한국시뮬레이션학회 논문지
Vol. 28, No. 1, pp. 41-47 (2019. 3)
http://doi.org/10.9709/JKSS.2019.28.1.041
ISSN 1225-5904

AddSIM 환경에서의 사용자 코드 동적 검증 방법론

양지용 · 최창범 †

Dynamic Verification Methodology of User Code in AddSIM Environment

Jiyong Yang · Changbeom Choi†

ABSTRACT

Defense simulation is actively used to test various weapon systems and evaluate their effectiveness. The AddSIM environment is a simulation framework designed to support the weapon systems dealt with in defense simulation from an integrated point of view and is designed for reuse and scalability. Models used in AddSIM require base model structure fidelity and verification of user code area. Therefore, this paper describes the dynamic verification method used for completeness of models used in AddSIM. For the dynamic verification of user code, the specification method and the verification algorithm are described. Also, we introduce the prototype of the dynamic verifier implemented based on verification specification method and algorithm. The case study analyzes the verification results based on the simulation example implemented in AddSIM environment.

Key words : Model Verification, Dynamic Verification Methodology, Simulation Environment

요 약

국방 시뮬레이션은 여러 무기체계를 실험하고 그 효용성을 평가하기 위해 활발히 사용된다. AddSIM 환경의 경우 국방 시뮬레이션에서 다루는 무기체계들을 보다 통합적인 관점에서 지원하기 위해 고안된 시뮬레이션 프레임워크로 재사용성과 확장성을 고려하여 설계되었다. AddSIM에서 사용되는 모델의 경우 기본 모델 구조에 대한 정보와 실제 무기 체계의 설계를 위해 사용되는 사용자 코드 영역에서의 검증이 통합적이며 정확한 시뮬레이션을 위해 필요하다. 따라서 본 논문에서는 AddSIM에서 사용되는 모델들에 대한 완전성을 위해 사용된 동적 검증 방법에 대해 설명한다. 사용자 코드 동적 검증 방법에 대해서는 명세 기법과 검증을 위한 알고리즘에 대해 설명한다. 또한, 검증 명세 기법 및 알고리즘을 바탕으로 구현된 동적 검증기 프로토타입에 대해 소개하며, 사례 연구에서는 AddSIM 환경에서 구현된 시뮬레이션 예제를 바탕으로 수행한 검증 결과를 분석한다.

주요어 : 모델 검증, 동적 검증 방법론, 시뮬레이션 환경

1. 서론

1.1 연구의 필요성

현대 컴퓨터 능력의 급속한 발전과 더불어 다양한 분야의 체계 개발에 있어서 모델링 및 시뮬레이션(Modeling & Simulation: M&S, 이후 M&S)의 필요성이 점차 증대되고 있다. M&S는 시뮬레이션 목적에 따라 특정 상황이 주어졌을 때 모의 대상 시스템을 통하여 도출되는 결과를 분석하는 분석 수단으로 활용될 수 있으며 실제 환경에서 운용하기 어려운 시스템을 가상으로 조정하는 가상 훈련과 같은 분야에서 활용될 수 있다. 특별히 국방 분야와 같이 다수의 체계가 사용되는 환경에서는 시뮬레이션 기술이 유용하게 사용될 수 있다. 이에 반하여 M&S를 활용하기 위하여 매번 새로운 시뮬레이션을 설계하는 과정은 큰 비용과 시간이 투입되어야 하는 문제가 있다. 이에 따라 국방과학연구소에서는 모델링 비용 및 시뮬레이

* 본 연구는 국방과학연구소 'BSM기반 검증 및 사용자 코드 검증 모듈(UD160075BD)'의 지원에 의해 수행된 연구결과의 일부입니다.

Received: 25 January 2019, **Revised**: 4 March 2019, **Accepted**: 14 March 2019

† **Corresponding Author**: Changbeom Choi
E-mail: cbchoi@handong.edu
Dept. of Global Entrepreneurship and ICT, Handong Global University, Pohang, Gyeongsangbuk-do, Korea

션 모델 개발 비용 절감을 위하여 재사용성과 확장성을 갖춘 국방 공통모의환경인 AddSIM을 제안하였다(Oh, 2014). AddSIM 환경은 국방 M&S를 수행하는 사용자가 쉽게 시뮬레이션을 구성할 수 있도록 시뮬레이션 실행과 관련된 코드를 자동으로 생성하고 도메인 분야에 있어서 확장이 필요한 부분을 사용자가 작성할 수 있도록 사용자 코드(User Code)를 구분하여 시뮬레이션 모델을 개발한다. 따라서 본래 시뮬레이션의 요구사항에 맞게 시뮬레이션 모델이 개발되었는지를 확인하기 위해서는 개발된 시뮬레이션 코드를 실행이 시뮬레이션 요구사항에 부합하는지 확인하는 것이 필요하다.

본 연구에서는 AddSIM 환경에서 사용자가 개발한 시뮬레이션 모델이 시뮬레이션 모델의 요구사항에 부합하는지 확인하는 방법으로 동적 검증 방법(Dynamic Verification)을 적용한 AddSIM 동적 검증기에 대하여 소개한다. AddSIM 동적 검증기는 AddSIM 환경의 특징을 활용하여 AddSIM을 활용한 시뮬레이션에서 이벤트를 추출하여 사용자가 개발한 시뮬레이션 모델의 구현이 요구사항에 부합하는지 확인한다.

본 논문의 구성은 다음과 같다. 2장에서는 이론적 배경에 대하여 소개하며 3장에서는 제안하는 AddSIM 동적 검증기에 대하여 제안한다. 4장에서는 AddSIM 동적 검증기를 활용한 AddSIM 시뮬레이션 검증사례를 소개하고 5장에서 결론을 제시한다.

2. 이론적 배경

2.1 AddSIM 환경

국방 시뮬레이션의 경우 소요제기 시에 다양한 수준에서 대상 시스템을 모의하기 때문에 다양한 수준에서 시뮬레이션이 설계될 수 있다는 문제가 있다. 이를 극복하기 위한 방안으로 제안된 AddSIM은 모듈형 시뮬레이션 프레임워크로써 국방에서 사용되는 무기체계들에 대해 기본 구조(Base System)를 제공하며, 이를 바탕으로 시뮬레이션 모델을 설계한다. 따라서, 국방 시뮬레이션을 수행하기 위해 사전에 고안된 모델들을 재사용하거나 혹은 수정하여 사용함으로써 높은 재사용성을 보장하고 같은 형식론과 구조 안에서 논의되는 모델들의 상호 연동성을 갖출 수 있다.

특별히 AddSIM은 개발 효율성을 위하여 시뮬레이션 환경에서 사용하는 시뮬레이션 코드와 체계의 특성과 전술이 반영될 사용자 코드 부분을 구분하여 시뮬레이션의 확장성을 보장한다. 자동으로 생성되는 시뮬레이션 관련 코드와 다르게 사용자 코드는 작성자의 자율성을 보장받기 때문에 사전에 AddSIM 환경 입장에서는 해당 영역에 어떤 코드가 기술될지 모르며, 따라서 완전성에 대한 보장이 어려워짐으로 폭넓은 형태의 검증이 필요가 있다.

2.2 M&S에서의 검증

오늘날 사용하는 M&S에서는 컴퓨터를 활용한 컴퓨터 시뮬레이션을 수행하고 있으므로 M&S 측면의 검증과 소프트웨어 검증 측면으로 구분하여 검증을 고려할 수 있다. Balci 등(2003)의 연구에서는 M&S 프로그램의 검증, 실증 및 인증에 대한 기본 개념을 정립하였다. 그리고 M&S에 대한 개발주기(lifecycle)과 함께 시행되는 V&V(Verification & Validation) 절차 즉, 평가 절차에 대해서도 제안하였다. Kung 등(2007)은 소프트웨어의 V&V에 대해 좀 더 명확하며 간결한 정의를 수행하였다. 소프트웨어를 대상으로 진행하는 V&V에 대해 보다 상세한 분류를 하며 정확한 V&V를 위한 통합적인 프로세스에 대해 정립하였다. 이상의 검증은 검증 절차에 대해서 정의하고 있으며 개발된 M&S 소프트웨어에 대한 검증 방법으로 크게 정적 검증(Static Verification)과 동적 검증(Dynamic Verification)을 고려할 수 있다. 정적 검증의 경우 소스 코드와 관련된 데이터 파일을 대상으로 하며, 소프트웨어의 실행 없이 수행하는 일련의 검증을 뜻한다. 특별히 정적 검증은 시스템에 영향을 줄 수 있는 보안 요구사항 또는 0으로 나누기 예외(Divide by zero exception)과 같은 문제를 실행하지 않고 검출할 수 있다. 정적검증의 시뮬레이션 모델이 실행되는 운영체제 환경과 사용되는 프로그래밍 언어적인 특성에 대한 분석을 수행하기 때문에 본 연구는 시뮬레이션 모델 코드에 대한 정적검증에 대해서는 다루지 않는다.

동적 검증은 실행 중인 프로그램을 대상으로 실행 중에 발생하는 사건들을 토대로 대상 프로그램이 요구사항에 부합하는지 검증을 수행하는 것을 뜻한다. 동적 검증은 동적 검증 대상 내에 정보를 추출할 수 있는 탐침(Probe)을 추가하고 사건과 사건 발생 시점을 토대로 검증을 수행한다. 양지용 등(2017)은 시뮬레이션 모델에 대한 동적 검증에 대한 전반적인 개념에 대해 소개하였으며, 최재웅 등(2017)은 DEVS 형식론 기반의 동적 검증 방법론을 제시하였다. 본 연구에서는 최재웅 등(2017)의 연구 결과를 바탕으로 DEVS 형식론 기반의 동적 검증 방법론을 AddSIM에 적용한다.

2.3 이산사건시스템 명세 형식론

이산사건시스템명세(Discrete Event System Specification: 이하 DEVS) 형식론은 B. P. Zeigler에 의해 고안된 집합론에 근거한 형식론이다(B. P. Zeigler, 1976). DEVS 형식론은 목표 시스템을 단위 요소 별로 나누고 이를 기반으로 각각의 모델을 만들고 조합하는 과정을 거쳐 전체 시스템을 표현하는 시뮬레이션 형식론이다. DEVS 형식론에는 시스템 단위 행동을 정의하기 위한 원자모델(Atomic Model)과 여러 모델들을 조합하여 새로운 모델을 구성할 수 있는 결합모델(Coupled Model)이 있다. 이 두 종류의 모델을 조합하여 대상 시스템을 계층적이고 조립 가능한 형태로 표현할 수 있다.

원자 모델은 단위 모델을 표현하기 위해 사용되는 기본 모델로써 세 개의 집합과 네 개의 함수로 구성된다. 원자 모델에 대한 명세는 다음과 같다.

$$AM = < X, Y, S, \delta_{int}, \delta_{ext}, \lambda, ta >$$

X : Input Events Set

Y : Output Events Set

S : States Set

$\delta_{ext} : Q \times X \rightarrow S$; External Transition Function

$Q = \{(s,e) \mid s \in S,\ 0 \le e \le ta(s)\}$; Total state of AM,

e : elapsed time

$\delta_{int} : S \rightarrow S$; Internal Transition Function

$\lambda : S \rightarrow Y$; Output Function

$ta : S \rightarrow R_{\infty}^{+}$; Time Advance Function

결합 모델은 여러 모델을 내부적으로 연결하여 만든 모델로, 결합 모델에 대한 명세는 다음과 같다.

$$CM = < X, Y, \{M_i\}, EIC, EOC, IC, SELECT >$$

X : Input Events Set

Y : Output Events Set

$\{M_i\}$: Component Models Set

$EIC \subseteq X \times \bigcup_i X_i$: External Input Coupling Relation

$EOC \subseteq \bigcup_i Y_i \times Y$: External Output Coupling Relation

$IC \subseteq \bigcup_i Y_i \times \bigcup_j X_j$: Internal Coupling Relation

$SELECT : 2^{\{M_i\}} - \phi \rightarrow M_i$

3. AddSIM 환경의 동적 검증 방법론

공통모의환경인 AddSIM에서는 다양한 종류와 서로 다른 수준의 시뮬레이션 모델들이 연동되어 시뮬레이션이 실행된다. 이 때 AddSIM은 자동으로 생성되는 시뮬레이션 코드와 사용자가 자유롭게 확장하는 사용자 코드를 바탕으로 구현된 시뮬레이션 모델들을 활용하여 시뮬레이션을 수행한다. 따라서 시뮬레이션 모델의 사용자 코드에 대한 동적검증은 AddSIM 환경 내에서 AddSIM 시뮬레이션 환경을 구성하는 모델들을 실행하면서 모델링 및 구현 과정에서 발생한 결함을 찾는다. 따라서 동적 검증을 수행하기 위해서는 시뮬레이션 모델이 실행되는 과정에서 발생하는 시뮬레이션 이벤트를 검출하여 이를 바탕으로 검증을 수행해야 한다. 본 장에서는 AddSIM 환경의 시뮬레이션 모델로부터 시뮬레이션 정보를 추출하는 방법과 이를 바탕으로 시뮬레이션 검증을 수행하는 방안에 대하여 제안한다.

3.1 AddSIM 사용자 코드의 동적 검증을 위한 명세기법

AddSIM 환경의 시뮬레이션 모델은 시뮬레이션 엔진이 관리하는 시뮬레이션 알고리즘에 의하여 시뮬레이션 모델에 명시된 특정 사건과 시간에 따라 시뮬레이션을 수행한다. 따라서 시뮬레이션 검증을 위해서는 시뮬레이션 모델이 발생시키는 사건에 대하여 특정 시간과 데이터에 대한 명세 방법을 활용해야 한다. 하지만 사용자가 무수히 많은 데이터와 발생 시간과의 조합을 명세할 수 없기 때문에 시스템의 상태와 관찰에 대한 방안을 활용한 명세 방안을 고려해야 한다.

본 연구에서 활용하는 DEVS형식론 기반의 동적 검증 요구사항 명세 방법은 입력 사건과 출력 사건 사이의 관계에 대하여 검증 대상 시뮬레이션 모델의 상태를 고려하여 명세를 수행한다. 즉, 검증 대상 시뮬레이션 모델이 가져야 할 상태에 대하여 DEVS 형식론의 원자모델의 상태로 기술하고 입력된 사건에 따라 상태 천이를 정의하고 반드시 관측해야 하는 데이터가 관측되는지 여부를 확인하여 검증을 수행한다. 이를 돕기 위하여 AddSIM의 시뮬레이션 모델의 동적검증을 위한 템플릿은 특정 변수(Target attribute), 연산자(Operator), 피연산자(Operand), 시간 제한조건(Time constraint), 그리고 결과(Result)으로 정의할 수 있다. 시뮬레이션 모델의 검증 요구사항은 기본적으로 특정 조건에서 특정 상태를 계속 유지하거나 특정 상태로 변화해야 하는 형태로 표현될 수 있다. 즉, 특정 상태를 표현하는 모델의 특정 변수가 어떤 값보다 작거나, 같거나, 크거나 혹은 변화하거나 변화하지 않거나 등으로 조건에 관해 확인할 수 있다. 템플릿의 구조는 다음 Fig. 1과 같다.

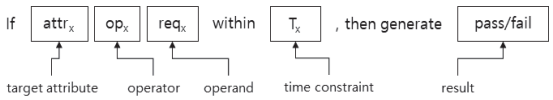

Fig. 1. Pseudo-code requirements specification template

Fig.1에서 정의한 템플릿은 시뮬레이션 모델이 실행하는 도중에 입력/출력 사건의 데이터가 특정 조건의 만족 여부를 확인해야 한다는 것을 의미하며 이를 확인하는 DEVS 명세를 생성할 수 있다. 즉, AddSIM 내에서 동작하는 모델에 특정 상태를 표현하기 위해 사용되는 변수를 추적하기 위해 DEVS 형식론을 기반으로 구성된 모델에 매 시뮬레이션 시간에 따라 해당 변수의 값 변화를 확인하면서 의사코드 템플릿의 조건에 맞게 실행되는지 혹은 값이 변화하는 지를 확인한다. 연산자와 피연산자를 활용하여 대상의 속성을 기술하고 속성의 연산을 정의함으로써 어떤 방식으로 검증을 수행할지 결정한다. 연산의 결과는 참/거짓의 형태로 반환된다. 동적검증 요구사항 명세의 구성요소는 다음 Table 1과 같다.

Table 1. Components of the Dynamic Verification Requirements Specification Model

Components	Description
Target attribute	Attributes that are directly used in verification among the attributes that exist in the verification target model
Operator	Various operators such as <, <=, >, >=, ==, !=
Operand	Certain values, such as numbers / letters
Result	Dynamic verification result given by Pass / Fail

Table 1에 소개한 명세 구성요소와 Fig. 1의 템플릿을 바탕으로 자동으로 생성된 동적검증의 요구사항 DEVS 모델은 Fig. 2과 같다.

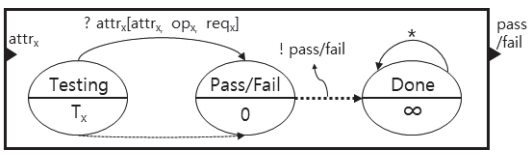

Fig. 2. Dynamic Verification DEVS Model

동적 검증 요구사항 DEVS 모델의 상태 집합은 총 세 개로 구성되며, 이는 Testing 상태, Pass/Fail 상태, 그리고 Done 상태이다. 모델의 초기 상태는 Testing 상태이며 이는 사전에 명시된 시간의 제한 조건에 맞춰 해당 상태를 유지할 수 있는 시간을 갖는다. 따라서 해당 상태 내에서 외부로부터 입력 사건을 받았을 때 주어진 조건의 여부를 반환한다. 이외에 특정 시간을 지나는 경우에도 주어진 조건을 만족하지 못하는 경우에도 이에 따른 검증 결과값을 반환한다. 검증 모델의 입력 포트를 통해 들어온 입력값은 특정 변수, 연산자, 피연산자 정보를 포함하고 있으며, 이를 통해 사전에 등록된 조건을 바탕으로 검증을 수행할 수 있다.

3.2 AddSIM 환경에서의 데이터 검출 방법

AddSIM 환경은 개발된 시뮬레이션 모델을 활용하여 사용자가 기술한 시나리오를 바탕으로 시뮬레이션을 구성하고 이를 실행한다. 따라서 AddSIM 모델의 동적 검증을 수행하기 위해서는 AddSIM에서 제공하는 기능을 활용한 방법과 시뮬레이션 모델로부터 정보를 검출하는 방안을 활용할 수 있다. AddSIM은 자체적으로 시뮬레이션 모델에서 발생하는 모든 정보를 저장하는 저널링(Journaling) DB를 활용한 동적 검증과 탐침 코드(Probe Code)를 활용한 동적 검증으로 구분할 수 있다. 동적검증은 정적검증을 통해 발견할 수 없는 오류를 찾을 수 있다는 점에서 그 활용도가 높다. 단일 모델의 동작(공학방정식 등)을 검증 가능하며 해당 과정에 대한 흐름은 다음 Fig. 3와 같다.

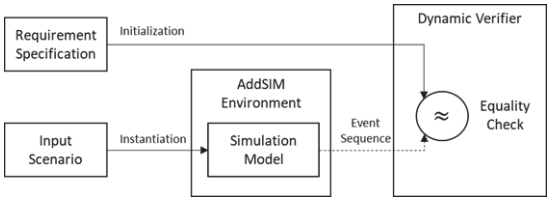

Fig. 3. Dynamic Verification of Single AddSim Model

Fig. 3에서의 실선은 AddSIM 환경과 동적검증기의 초기화 및 시뮬레이션 모델 생성을 나타내며 점선은 AddSIM 환경에서 발생한 이벤트들을 나타낸다. 이들 정보를 바탕으로 동적 검증기는 생서된 시뮬레이션 모델로부터 발생한 사건들에 대하여 검증을 수행한다. 이외에 AddSIM의 동적검증기는 다수의 시뮬레이션 모델에 대해서 검증 가능하다. 다음 Fig. 4는 다수의 AddSIM 시뮬레이션 모델에 대해서 검증에 활용하는 것에 대하여 도식화한 것이다.

Fig. 3과 유사하게 Fig. 4에서 실선은 초기화와 모델

생성에 대하여 나타내고 점선은 시뮬레이션 모델에서 생성된 이벤트 흐름을 나타낸다. 다수의 시뮬레이션 모델 동적 검증은 단일 시뮬레이션 모델의 동적 검증과 동일하게 개별 시뮬레이션 모델의 요구사항 부합성을 확인할 수 있음과 동시에 모델 간 이벤트 선후관계에 대한 검증도 수행할 수 있다. Fig. 4의 경우 입력 시나리오를 통해 생성된 Simulation Model A, B에 대하여 Simulation A에서 발생한 이벤트가 Simulation Model B로 전달될 때 발생한 동적 검증기에 전달되는 사건의 시뮬레이션 시간을 바탕으로 동적 검증을 수행할 수 있다. 예를 들어 시뮬레이션에 참여하는 모든 개체의 기동 모델이 실행 완료된 이후에 시뮬레이션 개체 모델의 탐지 모델이 실행되어야 한다는 요구사항에 대해서 동적으로 검증을 수행할 수 있다.

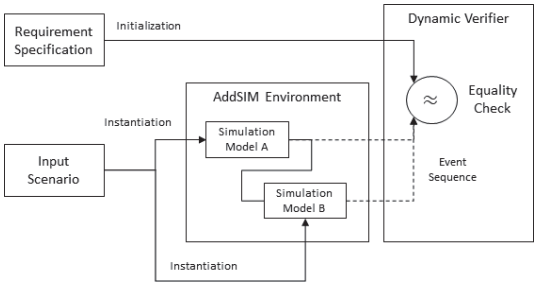

Fig. 4. Dynamic Verification of Multiple AddSim Models

3.3 사용자 코드의 동적 검증을 위한 검증 알고리즘

AddSIM의 사용자 코드에 대한 동적검증은 AddSIM 시뮬레이션 환경에서 제공하는 저널링 DB 데이터를 활용하는 방안과 탐침 코드를 활용한 것으로 나뉜다.. 다음 Fig. 5는 탐침코드를 시뮬레이션 모델에 삽입하여 동적 검증을 수행하는 방법에 대한 그림이다.

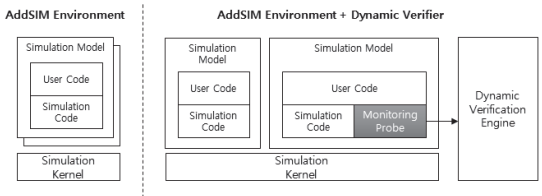

Fig. 5. Schematic of AddSIM's Dynamic Verification

Fig. 5와 같이 동적검증을 수행하지 않는 경우 모니터링 탐침이 없는 상황에서 시뮬레이션이 수행되고 동적검증을 수행하는 경우 탐침 코드를 삽입하여 시뮬레이션을 수행하여 동적 검증기가 정보를 받아 검증을 수행한다. 탐침 코드의 경우 사용자가 모니터링하고자 하는 정보를 분석하기 위해서 사용자 코드가 호출되는 영역에 탐침 코드를 삽입하고 시뮬레이션 수행하는 과정을 추적해야 하기 때문에 AddSIM의 시뮬레이션 커널(Kernel)에서는 일반적으로 수행할 수 없다. 따라서 사용자는 탐침 코드를 활용한 동적 검증을 수행하기 위하여 사용자가 탐침 코드를 자신이 생성한 모델에 삽입하고 모니터링을 수행해야 한다.

저널링 DB를 활용한 동적 검증의 경우 AddSIM에서 제공하는 기능을 활용한 방법이다. 일반적으로 시뮬레이션을 수행하는 경우 시뮬레이션 결과를 분석하기 위하여 시뮬레이션 시간과 해당 시뮬레이션 시간에 발생한 이벤트에 대하여 저장한다. 이를 활용한 방법하여 저장된 데이터로부터 발생한 이벤트를 추출하고 이를 기존의 동적 검증 모델에 입력을 활용하여 검증을 수행한다. 이상의 검증 방법을 활용하여 AddSIM의 동적검증을 수행하는 절차는 다음과 같다.

1. 요구사항에 따라 DEVS 동적검증 모델 생성
2. 동적검증 모델을 DEVS 시뮬레이션 엔진에서 실행
3. 모델 실행 결과에 따라 해당 요구사항에 대한 동적 검증 통과/실패 여부 판단

4. 사례연구

본 논문에서는 AddSIM의 동적검증기에 대한 사례 연구로 AddSIM의 시범 시제를 대상으로 검증을 수행한 사례를 소개한다. AddSIM의 시범 시제는 아군 레이더가 감지한 적 항공기를 아군 미사일 발사기와 미사일을 활용하여 요격하는 시나리오로 구성되어 있다. 시범 시제를 도식화한 결과는 아래 Fig. 6과 같다.

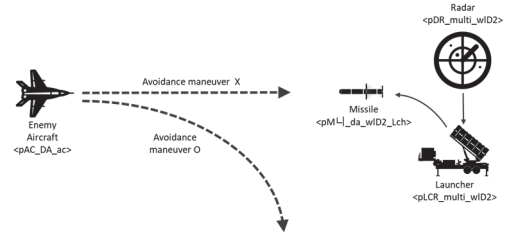

Fig. 6. Scenario of simulation example

Fig. 6에서 볼 수 있듯이 시뮬레이션이 수행되는 과정에서 검증해야 할 요구사항으로 적 비행체를 요격하기

위하여 지대공 미사일이 발사되는 상황에서 시범 시제에서 회피기동을 수행 여부 또는 특정 시간 내에 적 항공기를 탐지하고 이에 대한 적절한 대응을 수행하는지를 확인하기 위한 검증을 수행하였다. 시범 시제에 대한 요구사항은 다음과 같이 정의하였다.

요구사항 1: 적 항공기 모델은 시뮬레이션 시간 9 이내 반드시 아군 레이더 모델의 탐지범위(R_D: 4500) 내로 접근해야 한다.

요구사항 2: 아군 미사일 발사기 모델 내부의 요격 미사일 모델은 반드시 발사되어야 한다.

시범 시제 코드를 기반으로 요구사항 1에 대해 수식으로 표현하면 아래와 같다.

$$If\ |L_R - L_E| \leq R_D(4500)\ within\ 9, then\ generate\ r1\_pass$$

요구사항1에 대한 수식을 기반으로 생성한 DEVS 형식론 기반의 검증기 모델은 아래 Fig. 7과 같다.

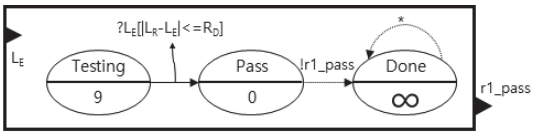

Fig. 7. Requirement 1 Verifier Model

상태 집합은 세 개로 구성되며 Testing 상태의 머무는 시간은 9이며, 해당 시간을 초과하거나 혹은 조건에 해당하는 값이 입력으로 들어오는 경우 Pass 상태로 전환되면서 결과를 출력한다.

요구사항 2에 대해 수식으로 표현하면 아래와 같다.

$$If\ M_I iLaunch = 1, then\ generate\ r2\_pass$$

요구사항2에 대한 수식을 기반으로 생성한 검증기 모델은 Fig. 8과 같다.

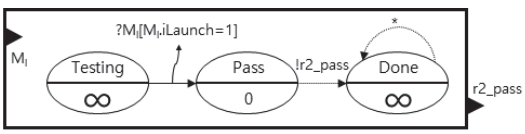

Fig. 8. Requirement 2 Verifier Model

마찬가지로 상태 집합은 세 개로 구성되나 이전과는 다르게 Testing 상태는 입력이 들어오기 전까지 계속적으로 입력 데이터가 들어오기를 기다리며 미사일에 발사되었다는 정보가 입력으로 들어오면 정상적으로 검증이 수행되었음을 알린다. 앞서 설계된 두 검증기 모델을 AddSIM에서 실행을 위한 DEVS 모델 변환 방법을 활용(김도형 등, 2014)하여 DEVSim++(Kim, 2017)를 사용하여 실행시킨 결과는 아래 그림 9와 같다.

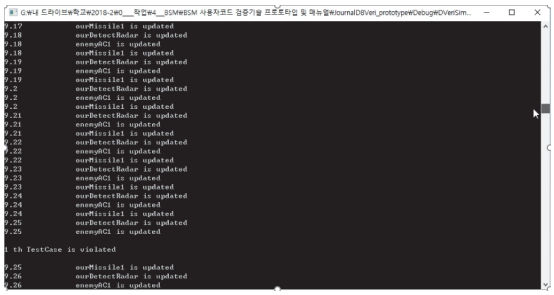

Fig. 9. Verification execution result

AddSIM의 시범시제가 요구사항 1을 만족시키지 못했음을 출력하고 있다. 이는 시뮬레이션 시간 9 이내에 적 항공기를 레이더가 탐지 못했음을 의미하며 시범시제의 소스코드를 내의 결함을 발견하여 이를 수정하였다.

5. 결론

공통모의환경인 AddSIM은 AddSIM환경 안에서 실행할 수 있도록 지원하는 시뮬레이션 코드와 사용자의 확장성을 제공하기 위해 제공하는 사용자 코드를 기술하여 시뮬레이션 모델을 개발한다. 이때 자동으로 생성되는 시뮬레이션 코드의 경우를 제외하고 사용자가 작성하는 사용자 코드의 경우 시뮬레이션의 완전한 동작을 보장하기 위하여 필수적으로 검증을 수행해야 한다.

본 연구에서 AddSIM 모델을 검증하기 위해 사용한 방식은 시뮬레이션 코드 내에 탐침코드를 삽입하여 시뮬레이션 중에 발생하는 이벤트를 모니터링하는 방법과 AddSIM 환경에서 제공하는 저널링 DB로부터 시뮬레이션 정보를 불러들여 검증하는 방법을 제안하였다. 탐침코드의 경우 도메인의 이해와 해당 모델 자체의 이해를 필요로 하기 때문에 모델 개발자가 필요에 따라 삽입하는 방식으로 수행되며, 저널링 DB의 경우 요구사항 명세 템플릿 기반으로 생성된 동적 검증기 모델을 활용하여 검증을 수행하였다.

제안하는 방법의 경우 기본적으로 동적검증을 수행하는 모델은 하나의 요구사항에 대해서만 검증을 수행할 수 있으며, 여러 요구사항에 대해 검증하고자 하는 경우 각 요구사항에 해당하는 검증 모델을 생성하여 실행할 수 있어 추후 연구로는 모델 간 연동에 대한 검증과 AddSIM 환경에서 이뤄질 수 있는 통합적인 정적 검증에 대해 진행한다.

References

Balci, O. (2003, December). Verification, validation, and certification of modeling and simulation applications: verification, validation, and certification of modeling and simulation applications. In Proceedings of the 35th conference on Winter simulation: driving innovation (pp. 150-158). Winter Simulation Conference.

Kim, T.G. (2017) DEVSim++ User Manual, available at http://smslab.kaist.ac.kr

Kung, D., & Zhu, H. (2007). Software verification and validation. Wiley Encyclopedia of Computer Science and Engineering.

Oh, H. S., Park, S., Kim, H. J., Lee, T., Lee, S., Kim, D., ... & Park, J. H. (2014, December). AddSIM: A new Korean engagement simulation environment using high resolution models. In Simulation Conference (WSC), 2014 Winter (pp. 2942-2953). IEEE.

Zeigler, B. P., Praehofer, H., & Kim, T. G. (1976). Theory of modelling and simulation (Vol. 7). New York: Wiley.

김도형, 박주혜, 오현식, & 박삼준. (2014). 컴포넌트기반 체계모의환경(AddSIM)에서 실행을 위한 DEVS 모델 변환 방법. 한국정보과학회 학술발표논문집, 81-83.

양지용, 최재웅, & 최창범. (2017). 모듈형 시뮬레이션 프레임워크인 ADDSim에서 모델 동적 검증 방법. 한국시뮬레이션학회 학술대회 논문집, 54-56.

최재웅, 양지용, & 최창범. (2018). DEVS 형식론을 활용한 AddSIM의 사용자 정의코드 및 기본체계모델 동적검증. 한국경영과학회 학술대회논문집, 2131-2135.

양 지 용 (21731002@handong.edu)

2017 한동대학교 컴퓨터공학, 전자공학 학사
2019 한동대학교 정보통신공학 석사

관심분야 : 모델링 형식론, 모델 검증, 시뮬레이터 검증

최 창 범 (cbchoi@handong.edu)

2005 경희대학교 컴퓨터공학 학사
2007 KAIST 전산학 석사
2014 KAIST 전자공학 박사
2014~ 현재 한동대학교 ICT창업학부 조교수

관심분야 : DEVS 형식론, 소프트웨어 품질 보증, VV/A