Journal of the KIMST, Vol. 24, No. 4, pp. 408-417, 2021
ISSN 1598-9127(Print) · 2636-0640(Online)
DOI https://doi.org/10.9766/KIMST.2021.24.4.408

Research Paper 시험평가 부문

무기체계 통합시뮬레이션 소프트웨어의 품질 속성 검토

오현식[*,1)] · 김도형[1)] · 이순주[1)]

[1)] 국방과학연구소 제2기술연구본부

Review on the Quality Attributes of an Integrated Simulation Software for Weapon Systems

Hyun-Shik Oh[*,1)] · Dohyung Kim[1)] · Sunju Lee[1)]

[1)] *The 2nd Research and Development Institute, Agency for Defense Development, Korea*

(Received 3 March 2021 / Revised 13 May 2021 / Accepted 5 July 2021)

Abstract

This paper describes the quality attributes of an integrated simulation software for weapon systems named Advanced distributed simulation environment(AddSIM). AddSIM is developed as a key enabler for Defense Modeling & Simulation(M&S) systems which simulate battlefields and used for battle experiments, analyses, military exercises, training, etc. AddSIM shall provide a standard simulation framework of the next Defense M&S systems. Therefore AddSIM shall satisfy not only functional but also quality requirements such as availability, modifiability, performance, testability, usability, and others. AddSIM consists of operating softwares of hierarchical components including graphical user interface, simulation engines, and support services(natural environment model, math utility, etc.), and separated weapon system models executable on the operating softwares. The relation between software architectures and their quality attributes are summarized from previous works. And the AddSIM architecture and its achievements in the aspect of quality attributes are reviewed.

Key Words : Integrated Simulation Software(무기체계 통합시뮬레이션 환경), AddSIM, Software Architecture(소프트웨어 아키텍처), Quality Attributes(품질 속성)

1. 서 론

지금까지 우리나라의 국방M&S(Modeling & Simulation) 체계는 선진국의 지원 및 이를 참조로 국내 개발한 체계들이 다양하게 운영되었으나, 고도화되는 현존, 미래 무기체계의 특성을 반영하여 미래 전장을 분석해야 하는 군의 요구사항을 수용하는데 한계가 있었다[1].

향후 요구되는 국방M&S체계는 기존 체계들에 비해 실 무기체계 객체 단위의 특성과 행위를 구체적으로 모사할 수 있어야 하며, 지속적으로 획득되는 무기체

* Corresponding author, E-mail: hyunshik96@gmail.com

계들을 유연하고, 안정적으로 모델링하여 시뮬레이션에 반영할 수 있어야한다. 또한 지, 해, 공으로 서로 분리된 전장이 아니라 분석목적에 따라 타 군의 전력과 함께 동일 시뮬레이션 안에서 모사될 수 있어야 한다.

이에 미래 각 군에서 필요한 표준화된 무기체계 교전 시뮬레이션 환경 개발을 목표로 AddSIM(Advanced distributed SIMulation environment)이 개발되었다. 2009년 v1.0을 시작으로[2], v2.0[3]을 거쳐, 무기체계 표준 라이브러리가 추가되어 v3.0이 완성되었다[4]. 즉, 고해상도 교전급 분석모델의 기능 요구사항을 만족시키는 기술이 확보되었다고 판단된다.

그런데 사용자들이 만족하는 좋은(good) 시스템을 개발하기 위해서는 단순히 기능 요구사항을 만족시키는 것에서 나아가 품질 요구사항에 대한 검토가 반드시 필요하다. 특히 설계도나 개발결과물을 직관적으로 볼수 있는 하드웨어 중심 시스템에 비해 소프트웨어 중심 시스템은 그 품질을 확인하기 위해 별도의 노력이 필요하다.

본 논문에서는 AddSIM 품질 속성들에 대한 성과를 검토하기 위하여 2장에서 국방M&S 시스템으로서의 AddSIM 개발 배경을 소개하고, 소프트웨어 품질 속성에 대한 개념을 요약한다. 3장에서는 AddSIM의 품질 속성을 검토하는데 가용성, 변경용이성, 성능, 시험용이성, 사용편의성, 그리고 모델 개발편의성에 대하여 기술한다. 끝으로 4장에서 결론을 맺는다.

2. 배경 및 관련 연구

2.1 AddSIM 개발 배경

고해상도 교전급 시뮬레이션에 대한 기본적인 요구사항은 2000년대 Joint Modeling and Simulation System (JMASS)에 처음 제시되었다[5,6]. JMASS는 시뮬레이션 엔진을 기반으로 아군과 적군의 무기체계 모델을 독립적으로 개발하고, 실행 중 공통된 환경모델을 사용하는 개념을 갖는다. JMASS 사업 이후 미 공군의 시뮬레이션 아키텍처가 발전되었을 것이라 추정되지만, 더 이상의 자료가 공개되지 않았다. 이러한 개념이 발전하여 페더레이트(federates)와 객체(entities), 그리고 객체를 구성하는 컴포넌트(components)를 구분하고 이들 간의 상호운용, 각 요소들의 구성 및 기술융합을 위한 계층형 아키텍처가 제안되었다[7,8]. 이어 Steinman 등 논문[9]를 통해 이를 구현한 Warp IV라는 오픈소스를 포함한 프레임워크(framework)를 공개하였으나 국내에서 추가적인 내용을 확보하는 데는 한계가 있었다.

국내에서도 국방M&S기술 특화연구센터의 지원으로 무기체계 효과도 분석을 지원하기 위한 통합 시뮬레이션 환경인 OpenSIM을 개발하였다[10]. OpenSIM은 공학급/교전급 모델을 작성하고 각 모델 간 통합 시뮬레이션을 지원하기 위해 서비스 및 도구를 개발하는데 초점을 두고 있으며, 유사 무기체계 모델들의 재사용성을 높이기 위해 모델의 물리 구조와 행위 구조를 하나의 단위로 묶어 캡슐화 한다. OpenSIM의 서비스 아키텍처는 큰 틀에서 Steinman[8]이 제안한 표준 시뮬레이션 아키텍처를 따르고 있으며, 아키텍처를 구성하는 소프트웨어 및 서비스를 목적에 맞게 일부 변경하였다. 이러한 연구들을 참조하여 시뮬레이션 엔진, 무기체계 모델, 자연 환경 모델의 분리하는 개념으로 AddSIM v1.0이 개발되었다[2].

교전 시뮬레이션 모델의 또 하나의 주요한 고려사항은 '컴포넌트로 개발되는 무기체계 모델을 어느 정도 표준화 할 것인가'이다. AddSIM v2.0의 무기체계 모델에서 플레이어라고 명명되는 개별 무기체계 모델 컴포넌트는 내부에 세부기능을 담당하는 하부 컴포넌트(sub-components)를 포함한다는 원칙적 가이드를 제시하였고 각 컴포넌트들 간의 인터페이스 연결방식과 시뮬레이션 엔진 상의 실행방식을 제시하였다. 참고로 AddSIM은 Table 1의 프레임워크 타입 중 맨 위의 공통 라이브러리(common library) 타입에 속한다고 볼 수 있다. 그런데 무기체계 모델 개발자의 자유도를 많이 보장하기 때문에 여러 개발자들이 유사한 무기체계 모델을 개발할 때 완전히 다른 구조로 작성하고 시뮬레이션 엔진과의 인터페이스만 맞추는 방식으로도 운영이 가능하게 되었다. 이는 개발자에게 유연하게 모델링 할 수 있는 프레임워크를 제공한다는 측면에서 장점도 있지만, 공용 활용을 위해 무기체계 모델을 축적해 나가는 측면에서는 모든 개발자들이 하위 컴포넌트 수준에서 공통적으로 이해할 수 있도록 표준 모델 구조를 제시하는 것이 필요하다. 이러한 개념은 OneSAF(One Semi-Automated Forces)도 채택하고 있으며[11], 상업적으로 판매되는 교전 시뮬레이션 도구들인 FLAMES(FLexible Analysis Modeling and Exercise System, 미 Ternion), SADM(Ship Air Defense Model, 호주 BAE Systems), TESS(Tactical Engagement Simulation

Table 1. Framework types summary[11]

Framework type	Components	Composition mechanism
Common library	Models implemented as software modules	Component interfaces defined by components or framework; components linked into common executable; data exchanged via method calls
Product line architecture	Software modules	Component interfaces defined by framework; components linked into common executable; data exchanged via method calls
Interoperability protocol	Independent executable	Components execute independently as separate processes; data exchanged via network messages
Object model	Conceptual model	None; connection depends on mapping and implementing conceptual module within other framework
Formal	Formal model	Interpreter for formal models
Integrative environment	Model implemented as file, spreadsheet, ot software module	Components “wrapped” with special interface software; components linked into common executable; data exchanged via method calls

System, 캐나다 TTI), AFSIM(Analytic Framework for Simulation, Integration, & Modeling, 미국 Boeing) 등에서도 무기체계 모델 개발을 위한 표준모델 아키텍처를 적용하고 있다. AddSIM에도 공통 시뮬레이션 소프트웨어 프레임워크 상에서 작동하는 무기체계 모델 표준인 기본체계모델(BSM: Base System Model)을 제공한다.

2.2 소프트웨어 아키텍처

Bass과 Kazman[12]은 소프트웨어 아키텍처를 다음과 같이 정의하였다. “그 시스템을 이해하기 위해 필요한 구조들의 집합으로, 소프트웨어 요소들(elements)과 그들 간의 관계, 그리고 각각의 특성들을 포함한다.” 그리고, 그 특성을 아래와 같이 기술하였다.

1) 소프트웨어 구조들의 집합으로 구성되는데 모듈 구조, 컴포넌트-연결 구조, 할당 구조로 구분된다.
2) 소프트웨어 아키텍처는 추상화된 형태로 소프트웨어를 이해하도록 한다. 주로 각 요소들의 외부로 드러난 내용이 표현된다.
3) 모든 소프트웨어는 아키텍처를 가진다. 아키텍처의 수준이 소프트웨어의 품질을 나타낸다.
4) 아키텍처는 요소들의 행위를 포함한다. 각 요소의 역할과 다른 요소들과의 관계가 표현되어야 한다.
5) 모든 아키텍처가 좋은 것은 아니다. 그러므로 적절한 아키텍처를 선정하고 발전시켜 가야한다.

더 나아가 이러한 특성을 만족하기 위한 설계 가이드라인을 제시하였다.

1) 정보은닉과 수행관점으로 잘 분리된 모듈화
2) 잘 알려진 아키텍처 패턴과 개발전술을 활용
3) 특정 도구(버전)에 의존성 지양
4) 데이터 생산 모듈과 사용 모듈 분리
5) 컴포넌트와 모듈의 1대1 할당을 지양
6) 하나의 프로세스(process)는 하나의 프로세서(processor)에서 실행되도록 지정
7) 컴포넌트들이 관계하는 방법의 수를 최소화
8) 아키텍처에 관련 있는 명확한 자원(예, 네트워크 부하, 처리시간 등)을 관심요인으로 포함

이러한 개념은 Table 2와 같이 지난 60여 년간의 소프트웨어 공학 역사의 산물로 프로그램의 복잡한 기능성을 어떻게 효과적으로 표현할 것인가의 수단으로 발전하였다.

Table 2. History of software engineering

Year	Keywords	Contents
1960s	Subroutine	preliminary programming language structure
1970s	Module	developing structure to satisfy requirements
1980s	Object	modular, information hiding, inheritance, objects having interoperable properties
1990s	Framework	standard object oriented architecture, separating technical and business parts (e.g. word processor, spread sheet, etc.)
2000s	Middleware Architecture	standard middleware and architecture using reliable services for security and performance from internet environment
2010s	Software Product Line	product scoping, feature modeling, and SPL platform

2.3 소프트웨어 아키텍처의 표현

소프트웨어 아키텍처를 표현하는 방법에는 여러 가지가 있을 수 있겠으나 복잡한 시스템을 추상화하기 위하여 보는 관점에 따라 여러 개의 뷰(view)로 표현할 수 있다. 다음은 대표적인 4+1 뷰이다.

1: 모듈 뷰(module view): 소프트웨어의 논리적 구조를 추상화한 것으로 Implementation(구현) 뷰라고도 불린다.
2: 컴포넌트와 커넥션 뷰(component and connection view, C&C view): 동시성과 기능분산에 의한 프로세스 관점의 뷰로 runtime(실행) 뷰라고도 불린다.
3: 할당 뷰(allocation view): 소프트웨어 모듈, 라이브러리, 하위시스템 등을 구분하는 개발 관점의 뷰
4: 배치 뷰(deployment view): 개발결과물에 대한 물리적 관점에서의 뷰
+1: 유즈케이스 뷰(use-case view): 시스템이 실행되는 내용의 이해를 도와 위의 다른 뷰들을 작성하는데 기초가 되는 활용 사례(use-case) 관점의 뷰

이런 아키텍처 뷰들 중에서 개발자는 시스템의 특성에 맞게 테일러링하여 활용하게 된다. 본 논문에서는 AddSIM의 아키텍처 자체를 제시하기보다 아키텍처를 설계하기 위해 고려한 품질 속성들에 대한 검토를 수행하였다.

2.4 소프트웨어 품질 속성의 이해

소프트웨어 발전과정을 통해 정리된 품질 속성과 각각을 달성하기 위한 전략들은 참고문헌 [12]에서 아래와 같이 소개되고 있다.

2.4.1 가용성(Availability)

가용성(Av)이란 소프트웨어가 필요한 시점에 적절히 그 임무를 수행하는 척도를 말하는데 Equation (1)과 같이 표현된다.

$$Av = \frac{MTBF}{(MTBF + MTTR)} \tag{1}$$

MTBF는 고장간 평균시간(Mean Time Between Failures)이며, MTTR은 평균 수리시간(Mean Time To Repair)을 말한다. 즉, 고장이 안 나는 신뢰도(reliability) 뿐 아니라 고장 시 수리를 얼마나 빠르게 할 수 있는가도 중요한 요소가 된다. 이를 위해 오류의 탐지, 회복과 오류 방지 전략이 활용된다.

2.4.2 상호운용성(Interoperability)

상호운용성은 2개 이상의 시스템이 의미 있는 정보를 유용하게 교환하는 정도이다. 이는 단순히 자료의 교환 뿐 아니라 의미적으로 정확한 정보를 교환하는 것을 포함한다. 상대 서비스를 찾는 방법과 인터페이스 관리 방안을 통해 확인할 수 있다.

2.4.3 변경용이성(Modifiablility)

소프트웨어는 필연적으로 개발 중 또는 개발 이후에도 다양한 변경요구가 발생하게 된다. 변경용이성은 이러한 변경에 대한 비용과 위험성의 수준을 말한다.

변경용이성을 확보하기 위해서는 모듈의 크기를 줄이는 일과 응집력(Cohesion) 증대, 상호 관계성 감소, 그리고 모듈들의 바인딩 시점 연기 등의 전략이 활용된다.

2.4.4 성능(Performance)

성능은 시간에 관련된 것이다. 이벤트가 발생하였을

때 자원요구와 자원관리가 효율적으로 수행되어 제한된 시간 내에 적정한 응답을 산출하는가에 대한 수준을 말한다.

성능을 높이기 위해서는 자원요구를 제어하는 것과 자원을 관리하는 전략이 활용된다.

2.4.5 보안성(Security)

보안성은 권한이 없는 외부 접속으로부터 자료와 정보를 보호하는 시스템의 능력이다.

외부 보안공격 탐지, 저항, 대응조치 및 공격으로부터의 복구 전략이 활용된다.

2.4.6 시험용이성(Testability)

시험용이성은 오류가 있는 부분을 시험을 통해 최대한 빠르게 찾아낼 수 있는 성질을 의미한다.

시험용이성을 증대하기 위해서는 시스템의 상태를 제어하고 관찰할 수 있게 하는 방안과 시스템의 복잡도를 감소시키는 전략이 활용된다.

2.4.7 사용편의성(Usability)

사용편의성은 사용자들이 얼마나 쉽게 소프트웨어를 사용할 수 있게 하는가의 수준을 말한다. 여기에는 시스템 특성 학습의 용이성, 효율적인 사용법 제공, 사용자 실수 효과를 최소화(안정적인 실행취소(undo) 기능), 사용자 요구에 부합하는 시스템, 사용자 신뢰성 및 만족도 증대가 포함된다.

사용편의성을 증대하기 위해서는 사용자인터페이스를 별도로 분리하는 전략이 필요하다. 이를 통해 사용자의 실수가 시스템에 미치는 영향을 분리할 수 있으며, 시스템에서도 임무 모듈과 사용자 모듈 그리고 시스템 모듈을 분리하는 방법이 활용된다.

2.4.8 그 외 품질 속성

그 외에도 가변성(Variability), 휴대성(Portability), 분산개발성(Development Distributability), 이동성(Mobility), 확장성(Scalability), 배포성(Deployability), 안전(Safety), 관제성(Monitorability) 등 다양한 품질 속성이 있다.

아키텍처 설계에 있어서 이러한 모든 품질 속성을 모두 고려하기 보다는 시스템의 특성에 맞게 필요한 품질 속성을 선정하고 적용하면 된다.

3. AddSIM 품질 속성 검토

3.1 AddSIM 개요

AddSIM은 무기체계 통합시뮬레이션 소프트웨어이다. Figure 1에 나타난 바와 같이 사용자는 AddSIM의 사용자 인터페이스(User Interface, UI)를 통해 무기체

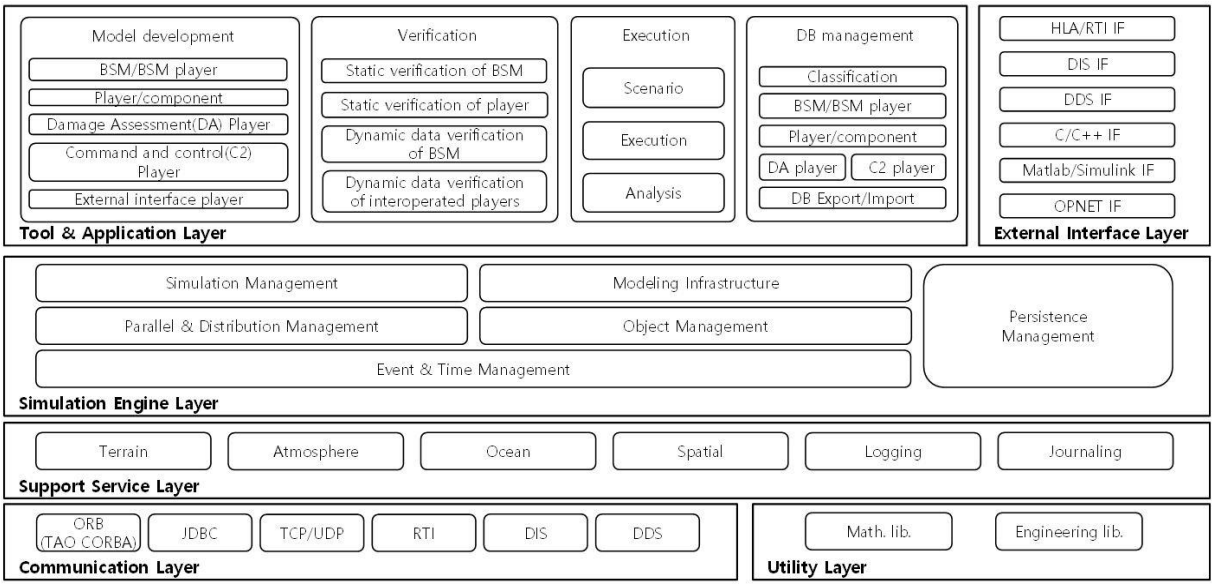

Fig. 1. Layered structure of AddSIM 3.0

계 모델을 개발한 뒤, 분석 목표에 따라 시나리오를 설정하고 시뮬레이션 실행을 통해 결과를 확인하고 분석할 수 있다.

이러한 기능을 만족할 수 있는 AddSIM의 구조는 Figure 1과 같다. 좌측 상단의 운용도구 계층에는 무기체계 모델을 개발하고 시뮬레이션을 실행하기 위한 여러 사용자 인터페이스를 담당한다. 이러한 사용자의 입력에 따라 시뮬레이션을 실행하는 시뮬레이션 엔진 계층이 있다. 이 시뮬레이션 엔진은 필요에 따라 지원 서비스와 통신, 유틸리티, 외부 인터페이스 계층을 활용한다.

또한 한국군의 주요 무기체계를 대상으로 Table 3에서 보는 바와 같이 21종의 기본체계모델(Base System Model, BSM)을 개발하였으며, 무기체계의 행위와 인터페이스를 표준화하여 정의하는 아키텍처를 개발하였다.

Figure 2는 기본체계모델의 일반적인 구조를 보여주는데 공유데이터와 물리컴포넌트, 논리컴포넌트를 포함하고 있음을 알 수 있다. <괄호>는 사례를 표시한 것으로 무기체계별로 다르게 모델링된다.

공유데이터에는 정보(다른 모델로부터 받은 정보 관리), 파라미터(해당 무기체계의 고정적인 특성), 그리고 속성(시뮬레이션 중에 변경되는 무기체계의 상태변수)이 포함된다.

물리모델은 무기체계의 물리적 거동이나 기능을 모델링하고, 논리모델은 무기체계를 운용하는 교전 수칙이나 운용자(인간 또는 인공지능)의 논리를 모델링한다.

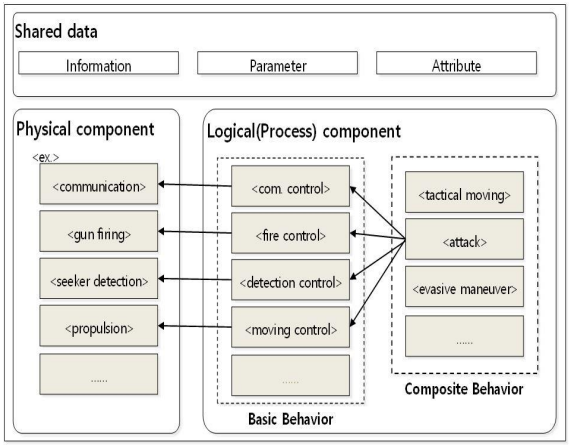

Fig. 2. BSM's common structure

Table 3. Base system models(21 systems)

Ground	Naval	Air	Missile	Etc.
• Tank • Ground vehicle • Artillery • UGV • Anti-tank • Missile ground system	• Surface ship • Submarine • UUV/USV • Torpedo	• Fix-wing aircraft • Rotary -wing aircraft • UAV	• Ground missile • Air missile • Navel missile • Anti-air missile	• Sonar • Mine • Naval decoy • Air decoy

3.2 AddSIM 품질 속성 검토

2장에서 언급한 바와 같이 소프트웨어의 아키텍처는 소프트웨어의 품질 속성을 파악할 수 있는 근거를 제공한다.

이러한 관점에서 본 절에서는 AddSIM이 가지고 있는 주요 품질 속성에 대해 분석해 본다. 2.4절에서 제시한 품질 속성 중 고해상도 교전급 분석모델을 만들기 위한 AddSIM임을 고려하여 가용성, 변경용이성, 성능, 시험용이성, 사용편의성을 검토하였으며 무기체계 모델을 별도 개발자들이 개발하게 됨에 따라 모델 개발편의성을 추가하여 검토한다. 이를 통해 AddSIM 아키텍처에 대한 정성적인 평가를 실시하고 향후 발전 요소를 식별해 보고자 한다.

3.2.1 가용성(Availability)

AddSIM의 가용성을 논하기 위해 AddSIM의 운용시간을 구분하면 무기체계 모델을 개발하는 시간과 시나리오를 만드는 시간, 시뮬레이션을 실행하는 시간, 시뮬레이션 결과를 확인하는 시간이 있다. 또한 발생 가능한 오류를 식별해야 하는데 AddSIM 운용도구 및 시뮬레이션 엔진이 정상작동하지 않는 오류, 사용자가 개발하여 넣은 무기체계 모델(외부 인터페이스 포함)의 오류, 사용자가 설정한 시나리오의 오류로 구분한다. 각 단계별 발생 가능한 오류는 Table 4와 같다. 즉시명적인 영향을 주는 도구와 엔진에 대하여는 많은 시험과 수정을 통해 AddSIM 이 완성되어 배포되는 시점에는 거의 발생하지 않도록 개발이 이루어진다. 반면 사용자가 개발한 무기체계 모델에는 오류가 자주 발생하게 된다. 이는 C++언어 개발도구를 활용하여 사용자가 코딩을 해야 하는 부분이기 때문이다. 또한 시나리오 작성의 경우 주로 도구를 사용하기 때문

에 많은 오류를 예방해 준다. 하지만 가끔 사용자가 필요한 무기체계를 누락하거나 연결하지 않을 가능성은 존재한다.

Table 4. Frequency/consequency of faults

Fault / Phase	Tool & Engine Fault	Model Fault	Scenario Fault
Developing Model	Remote/ Critical	Frequent/ Moderate	-
Setting Scenario	Remote/ Critical	-	Occasional/ Moderate
Executing Simulation	Remote/ Critical	-	-
Checking Results	Remote/ Moderate	-	-

그리하여 AddSIM은 사용자 만든 오류에 대처하기 위해 Figure 3과 같은 개념의 모델 검증 기능을 제공한다. 이러한 조건을 고려하여 Figure 3과 같은 검증 도구를 개발함으로써 가용성 향상에 기여한다. 즉 식 (1)의 가용성을 구성하는 항목 중 MTBF는 실세계를 반영하므로 동일하다고 보면 MTTR을 감소시키는 방향으로 개발하여 가용성을 높일 수 있다. 자동화된 검증 기능을 제공함으로써 조기에 오류를 찾고 수정이 가능하도록 하였다. AddSIM의 모델 검증 기능은 실행 전 사용자 개발 모델을 검증하는 정적도구와 시뮬레이션 실행 중 모델을 검증하는 동적도구로 구성된다. 정적도구는 AddSIM 코딩규칙 위배여부 검증, BSM 구조 유효성 검증, 플레이어 객체 초기값 유효성 검증, 도메인 유효성 검증(e. g. 항공기의 최대 적재 가능 무게) 기능을 제공하여 사용자의 실수를 컴파일하여 실행하는 작업을 수행하기 전에 검출해 준다. 또한 동적도구는 BSM 실시간 데이터 유효성 검증, 연동 플레이어 실시간 데이터 유효성 검증, 상태 천이에 대한 사용자 요구사항 충족 검증(DEVS 형식론을 바탕으로 행위에 대한 검증 수행: 이벤트의 선/후 관계)을 가능하게 하여 오류가 발생한 즉시 실행을 중단하고 문제점을 파악할 수 있으므로 불필요한 시뮬레이션 실행시간을 줄여 줄 수 있다. 각 검증모듈에 대한 구체적인 방법론은 Yang과 Choi의 논문[13]을 참고하기 바란다.

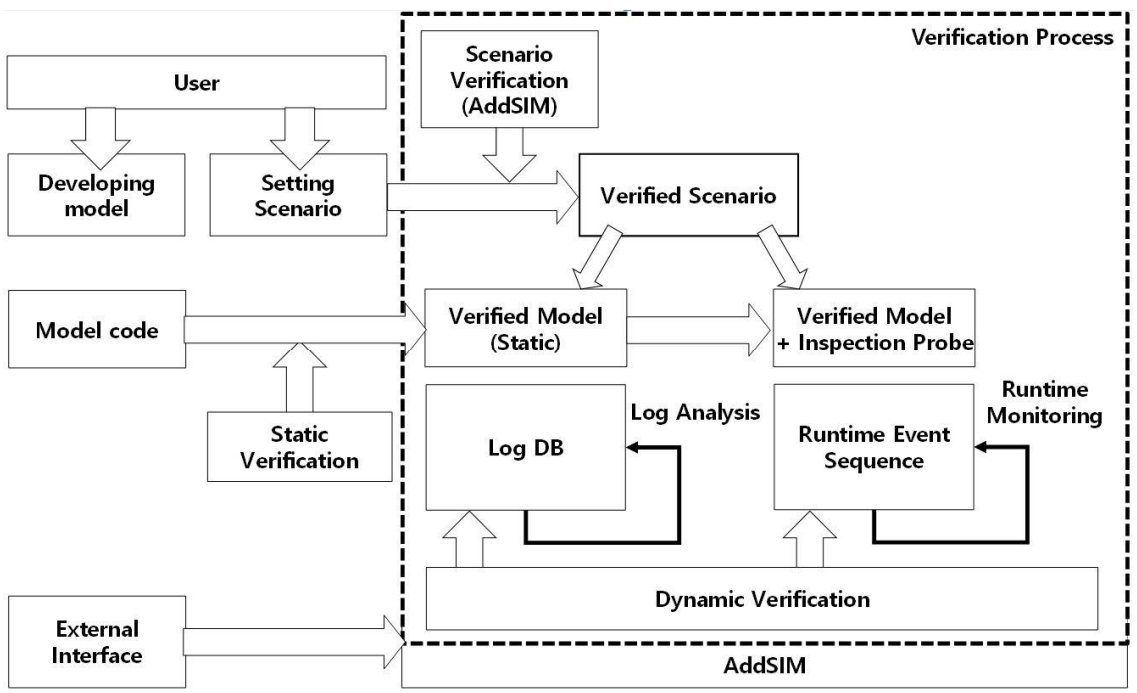

Fig. 3. AddSIM verification concept

3.2.2 변경용이성(Modifiablility)

사용자가 빈번하게 무기체계 모델을 수정해야 하는 AddSIM은 변경용이성에도 주의를 기울였다. AddSIM과 무기체계 모델을 분리하는 개념에서부터 AddSIM을 Figure 1과 같이 운용도구, 커널(시뮬레이션 엔진), 지원 계층으로 기능에 따라 모듈화 하고 하위 컴포넌트가 기능별로 구분되어 있으며 각 모듈의 사용자 접근성과 상호 호출관계에 따라 계층화 되어 있다. 이를 통해 변경이 필요한 경우 다른 모듈로의 수정사항 전파가 최소화되고 해당 모듈에 대한 수정이 가능해지므로 변경용이성이 증대된다. 또한 BSM의 경우도 Figure 2와 같이 물리 컴포넌트와 논리 컴포넌트로 구분되어 유지보수성에 기여한다. 상세한 설계 내용은 AddSIM v3.0의 소프트웨어 설계기술서(SDD)에 포함되어 있다.

또한 시뮬레이션 엔진과 독립적으로 우선 운용도구(UI)만 실행되어 시뮬레이션 준비 및 시나리오 생성을 할 수 있게 분리하였다. 운용도구에서 실행버튼을 누르는 경우 운용도구에서 설정된 모든 내용이 파일(xml) 형태로 시뮬레이션 엔진에 전달되고, 시뮬레이션 엔진의 모든 구성요소들은 개별적인 동적 연결 라이브러리(Dynamic Link Library, DLL)로 구성되어 실행 시점 바인딩이 이루어진다. 상호 실행이 분리된 운용도구와 시뮬레이션 엔진 간의 정보교환은 TAO 공용객체 브로커(CORBA: Common Object Request Broker Architecture)[14]를 통해 이루어진다. Figure 4는 모델 개발자(developer) 또는 분석자(analyst)인 사용자가 AddSIM을 사용하는 동안 각 모듈이 독립적으로 실행되면서 필요한 정보를 교환하는 지에 대한 개념을 보여준다.

3.2.3 성능(Performance)

AddSIM은 실행속도를 주요 성능지표로 판단하여 시뮬레이션 시간 전진을 객체별로 비동기적(asynchronous)으로 관리할 수 있게 하여 불필요한 모델 호출을 감소하도록 설계하였을 뿐 아니라 실행 단계에서 화면의 로그표시 여부, 각 속성(attributes)들에 대한 저널링 여부, 저널링 주기 설정, 시뮬레이션 실행간 지도상 객체 이동 표시 여부, 롤백기능 사용 여부, 시뮬레이션 배속 설정 등 성능에 영향을 줄 수 있는 선택항목을 제공하여 사용자의 목적에 따라 실행시간을 감소시킬 수 있게 한다. 또한 자원 관리측면에서 다중 스레드(Thread)를 이용한 병렬 시뮬레이션 기능을 개발하였고, batch 시뮬레이션이나 Monte-Carlo 시뮬레이션 시 다수의 컴퓨터에서 동시 수행할 수 있게 하여, 여러 컴퓨터의 자원을 활용할 수 있도록 설계하였다.

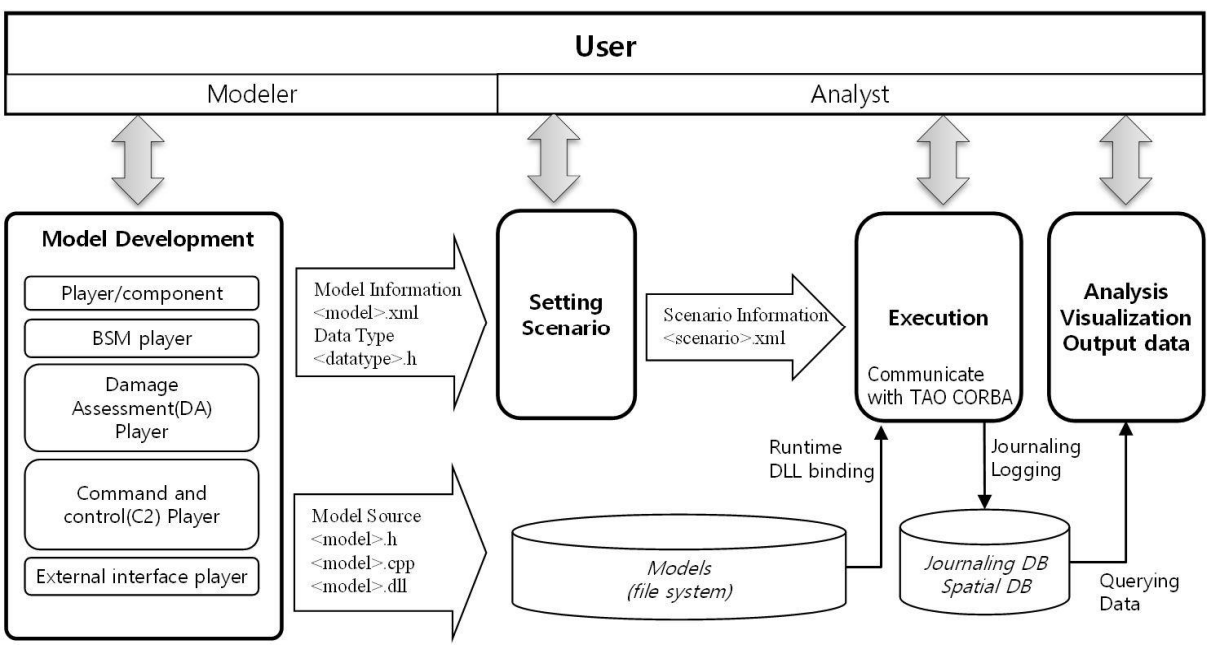

Fig. 4. Conceptual component & connection view of AddSIM

3.2.4 시험용이성(Testability)

시험용이성을 증대하기 위하여 시스템의 상태를 제어하고 관찰할 수 있게 하는 방안으로 시뮬레이션 정보를 기록하고 재생하는 기능을 제공한다. 즉, 각종 이벤트나 사용자 요구에 따른 로그 전시 기능, 저널링을 통하여 시뮬레이션 종료 후 결과를 직관적으로 볼 수 있는 기능을 제공한다. 타임라인 그래프, 저널링 속성에 대한 결과 그래프, 텍스트 파일 출력을 통해 시뮬레이션 과정에 대한 상태를 간접적으로 관찰할 수 있으며, 특히 미 해군연구소에서 무료로 배포하는 3차원 결과 가시화도구인 SIMDIS로 결과를 출력하여 직관적인 가시화 기능을 제공한다. 시스템의 복잡도 감소를 위한 전략으로는 무기체계 소프트웨어 개발관리 매뉴얼[15]에서 추천하는 소스코드 메트릭인 순환복잡도 20 이하, 호출 층수 6 이하, 입력변수 개수 8 이하, 호출함수 개수 8 이하, 피호출 개수 10 이하, 실행코드 수 200 이하를 만족하였다.

3.2.5 사용편의성(Usability)

사용편의성을 위해 운용도구를 시뮬레이션 엔진과 분리하여 사용자의 입력이 시뮬레이션 엔진의 오류로 전파되는 여지를 차단하였으며, 운용도구에서도 사용자의 실수가 시스템에 미치는 영향을 분리하기 위하여 운용도구 상 모든 사용자 입력에 대한 cancel, undo, pause/resume이 가능토록 한다. 또한 시뮬레이션 결과는 자동으로 실행 시간별 파일 시스템 위치에 저장되어 기존의 시뮬레이션 결과를 손실할 가능성을 최소화하였다. 그리고 하부 컴포넌트의 수정 시 변경내용에 영향을 받을 수 있는 상위 컴포넌트들을 식별하여 사용자에게 제시함으로써 개발의 안정성을 제고하였다.

3.2.6 모델 개발편의성

앞 절에서 소개한 품질 속성 외에도 AddSIM은 사용자에게 친숙한 인터페이스를 제공하기 위하여 운용도구 설계에 다양한 노력을 기울였으며, 버전 3.0까지 개발되면서 지속적으로 운용도구 개선에 노력하였다. 특히 사용자가 소스코드를 직접 코딩해야 하는 점을 감안하여 운용도구에서 설정한 모델 정보를 바탕으로 골격코드를 자동 생성하고, 이를 통합개발환경인 Visual Studio(C++)와 연결하여 작업할 수 있는 체계를 구축하였다. 자동 생성된 골격코드는 사용자가 입력한 부분과 운용도구가 자동 생성한 코드의 영역을 분리하여 사용자 코드를 보호할 수 있는 방법을 제공한다.

4. 결 론

AddSIM의 소프트웨어 아키텍처는 시뮬레이션 엔진을 포함한 무기체계 모델 운용 시스템과 무기체계 모델을 분리하는 구조를 택하여 시뮬레이션 엔진이 안정적으로 실행됨을 보장하며, 분석 목적에 따른 변경사항을 유연하고 확장성 있게 반영할 수 있도록 설계되었다. 또한 콘텐츠가 되는 무기체계 모델을 위한 표준화된 형태의 기본체계모델 아키텍처를 설계하여 지속적이고 안정적인 무기체계 모델의 축적에 기여할 수 있다.

본 논문에서는 AddSIM 아키텍처를 소프트웨어 품질 속성측면에서 검토해 보았다. 발생 가능한 오류 유형을 분석하고 사용자의 오류를 검증할 수 있는 기능을 적용하여 가용성을 증대하였으며, 사용자 활용 단계 별 소프트웨어 모듈의 독립성을 확보하여 변경용이성을 제고하였다. 시뮬레이션 엔진의 실행속도 증대 및 다중 컴퓨터자원을 효율적으로 사용하는 기법을 활용하여 성능 품질을 높였다. 시험용이성에 관하여는 시뮬레이션 결과를 신속 정확하게 파악할 수 있도록 설계하여 시험결과를 쉽게 확인할 수 있게 하였다. 또한 버전 3.0까지 개발이 진행되면서 지속적으로 사용자 편의성을 제고할 수 있었다.

이렇게 확보된 AddSIM 소프트웨어 아키텍처와 기본체계모델을 기반으로 향후 국방M&S체계 개발 시 필요한 무기체계 모델을 신속하게 확보하고. 특화된 사용자 인터페이스를 개발함으로써 개발 위험을 감소시킬 수 있으리라 판단된다. 또한 표준화된 국방M&S 모델들 간의 데이터 호환성, 미래 신규 무기체계 모델 확보 용이성이 증대될 것으로 판단된다.

References

[1] H.-S. Oh, D. Kim, “Current Status and Plan of Defense Modeling and Simulation Technologies,” Defense Science and Technology Plus, Vol. 242, 4th Quarter 2018, Agency for Defense Development, pp. 91-100, 2018.

[2] T. Lee, S. Lee, S. Kim, and J. Baik “A Distributed Parallel Simulation Environment for Interoperability and Reusability of Models in Military Applications,” Defence Science Journal, Vol. 62, No. 6, pp. 412-

419, Nov. 2012.

[3] H.-S. Oh, S. Park, H. Kim, T. Lee, S. Lee, D. Kim, O. Paek, J. Park, “AddSIM: A New Korean Engagement Simulation Environment Using High Resolution Models,” Proceedings of the 2014 Winter Simulation Conference, Savannah, USA., pp. 2942-2953, 2014.

[4] H.-S. Oh, D. Kim, S. Lee, Y. Rhie, S. Lee and S. Min, “Developing Weapon Component Model Library,” Proceeding of 12th IAMSEC, Deajeon, p. 76, 2019.

[5] B. McCauley, J. Hill, P. Gravitz, and D. McFarland, “JMASS98-engagement Level Simulation Framework++,” in Simulation Interoperability Workshop, 2000.

[6] P. Faye, Joint Modeling and Simulation System (JMASS) Program Brief in 10th Annual Executive Forum on Modeling and Simulation, 2001.

[7] J. Steinman and H. Doug, “Evolution of the Standard Simulation Architecture,” Overview of the full SIW Paper in Modelling and Simulation. Soc. Model. Simul. Int., Vol. 3, No. 2, 2004.

[8] J. Steinman, “The WarpIV Simulation Kernel”, 19th Workshop on Principles of Advanced and Distributed Simulation, IEEE Computer Society, pp. 161-170, 2005.

[9] J. Steinman, G. Lammers and M. Valinski, “A Proposed Open Cognitive Architecture Framework,” Proceeding of 2009 Winter Simulation Conference, Orlando FL. USA, pp. 1345-355, 2009.

[10] T. S. Kim, H. J. Chang, J. M. Lee, and K. S. Lee, “A Modeling & Simulation Engine for Analyzing Weapons Effectiveness : Architecture,” Journal of the Korea Society for Simulation, Vol. 19, No. 2, pp. 51-62, June 2010.

[11] M. D. Petty, J. Kim, S. E. Barbosa and J-J. Pyun, “Review Article: Software Frameworks for Model Composition,” Modelling and Simulation in Engineering, Vol. 2014, Article ID 492737, Hindawi Publishing Corp., 2014.

[12] L. Bass, P. Clements and R. Kazman, Software Architecture in Practice, 3rd. ed., Addison Wesley, 2013.

[13] J. Yang and C. Choi, “Dynamic Verification Methodology of User Code in AddSIM Environment,” Journal of the Korea Society for Simulation, Vol. 28, No. 1, pp. 41-47, March 2019.

[14] The ACE ORB(2016) Web Page, “Real-time CORBA with TAO,” http://dre.vanderbilt.edu/~schmidt/TAO.html (Accessed July 7, 2020).

[15] DAPA, Development & Management Manual for Weapon System Softwares, 2018.