations.html.
[3] ROLANDS & ASSOCIATES Corporation. (2017) JTLS-GO Executive Overview. https://www.rolands.com/jtls/j_vdds/executive_overview.pdf.
[4] Roland R.J., Roland, E.F., Kelleher, E.P. (1988) Approaches and Aspects of Implementing A Computer Wargame Simulation A Historical Perspective. https://www.rolands.com/pp/Papers/treatise.pdf.
[5] Markowitz, H.M. (1966) Simulating with SIMSCRIPT. Management Science, 12: 396-405.
[6] CACI Products Co. (1997) SIMSCRIPT II.5 Programming Language. CACI Products Co., La Jolla.
[7] Abrahams, P., Kiviat, P. J., Villaneuva, R., and Markowitz, H.M. The SIMSCRIPT II Programming Language (110)
[8] Garrison, W.J. (1990) A Brief SIMSCRIPT II.5 Tutorial (Tutorial Session). In: Proceedings of the 22nd Conference on Winter Simulation (WSC’ 90). New Orleans. pp. 115–7.
[9] Roland, E.F. (2020) Introduction To JTLS-GO 6.0. https://www.rolands.com/jtls/j_iuc2019/ellen_60update.pptx.