JANT 한국항행학회논문지 Journal of Advanced Navigation Technology

항공 우주 해상

J. Adv. Navig. Technol. 24(3): 184-191, Jun. 2020

항공용 SIL에 적용 가능한 이벤트 기반 모델링 및 시뮬레이션 방법

Event-Driven Modeling and Simulation Method Applicable to Avionics System Integration Laboratory

신주철*·서민기·조연제·백경훈·김성우
LIG 넥스원 항공전자연구소

Ju-chul Shin* · Min-gi Seo · Yeon-je Cho · Gyong-hoon Baek · Seong-woo Kim
Avionics R&D Lab, LIG Nex1, Daejeon, 34115, Korea

[요 약]

항공용 SIL은 항공전자시스템의 통합 및 검증에 사용되는 통합시험환경이다. 항공용 SIL에는 여러 가지 제약으로 인해 항공기에 탑재되는 장비를 직접 연동할 수 없을 때 장비의 소프트웨어 모델이 필요하다. 지금까지 항공기 개발에 적용한 항공용 SIL의 소프트웨어 모델은 표준화된 방법 없이 일반적인 소프트웨어 개발 방법의 적용으로 재사용이 어려워 소프트웨어 모델 재사용을 위한 프레임워크의 필요성이 제기되었다. 이러한 항공용 SIL 모델의 표준화된 모델링 방법을 위해 DEVS (discrete event system specification) 형식론을 채용하였다. DEVS 형식론은 이벤트 구동 (event-driven) 알고리즘이며 이는 기존의 항공용 SIL에 적용되는 절차적이고 반복적인 알고리즘과 어울려 동작하기 힘들다. 이에 본 논문에서는 항공용 SIL 모델의 특징과 기존 방식이 가지는 한계를 보완하고 모델의 재사용성을 극대화할 수 있는 이벤트 기반의 모델링 방법과 실시간 시뮬레이션 방법을 제안한다.

[Abstract]

Avionics System Integration Laboratory is the integrated test environment for integration and verification of avionics systems. When real equipment can not be used in the laboratory for various reasons, software models should be needed. Because there hasn't been any standardized method for the models so that it is difficult to reuse the developed models, the need for a framework to develop the avionics software models was emerged. We adopted DEVS(discrete event system specification) formalism as the standardized modeling method for the avionics software models. Due to DEVS formalism is based on event-driven algorithm, it doesn't accord a legacy system which has sequential and periodic algorithms. In this paper, we propose real-time event-driven modeling and simulation method for SIL to overcome these restrictions and to maximize reusability of avionics models through the analysis of the characteristics and the limitations of avionics models.

Key word : System integration laboratory (SIL), Discrete event system specification (DEVS), Event-Driven, Simulation algorithm.

https://doi.org/10.12673/jant.2020.24.3.184

Received 3 June 2020; **Revised** 4 June 2020
Accepted (Publication) 27 June 2020 (30 June 2020)

***Corresponding Author; Ju-chul Shin**

Tel: +82-42-718-3577
E-mail: juchul.shin@lignex1.com

Ⅰ. 서 론

항공기 체계 및 부체계의 통합 및 검증 환경인 통합시험환경(SIL; system integration laboratory)은 항공기를 구성하는 다양한 탑재장비를 통합하여 비행 시험 전 실험실 환경에서 항공기 체계의 요구사항을 검증할 수 있는 환경을 제공한다. 그러나 실험실 환경에서는 비행기의 동역학 데이터와 각종 센서류의 데이터 획득이 어렵고 장비를 가용할 수 없는 상황이 발생한다. 또한, 실제 탑재장비를 이용하는 시험은 고의적 고장 발생 모의가 힘들기 때문에 비정상 상황을 가정하는 요구사항에 대한 검증이 어렵다. 이로 인해 항공용 SIL에서는 가상의 비행 및 센서 데이터를 모의할 수 있고 정상 상태 뿐 아니라 고장을 모의할 수 있는 탑재장비 LRU(line-replaceable unit)의 소프트웨어 모델이 필수적이다[1].

국내 항공용 SIL 개발 사례를 살펴보면, 항공기 개발 사업 별로 SIL의 하드웨어/소프트웨어 환경 및 요구도가 결정되었다. 모델은 SIL HW/SW 환경 하 In-house 방식으로 개발되었고, 짧은 개발 일정 내 진행하다보니 타 개발 사업에 적용할 수 있는 이식성을 고려하기 어려웠었다. 점차 국내의 유무인기 사업이 진행되어 SIL 개발 노하우가 축적되면서 생산성/품질 향상 필요성이 대두되었고, SIL의 개발 생산성 향상을 위한 개선 방안으로 모델 표준화 및 이식성에 초점을 둔 항공용 SIL의 비행체 탑재장비 모델 프레임워크 (ASMF; avionics simulation model framework)가 논문을 통해 제안되었다[2].

본 논문에서는 ASMF에서 제안하는 항공용 SIL 탑재장비 모델(ASM, avionics simulation model)의 DEVS 형식론 기반 모델링 방법과 실시간 컴퓨터에서 이벤트 구동(event-driven) 방식으로 ASM을 모의할 수 있는 시뮬레이션 방법을 제안한다.

Ⅱ. ASM의 특성 및 구성

2-1 ASM의 설계 원칙과 구성

모델링은 대상 시스템을 모델링 목적에 부합하는 추상화 수준에 맞게 각기 다르게 표현할 수 있다. 항공용 SIL에서 요구되는 모델의 기준은 LRU마다 다르며 크게 ICD(interface control document) 모델과 ICD 모델의 기능을 포함하는 동작(behavior) 모델로 구분할 수 있다.

모든 LRU 모델은 ICD 정보를 포함한 ICD 모델로 각 LRU간 통신, I/O 프로토콜을 모의하고 관련된 결함을 모의한다. ICD 모델의 기능 외에 계산 로직이 필요한 모델은 동작 모델로 상태기계, 수식의 계산, 동작 모드의 변경과 같은 로직이 구현된다. 동작 모델 중 GPS/INS, RA(radio altimeter), ADC(air data computer)등 동적인 비행모의 데이터가 필요한 모델은 동적(dynamic) 모델로 모의 중 비행 모델과 데이터를 송수신한다[3].

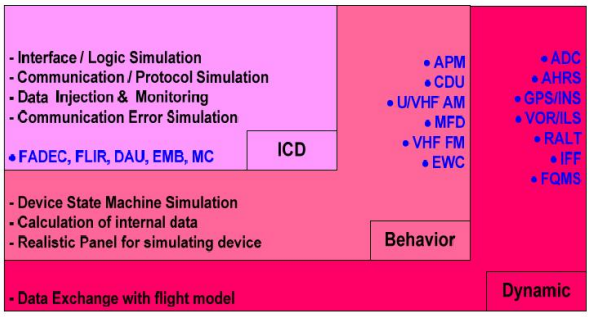


그림 1. 항공전자 탑재장비(LRU) 모델의 분류
Fig. 1. Classification of avionics LRU models

대부분의 항공용 SIL 모델은 동작 모델이다. 동작 모델은 ICD 모델의 기능인 인터페이스 입출력 기능과 계산 로직으로 두 모듈로 구분할 수 있다. 소프트웨어 공학 이론에서 보면 모듈 간 결합이 강할수록 의존성으로 인해 모듈의 재사용이 어렵고 재사용성을 강화하는 방법으로 약한 결합(loose coupling)을 강조한다[4]. 실제로 기존 항공용 SIL 모델 개발 사례를 보면 하나의 LRU 모델에서 ICD 모델 기능과 계산 로직의 결합이 강할수록 로직의 재사용이 어려웠다. 예를 들어 A 항공기 개발 사업의 SIL 에서 개발한 RA 모델을 ICD와 내부 로직이 강한 결합으로 개발되었다면, B 항공기 개발 사업 SIL에서 RA 모델의 ICD가 다른 경우 고도를 출력하는 비슷한 기능을 하는 LRU 모델이지만 재사용하기 힘들다. ASMF는 모델의 재사용성을 극대화하기 위해 두 모듈을 약한 결합으로 동작할 수 있는 방법을 제공한다.

그림 2에서 ASM의 연동블록은 ICD 모델에 해당하고 DEVS 모델은 동작 모델의 계산 로직에 해당한다. 이 두 모듈은 포트(port)와 연동 속성(interface attribute)을 매개로 작동하게 되어 ICD의 변경이나 계산 로직의 변경이 발생하는 경우 서로의 변경 영향성을 최소화하고 재사용성이 개선될 수 있다. ASMF의 통합개발환경은 연동블록과 DEVS 모델을 모델링할 수 있는 GUI를 제공하고 그 결과를 빌드하면 XML파일과 실행파일의 형태로 생성한다.

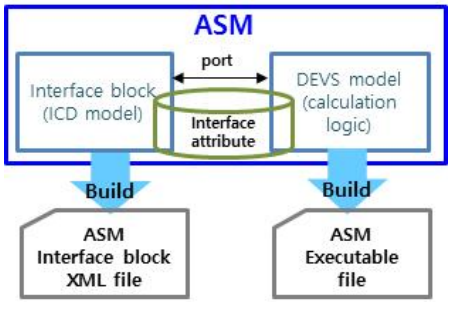


그림 2. ASM 구성 요소와 산출물
Fig. 2. ASM Components and output

2-2 DEVS 형식론 기반의 ASM 계산 로직

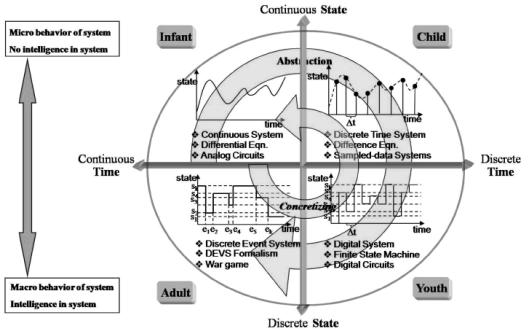


그림 3. 시스템의 이론적 분류
Fig. 3. Dynamic system types classified by system theory

시스템의 입력, 출력, 상태 변수를 정의한 후 상태 변수가 시간에 따라 변하는 규칙을 가지는 모델을 수학적 방정식 형태로 기술하는 틀을 모델링 형식론이라 한다. M&S(modeling and simulation) 이론에서는 모델의 상태 변수와 시간이 가질 수 있는 값의 범위를 각각 연속적인(실수) 값과 이산 값으로 나누고 이에 따라 시스템을 분류한다. 따라서 시스템 종류는 상태 변수 및 시간 값의 표현 방법을 조합한 네 가지가 존재하게 되며 다음 그림 3은 각각의 학술 명, 수학적 모델링 형식론 및 예를 나타낸다[5].

실시간 시스템을 모의하는 ASM은 연속적인 시간인 실시간 흐름에 따른 이산 값의 상태 변수를 가지는 모델로 위 그림의 3사분면에 해당하고 이를 모델링하는 대표적인 형식론은 DEVS 형식론이다. DEVS 형식론의 수학적인 이론은 참고 문헌을 통해 접할 수 있다[6]. ASMF에서 DEVS 형식론을 채용한 큰 이유는 DEVS 형식론이 수학적 표현을 사용하지 않고도 관찰자 관점에서 직관적 개념인 상태도(그림 4)와 블록도(그림 5)로 모델링이 가능하고 객체지향 프로그래밍 언어로 구현이 용이하다는 점이다[7].

ASMF는 통합개발환경을 통해 GUI 기반 DEVS 형식론 모델링하는 기능과 모델을 C++ 소스코드로 자동 생성하여 실행 파일로 빌드하는 모델주도개발(model-driven development) 도구 기능을 제공한다.

2-3 ASM 연동블록(ICD 모델)

연동블록은 ASM의 ICD 모델로 각 ASM은 연동블록을 통해 SIL의 다른 구성 요소(다른 ASM 또는 실제 탑재장비)와의 연동을 정의한다. ASMF의 통합개발환경은 GUI 기반 연동블록 개발 기능을 제공하고 연동블록을 실행하는 시뮬레이션 엔진의 설정을 위한 XML 포맷의 파일을 자동 생성한다.

각 연동블록의 정의는 세 단계로 이루어진다. 첫째, 인터페이스에 대한 스펙을 정의한다. 이는 ASM이 사용할 하드웨어에 대한 정의로 인터페이스 타입, 송수신 채널, 속도, 주소 등 물리

표 1. 연동블록 XML 구성요소 계층 구조
Table 1. XML element hierarchy for interface block

element	child element	attribute
interface channel information (Lv. 1)	message information	interface type, channel id, address information, etc.,
message information (Lv. 2)	data link information	header length, header value, endian, period, rx/tx, monitor flag, etc.,
data link information (Lv. 3)	-	data offset, length, linkage type(port, interface attribute), data types(C++ underlying types, struct, array), etc.,

적인 인터페이스채널에 대한 스펙이다. 둘째, 위에서 정한 인터페이스를 통해 송수신할 메시지를 정의한다. ICD는 인터페이스 채널로 송수신하는 메시지를 기술하고 있고, 그에 따라 메시지 정보인 메시지 헤더, 길이, 송수신 주기 등을 정의한다. 셋째, 메시지와 DEVS 모델 내부와의 데이터 교환을 정의한다. ICD 메시지는 의미 단위를 갖는 여러 데이터의 집합이다. 수신된 메시지는 이 단계에서 정의된 데이터 교환 규칙에 따라 분해되어 DEVS 모델로 입력되고, 반대로 DEVS 모델에서 계산된 결과 데이터는 송신 메시지로 합성된다. 통합개발환경은 연동블록의 세 가지 정의를 XML 형태로 생성한다. 연동블록 XML의 구성 요소(element)와 하위 속성은 기본적으로 표 1과 같은 정보를 포함한다.

ASMF의 시뮬레이션 엔진은 이 XML 파일을 로드하여 ASM 모의전 인터페이스를 초기화 하고 메시지 송수신을 수행하며 DEVS 모델과의 데이터를 교환한다.

Ⅲ. ASM 모델링과 시뮬레이션 방안

3-1 DEVS 형식론 기반 모델링

이번 장부터는 ASMF의 핵심에 해당하는 DEVS 모델링, 시뮬레이션 알고리즘, 연동블록과 DEVS 모델 간 데이터 교환에 대한 방법을 기술한다. DEVS 형식론은 원자 모델(atomic model)과 결합 모델(coupled model)의 두 모델로 계층적 모델을 표현한다. 두 모델을 이용하여 ASM을 다이어그램으로 모델링하는 방법은 다음과 같다.

원자 모델은 DEVS 형식론을 구성하는 단위 모델로서 ASM의 단위 기능을 설계할 수 있다. 상태도는 하나의 원자 모델에 대한 다이어그램이다. 하나의 상태도는 원자 모델의 상태 변화를 나타낸다. 상태도의 실선은 포트를 통한 이벤트 발생으로 인한 상태 천이(외부 천이, external transition)를 나타내고, 점선은 시간의 흐름에 따른 상태 천이(내부 천이, internal transition)를 나타낸다. 즉, 시간의 흐름 또는 외부 데이터(포트) 입력이라는 두 종류의 이벤트가 원자 모델의 상태를 변화 시킨다. 내부 천

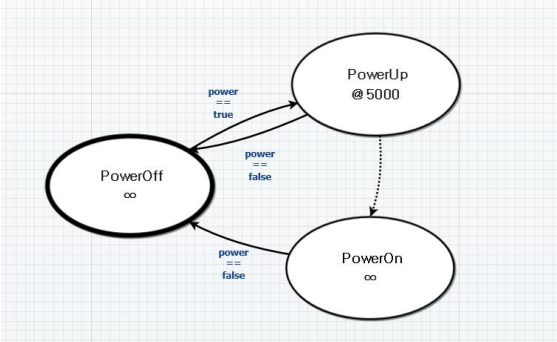


그림 4. PBIT 기능 원자 모델의 상태도 예시
Fig. 4. An example of a state diagram for a PBIT(power-up built-in-test) atomic model

이가 발생하면 원자 모델은 데이터를 포트를 통해 출력할 수 있고 해당 출력 포트와 연결된 다른 원자 모델은 외부 천이가 발생한다. ASMF 통합개발환경은 원자 모델의 소스코드를 시뮬레이션 엔진에서 제공하는 클래스 라이브러리 중 원자 모델 클래스를 상속받아서 생성한다. 하나의 원자 모델은 하나의 클래스로 생성된다. 라이브러리는 그 원자 모델에서 정의한 포트, 연동속성, 멤버변수를 모두 멤버 변수처럼 취급할 수 있는 메커니즘을 제공하고 ASM 개발자는 계산이 필요한 경우 상태천이 시에 호출되는 함수에 유저 코드를 추가하여 계산 기능을 구현할 수 있다.

그림 4의 상태도는 PBIT(power-up built-in-test)를 처리하는 원자 모델의 상태도를 ASMF 통합개발환경으로 모델링한 예이다. 초기에 PowerOff 상태에서 무한히 대기한 후 power 입력 포트를 통해 전원이 인가(true)되면 PowerUp 상태로 외부 천이하여 초기화 시간(5000ms)만큼 대기 후 PowerOn 상태로 내부 천이하며 PBIT 결과를 포트로 출력하며 무한히 대기한다. PowerUp 상태와 PowerOn 상태에서는 power 입력 포트를 통해 전원이 차단(false)되면 초기 상태로 외부 천이한다.

결합 모델은 하위에 원자 모델(그림 4 rectangle block)과 다른 결합 모델(그림 5 rounded-rectangle block)을 가질 수 있다. ASMF의 블록도는 하나의 결합 모델에 대한 다이어그램이다. 즉, 하나의 결합 모델은 블록도를 통해 하위 모델들의 관계를 나타낸다. 이 블록도를 보면 결합 모델이 어떻게 하위 모델 사이 또는 하위 모델과 결합 모델 사이 포트(블록 외곽 노드)를 통해 서로 입/출력 데이터(이벤트)를 주고받으며 상호 연동하는지 알 수 있다. ASM 은 하나의 최상위 결합 모델을 가지고 있으며 그 안에 원자 모델들과 결합 모델들을 기능 단위로 구분하여 블록도로 설계할 수 있다. 그림 5은 결합 모델의 블록도로 그림 4의 상태도를 포함하는 PBIT 원자 모델을 포함한 두 개의 원자 모델과 하나의 결합 모델로 구성된 예이다.

이러한 계층 구조를 통한 기능 분할 설계는 모델의 확장(scalability)을 용이하게 하고 모듈화(modularity)를 향상시킨다. 즉, 기존의 모델링에서 원자 모델과 결합 모델들을 추가 또는 제거하여 기능을 확장하거나 제거하기 용이하고, 블록단위

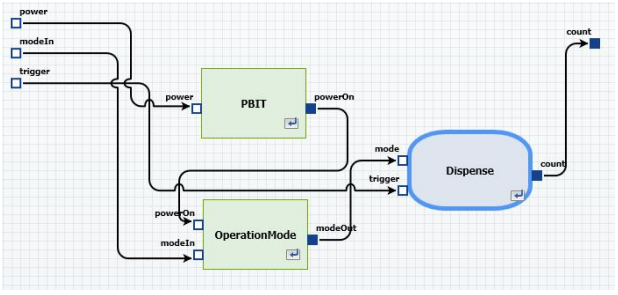


그림 5. ASM 최상위 결합 모델의 블록도 예시
Fig. 5. An example of an ASM top coupled model

로 기능을 캡슐화하여 모듈을 재사용하기 쉽다. 또한, 다이어그램을 통하여 표준화하는 것이므로, 개발자간 원활한 소통에 도움이 된다.

3-2 시뮬레이션 엔진 구조

본 논문에서 소개하는 시뮬레이션 엔진은 ASMF의 블록도와 상태도 기반의 모델링 결과를 소스코드로 생성하기 위한 기반 클래스 라이브러리와 그 기반 클래스로부터 파생된 클래스의 객체를 실행하는 시뮬레이터 그리고 ASM의 연동블록이 정의한 XML 파일 설정으로 인터페이스를 운용하고 이를 통해 송수신하는 메시지의 데이터를 교환하는 기능을 포함한 미들웨어를 지칭한다. ASMF의 시뮬레이션 엔진의 구조는 다음 그림 6와 같다.

그림 6의 HAL(hardware abstract layer)과 OSAL(operating system abstract layer) 는 각각 하드웨어 장치와 운영 체제에 대한 추상화 계층으로 향후 하드웨어와 운영 체제의 변경에도 상위 계층을 변경 없이 사용할 수 있도록 하는 일종의 어댑터이다. 각 인터페이스의 디바이스 드라이버와 OS 자원(태스크, 세마포어, 타이머 등)에 대한 API를 상위 계층에 제공한다. 시뮬레이션 제어 모듈(Simulation Controller)은 OSAL을 통해 각 ASM의 OS 자원을 초기화하고 시뮬레이션을 시작하거나 중지한다.

인터페이스 운용 모듈(Interface Executor)은 연동블록 XML 파일에 정의된 ICD 정보를 바탕으로 HAL을 이용해 인터페이스 채널과 메시지를 초기화 한다. 시뮬레이션이 시작되면 인터페이스 운용 모듈은 송수신 주기에 맞춰 메시지 버퍼를 HAL을 통해 송수신하고 메시지 수신 인터럽트를 처리한다. 수신된 메시지는 메시지 교환 모듈(Message Exchanger)에 전달하고 송신할 때는 메시지 교환 모듈로부터 메시지를 획득한다.

메시지 교환 모듈은 인터페이스 운용 모듈이 수신한 메시지를 분해하여 연결된 DEVS 모델 내부에 전달하고, 반대로 송신할 메시지는 DEVS 모델 내부로부터 합성하여 인터페이스 운용 모듈로 전달한다. DEVS 모델 내부와의 데이터 교환은 포트 또는 연동속성이라는 두 타입의 데이터 교환 수단을 이용하고, 자세한 내용은 3-4 장에서 기술한다.

시뮬레이터(simulator)는 DEVS 모델을 시뮬레이션 한다. 메

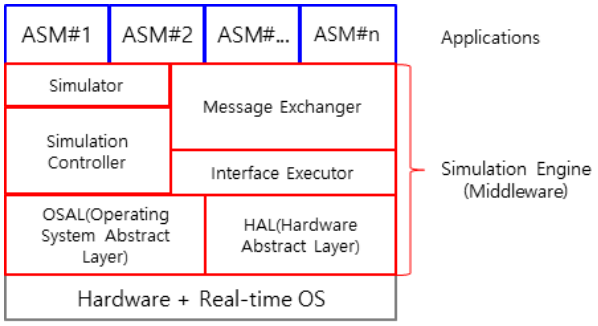


그림 6. ASMF의 실시간 시뮬레이션 환경 계층 구조
Fig. 6. Layer architecture of real-time simulation environment for ASMF

시지 교환 모듈을 통해 DEVS 모델의 외부 입력 이벤트 또는 타이머에 의한 시간 전진 이벤트가 발생하면 이벤트를 최상위 결합 모델로 전달한다. 시뮬레이터는 다수의 ASM과 연동하여 이벤트를 전달하고 수신한다.

3-3 시뮬레이터 알고리즘

이론적으로 시뮬레이션이란 모델링 형식론으로 모델링한 모델의 수학적 방정식을 푸는 과정이다[5]. 시뮬레이터 모듈의 알고리즘은 ASM DEVS 모델의 수학적 해법을 프로그래밍하여 구현한 것이다. 한 문제에 대한 해법이 다수가 존재하듯이 DEVS 모델 시뮬레이션 방법은 여러 가지가 있고 ASMF에서 적용한 시뮬레이션 알고리즘을 간단히 도식화 하면 다음 그림 7과 같다. [2]

DEVS 형식론의 원자 모델과 결합 모델의 계층적 모델 구조는 데이터 구조 중 트리 구조로 구현될 수 있다. 트리의 단말 노드는 원자 모델이고 내부 노드는 결합 모델이다. 트리의 루트 노드는 최상위 결합 모델이라 할 수 있다.

그림 7의 시뮬레이터는 최상위 결합 모델과 모델 외부를 연동하는 인터페이스 역할을 한다. 시뮬레이터는 메시지 교환 모듈에서 입력 데이터를 수신하면 최상위 결합 모델의 입력 포트로 전달하고 시뮬레이션 제어 모듈로부터 수신한 시간 전진(time advance)이벤트를 최상위 결합 모델로 전달한다. 이 두 종류의 이벤트는 결합 모델의 블록도에서 정의한 모델 간 포트 연결과 원자 모델의 상태도의 시간 전진에 따른 내부 천이 스케줄에 따라서 단말 노드인 원자 모델을 동작시킨다.

최상위 결합 모델은 입력 포트로 데이터를 수신하면 연결된 하위 모델로 전달하고 최종적으로 단말 노드인 원자 모델이 수신하면 외부 천이가 발생하여 상태가 천이된다.

그림 6의 시뮬레이션 제어 모듈은 설정된 단위 시간마다 시뮬레이터로 시간 전진 이벤트를 전달하면 시뮬레이터는 최상위 결합 모델로 시간 전진 이벤트를 전달한다. 단위 시간은 1/1000초(millisecond) 단위로 설정할 수 있다. 모든 결합 모델은 하위 모델들의 다음 내부 천이 중 가장 빠른 시간을 가지고 있다. 결합 모델은 시간 전진 이벤트를 수신하면 해당 시간에

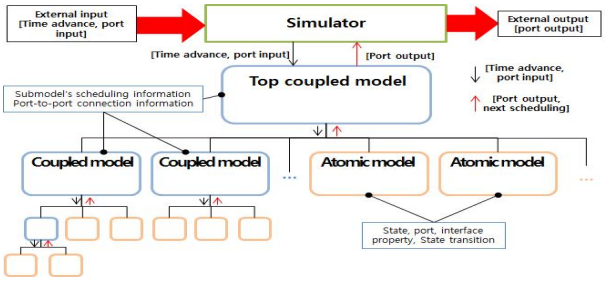


그림 7. 시뮬레이션 알고리즘 실행개념
Fig. 7. Execution concepts of the simulation algorithm

발생하는 내부 천이가 있는 경우 그 하위 모델로 이벤트를 전달하고 없는 경우 무시한다. 하위 모델로 전달된 시간 전진 이벤트는 결국 스케줄 대상이 되는 단말 노드인 원자 모델이 수신하고, 원자 모델은 내부 천이를 수행한다. 원자 모델의 내부 천이가 발생하면 출력 포트를 통해 데이터를 출력하고 출력 포트와 연결된 모델로 이벤트가 전파되어 또 다른 외부 천이가 발생한다. 최상위 결합 모델에서 포트의 출력이 발생하면 시뮬레이터는 메시지 교환 모듈로 포트의 데이터를 출력한다.

외부 천이 또는 내부 천이가 발생한 후 원자 모델은 천이된 상태의 다음 내부 천이 시간을 상위 결합 모델로 전달하여 그 결합 모델이 다음 내부 천이를 처리할 수 있도록 스케줄링을 변경한다. 시뮬레이터 알고리즘은 이와 같이 원자 모델의 외부 천이와 내부 천이를 유발하는 이벤트를 시간 순서대로 처리하여 ASM의 내부 로직을 동작시킨다.

3-4 데이터 교환 메커니즘

ASM의 ICD 모델(연동블록)과 DEVS 모델 간의 데이터 교환은 그림 2와 같이 포트와 연동속성 메커니즘을 이용하여 느슨한 결합을 이룬다. 포트를 이용한 데이터의 교환은 3-3에서 설명된 알고리즘과 같이 수신된 메시지 중 DEVS 모델의 최상위 결합 모델과 연결된 데이터는 메시지 교환 모듈에서 분해되어 시뮬레이터로 전달된다. 전달된 데이터는 시뮬레이터가 최상위 결합 모델의 포트로 전달하고, 그 이후는 ASM 블록도에서 정의한 모델(원자/결합) 간 포트 연결에 따라 전달된다. 이러한 포트 데이터는 원자 모델의 외부 천이를 유발한다. 따라서 ICD의 모든 데이터를 포트로 연결한다면 상태를 변경시키지 않아도 되는 이벤트를 처리해야 하고 블록도에서 포트와 연결선이 증가하여 모델링 복잡성이 커진다.

표 2. 데이터 연결 구분 기준의 예시
Table 2. Data linkage examples

linkage type	examples
port	power input, trigger signal, command(mode, BIT, enable/disable, query, erase, etc.,)
interface attribute	flight simulation data(position, velocity, attitude), target information, BIT result, LRU status simulation data, etc.,

이벤트 처리는 프로그램 내부적으로 함수의 호출로 구현되어 있어서 많은 이벤트 처리는 많은 함수 호출이 발생하고, 결국 필요 없는 부하가 되어 성능을 저하시킨다. ASMF는 성능과 모델링 복잡성을 개선하기 위해 연동속성이라는 개념을 도입하였다. 연동속성은 일종의 공유 메모리로 함수 호출의 오버헤드 없이 ICD 모델의 입출력 데이터와 DEVS 모델 내부의 데이터가 같은 값을 공유할 수 있도록 한다. 연동속성은 공유 메모리를 이용한 원자 모델 간 데이터 공유도 가능하기 때문에 포트 연결이 필요 없는 원자 모델 사이의 블록도의 모델링 복잡성을 개선한다.

포트와 연동속성을 구분하여 설계하려면 ICD의 분석이 선행되어야 한다. ICD의 데이터를 분석해보면 입력 데이터는 명령형 데이터(command data)와 참고형 데이터(reference Data)로 구분할 수 있다.

모드 변경, 데이터 출력 요청 등의 명령형 데이터는 원자 모델의 상태 천이를 유발하는 데이터로 이는 최상위 결합 모델의 포트를 통해 입력되어 특정 원자 모델의 상태를 천이시킬 수 있다. 반대로 원자 모델의 상태 천이와는 관계없고 계산을 위한 참고형 데이터는 연동속성으로 원자 모델 내부의 데이터와 ICD의 송수신 메시지 간 데이터를 공유한다. 연동속성과 연결된 데이터의 경우 메시지 교환 모듈은 그 데이터가 포함된 메시지를 수신하면 분해하여 공유 메모리에 저장한다. 원자 모델은 연동 속성을 정의하고 연동블록에 그 연동 속성을 메시지에 ICD에 맞게 매핑하면 수신한 데이터가 반영된 공유 메모리를 읽어서 계산에 활용할 수 있다.

참고형 데이터는 계산을 위한 참고 데이터로 원자 모델의 상태를 천이시킬 필요는 없지만 계산 로직을 수행하는 기초 데이터가 된다. 예를 들어, 음성통신 중계 장비인 U/VHF 무전기 ICD 메시지를 보면 매뉴얼 주파수 변경 명령과 변경할 주파수에 대한 메시지 정의가 있다. ICD의 매뉴얼 주파수 변경 명령에 해당하는 메시지의 해당 비트는 ASM의 포트와 연결되어 주파수 변경 기능을 모의하는 원자 모델의 상태를 변경시키고, 변경된 상태에서 설정할 주파수 모의에 필요한 데이터는 연동속성을 통해 공유 메모리에 갱신된 값을 사용하게 된다. 이와 같이 ASM은 입력 데이터를 원자 모델의 상태 천이를 기준으로 포트와 연동속성으로 구분하여 정의한다.

ASM의 현재 상태를 출력하는 ICD 출력 데이터의 경우는 모델의 상태 천이에 영향을 미치지 않으므로 포트 연결과 연동속성 연결을 데이터의 성격으로 구분하는 의미가 없다. 출력 데이터의 경우 포트와 연동속성을 메시지의 출력 주기로 구분하여 연결한다. 포트의 출력은 DEVS 모델의 내부 천이의 결과로 발생되는 이벤트로 즉시 외부로 출력되어야 하는 비주기 메시지에 매핑할 수 있고, 주기적으로 송신하는 메시지의 데이터인 경우 연동속성과 연결한다. 인터페이스 운용 모듈에서 주기적인 송신이 필요할 때 메시지 교환 모듈을 통해 연동속성 공유 메모리로부터 원하 모델이 갱신한 데이터를 합성하여 메시지를 구성하고 이를 송신한다.

Ⅳ. 실시간성(timeliness) 확인 방법

항공기의 항전체계는 경성 실시간 시스템(hard real-time system)으로 항전체계의 탑재장비를 모의하는 ASM도 실시간 성능이 보장되어야 하고 시뮬레이션 엔진은 그 방법을 제공해야 한다. 기존 항공용 SIL의 소프트웨어 모델을 위한 레거시 시스템은 절차 지향적 프로그램 언어인 C 언어로 구현되었고 각 모델은 실시간 운영체제의 타이머를 이용한 주기적인 작업을 스케줄링하여 실행하였다.

일반적으로 스케줄링 주기는 군용 항공기 인터페이스인 MIL-STD-1553B의 메시지 스케줄의 마이너 프레임 단위인 20ms의 배수로 결정되었고, 20ms 마다 타이머 인터럽트가 발생하면 모델은 출력, 입력, 계산 작업을 반복한다. 이러한 주기를 단위 시간 프레임이라 하고 시간의 흐름에 따라 소프트웨어는 다음 그림 8과 같은 일련의 동작을 반복한다.

프레임 타이머가 발생하면 먼저 이전 프레임에서 계산된 결과를 인터페이스로 출력하고, 이전 프레임에서 입력된 데이터를 인터페이스로부터 폴링(polling)하는 방식으로 획득한다. 획득한 데이터를 기준으로 필요한 모든 계산을 수행하고 수행된 계산 결과는 다음 프레임에 출력하게 된다. 이러한 레거시 시스템에서 실시간 성능을 보장하기 위해 다음과 같이 모니터링 하였다. 단위 시간 프레임이 시작할 때 이전 프레임의 모든 작업이 종료되었는지 체크하고 이전 프레임의 모든 작업이 종료되지 않았다면(프레임 오버런) 출력되는 계산 결과는 유효하다고 할 수 없음을 시현하거나 처리한다. 예를 들면, 그림 8의 네 번째 프레임에서는 데이터 계산이 다음 프레임 시작 시간까지 끝나지 않아 출력되는 계산 결과의 무결성을 보장할 수 없다고 경고 메시지를 시현하게 된다.

ASMF의 경우 레거시 시스템의 주기적인 일괄처리 방식의 알고리즘이 아닌 DEVS 형식론 기반의 이벤트 구동 알고리즘이며 인터페이스의 운용도 주기적인 폴링 방식 뿐 아니라 이벤트 구동에 효율적인 인터럽트 방식이 가능하다. 따라서 같은 방법으로 실시간성을 확인할 수 없고 ASMF의 시뮬레이션 엔진은 출력 메시지 주기 기준으로 메시지 갱신 여부를 통해 실시간

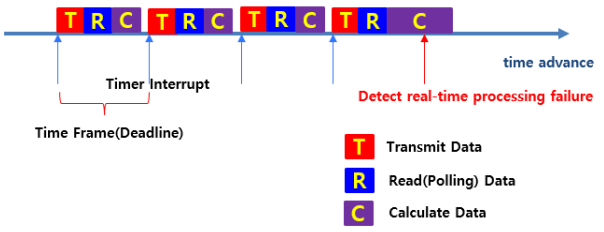


그림 8. 레거시 SIL 시스템의 작업 스케줄링과 실시간성 확인 방법

Fig. 8. Job scheduling concept and real-time simulation monitoring method of a legacy SIL system

성능을 확인한다.

실시간 시스템의 목적은 출력 데이터의 시간 적합성(timeliness) 보장이다. 실시간 시스템은 계산의 결과 뿐 아니라 계산 시점도 원하는 데드라인 이내에 완료되어야하기 때문이다[9]. 따라서, ASMF 시뮬레이션 엔진은 ASM의 출력 메시지 갱신 상태를 다음과 같은 방법으로 모니터링한다.

통합개발환경으로 ASM의 연동블록을 생성할 때 출력 메시지가 주기적 메시지라면 ICD에 정의된 주기를 설정하고, 그 메시지의 데이터 갱신 데드라인 모니터링 여부를 선택하면 연동블록 XML 파일의 메시지 속성에 모니터링 플래그가 설정된다.

시뮬레이션 엔진은 메시지 교환 모듈에서 ASM 실행에 따라 모니터링 플래그가 설정된 메시지의 최근 업데이트 시간을 갱신하고, 인터페이스 운용 모듈은 메시지 출력 주기 사이에 해당 메시지가 갱신이 되었는지 모니터링한다. 메시지가 주기 내에 갱신되지 않는 경우는 출력 결과의 시간 적합성을 보장해주지 못한다. 시뮬레이션 엔진은 주기가 다른 메시지를 각각 자신의 주기에 따라 메시지 갱신이 모니터링하고 주기가 없는 비주기 메시지의 경우 모니터링하지 않는다.

Ⅴ. 결 론

본 논문에서는 항공용 SIL에서 활용할 수 있는 DEVS 형식론 기반 모델링과 이벤트 구동 방식 시뮬레이션 엔진을 제안하였다. ASMF는 항공용 SIL 모델의 특징에 따라 모델을 동작 모델인 DEVS 모델과 ICD 모델인 연동블록으로 분리하였다. 연동블록으로 분리한 ICD 모델은 XML 파일의 형식으로 시뮬레이션 엔진의 인터페이스 운용 및 계산 로직과 데이터 교환을 정의한다. 이렇게 분리된 ICD 모델과 DEVS 모델은 포트와 연동속성이라는 데이터 교환 메커니즘으로 데이터를 교환하여 두 모듈의 변경에 상호 영향성을 최소화 하고 재사용성을 극대화 할 수 있다.

또한, 기존의 레거시 SIL 시뮬레이션 알고리즘의 실시간성 확인 방법은 이벤트 구동 방식의 시뮬레이션 엔진에 적합하지 않다. 이에 ASMF는 이벤트 구동에 적합하도록 출력 메시지의 주기 내에 메시지 갱신이 일어났는지 확인하여 실시간성을 보장한다.

레거시 SIL 시뮬레이션 방법은 인터페이스를 폴링하고 주기적이고 반복적인 스케줄링을 적용하였다. 폴링방식은 인터럽트 방식에 비해 자원을 낭비하고, 주기적이고 반복적인 스케줄링 방법도 필요 없는 계산까지 반복하여 효율성이 떨어진다. 이에 비교하여 ASMF의 이벤트 구동 방식 시뮬레이션 방법은 인터럽트 방식의 인터페이스 운용을 지원하고, 입력된 수신 데이터와 각 원자 모델의 상태에 따라 계산하여 자원을 더 효율적으로 사용한다.

본 논문에서 제안하는 방법으로 항공용 SIL에 DEVS 형식론 모델링과 이벤트 구동 방식의 시뮬레이션 엔진을 적용하면 형식론 없이 개발하던 기존 모델 개발 방식을 표준화기에 용이하고 기존 레거시 시스템의 방법보다 자원을 보다 효율적으로 사용할 수 있으며 통합개발환경을 통한 모델주도개발의 도입으로 모델 개발 기간 단축과 신뢰성 확보가 가능한 실시간 시뮬레이션 환경을 구축이 가능할 것으로 기대한다.

Acknowledgments

본 연구는 방위사업청과 방위산업기술지원센터의 지원(사업명 : 항공용 SIL의 비행체 탑재장비 모델 프레임워크 기술 개발, 계약번호 : UC170001D)하에 수행되었습니다.

References

[1] W. H. Chang, J. S. Park, Y. W. Jo and J. K. Byun, “Verification of hierarchically structured avionics system utilizing multi-mode system integration laboratory,” *Journal of The Korean Society for Aeronautical and Space Sciences*, Vol. 45, No. 11, pp. 998-1005, Nov. 2017.

[2] M. G. Seo, J. C. Shin, S. W. Kim and K. H. Baek, “Development of real time simulation environment based on DEVS formalism applicable to avionics system integration laboratory,” *Journal of Advanced Navigation Technology*, Vol. 23, No. 5, pp. 345-351, Oct. 2019.

[3] M. C. Kim, W. S. Oh, J. H. Lee, J. B. Yim and Y. D. Koo, “Development of a system integration laboratory for aircraft avionics systems,” in IEEE/AIAA 27th Digital Avionics Systems Conference, St. Paul : MN, USA, pp. 5.A.1-1-5.A.1-11, 2008

[4] G. Priyalakshmi and R. Latha, “Evaluation of software reusability based on coupling and cohesion,” *International Journal of Software Engineering and Knowledge Engineering*. Vol. 28. pp. 1455-1485, 2018

[5] T. G. Kim, “Modeling and Simulation Engineering,” *Communications of the Korean Institute of Information Scientists and Engineers,* Vol. 25, No. 11, pp. 5-15, Nov. 2007

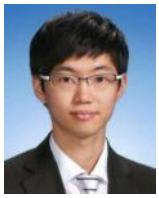
[6] T. G. Kim and B. P. Zeigler, “The DEVS formalism : Hierarchical, modular systems specification in an object oriented framework,” in *Proceedings of the Winter Simulation Conference*, Atlanta : GA, USA, pp. 559-566, 1987

[7] D. L. Eldredge, J. D. McGregor, and M.K. Summers, “Applying the object-oriented paradigm to discrete event simulations using the C++ language,” *Simulation*, Vol. 54, No. 2, pp. 83–91, 1990

[8] A. C. Chow, B. P. Zeigler and D. H. Kim, “Abstract simulator for the parallel DEVS formalism,” in *Fifth Annual Conference on AI, and Planning in High Autonomy Systems,* Gainesville : FL, USA, pp. 157-163, 1994

[9] G. C. Buttazzo, *Hard real-time computing systems : predictable scheduling algorithms and applications, 3rd ed.*, New York, NY : Springer, pp. 4-10, 2011

신 주 철 (Ju-chul Shin)
2009년 2월 : 포항공과대학교 컴퓨터공학과 (공학사)
2009년 1월 ~ 현재 : LIG넥스원 항공전자연구소 선임연구원
※ 관심분야 : 항공전자, 소프트웨어 아키텍처



서 민 기 (Min-gi Seo)
2012년 02월 : 한국항공대학교 컴퓨터공학과 (공학사)
2011년 10월 ~ 현재 : LIG넥스원 항공전자연구소 선임연구원
※ 관심분야 : 항공전자, 내장형 소프트웨어

조 연 제 (Yeon-je Cho)
2016년 02월 : 경희대학교 전자전파공학부(공학사)
2016년 01월 ~ 2018년 03월 : GS ITM 플랜트 ICT 팀 사원
2018년 07월 ~ 현재 : LIG넥스원 항공전자연구소 연구원
※관심분야 : 항공전자시스템, 내장형 소프트웨어

백 경 훈 (Gyong-hoon Baek)
2002년 02월 : 한국과학기술원 기계공학과 (공학사)
2002년 01월 ~ 2016년 05월 : ㈜도담시스템스 수석연구원
2016년 09월 ~ 현재 : LIG넥스원 항공전자연구소 수석연구원
※ 관심분야 : 항공전자, 실시간 시뮬레이션

김 성 우 (Seong-woo Kim)
2002년 08월 : 부산대학교 정보통신공학과 (공학석사)
2002년 10월 ~ 현재 : LIG넥스원 항공전자연구소 수석연구원
※ 관심분야 : 실시간 시뮬레이션 기법 및 시험환경 응용 개발