Journal of KOSSE. (2017. 6) Vol. 13, No. 1 pp. 1-6
DOI: https://doi.org/10.14248/JKOSSE.2016.13.1.001
www.kosse.or.kr

ISSN (print) : 1738-480X
ISSN (online) : 2288-3592

TMO 모델 기반 무장 관리 시스템 소프트웨어 설계

박한솔*

한화탈레스 전문연구원

Software Design of Stores Management System based on the TMO Model

Hansol Park*

Hanwha Thales Senior Engineer

Abstract : A stores management software which is embedded in the stores management system requires high-level reliability and real-time processing. It also required to implement and verify protocols which requires timing constraints to control various weapons. In this paper, we propose design methodology to design a stores management software and its support middleware based on the TMO (Time-triggered Message-triggered Object) model.

Key Words : Stores Management System, Control Software, Distributed Object Oriented Model, TMO Model

Received: July 12, 2016 / **Revised**: June 15, 2017 / **Accepted**: June 19, 2017

* 교신저자 : Hansol Park, hansol80.park@hanwha.com

1. 서 론

무장관리시스템은 다수의 탑재된 무장을 관리해야 하는 시스템이며, 탑재되는 소프트웨어는 고 수준의 신뢰성과 실시간성을 요구한다. 여러 가지 종류의 무장들은 고유의 무장 제어 프로토콜을 갖고 있으며, 해당 프로토콜들은 다양한 시간 제약조건들을 내포하고 있다. 이와 같은 시간 제약조건을 만족시키는 프로토콜을 구현하기 위해서는 실시간 속성을 만족 시킬 수 있는 소프트웨어 기술이 요구되며, 주로 실시간 운영체제의 우선순위 스케줄링 기술과 관련 태스크 제어 기술들을 이용하여 중요한 작업들을 우선 실행시키는 기법들이 사용된다. 그러나 이와 같은 저 수준의 방법론은 응용마다 다르게 적용해야 하고, 문제가 발생하는 경우 수정에 따르는 주변 효과(side effect)가 크게 발생하여 많은 시간과 노력이 요구된다. 시스템 엔지니어링 관점에서도 이와 같은 무장관리 시스템을 효과적으로 설계하고 통합하기 위해서는 소프트웨어 모델기반의 설계 및 구현 기법을 활용하여 많은 비용과 시간을 절약할 수 있다. 본 논문에서는 무장관리 소프트웨어에서 요구하는 실시간 특성과 시간 제약 특성을 고려하고 고 수준의 설계 및 다양한 응용 플랫폼에 공통으로 적용 가능한 방안으로써 분산 실시간 객체 모델인 TMO 모델 기반의 설계 방법론과 이를 지원하기 위한 미들웨어 설계 방안을 제시하고자 한다.

2. 관련 연구

2.1 TMO Model

TMO 모델은 다양한 종류의 분산 처리 실시간 시스템 응용에 적용 가능한 분산 실시간 객체 모델로 기존의 객체 모델에 대한 확장 모델이다[1]. TMO 모델은 일관성(uniformity) 및 다단계 모델링 수행 시, 각 단계에 대한 다양한 시간 정확도를 제공해 주고, 시간 구동 메소드를 갖고 있으며, 객체 내의 모든 메소드에 대한 데드라인 처리가 강화되었다. 다음 Figure 1은 TMO 모델의 구조를 나타낸다. TMO 모델만이 갖는 고유한 특징을 간단히 살펴보면 다음과 같다.

(1) ODSS(Object Data Sharing Segment) : SpM과 SvM 메소드 사이에 공유되는 데이터 집합.

(2) SpM(Spontaneous Method) : 메소드에 명시된 시간 조건에 의해 구동되는 메소드

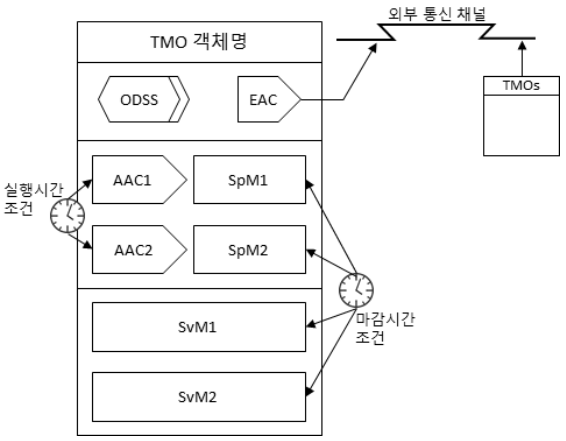
(3) SvM(Service Method) : 클라이언트로부터의 요청 메시지에 의해 구동되는 메소드

(4) AAC(Autonomous Activation Condition) : SpM 메소드에 부여되는 명시적인 실행 시간 조건으로, 아래와 같은 형태로 부여가 가능하다.

{"start-during (10am, 10:05am) finish-by10:10am", "start-during (10:30am,10:35am)finish-by10:40am"}

(5) BCC(Basic Concurrency Constrains) : SpM과 SvM의 실행에 대한 우선순위 조건에 대한 기본 규칙

(6) EAC(Environment Access Capability) : 외부 TMO 객체와의 통신 채널



[Figure 1] TMO 모델 구조

2.2 무장관리 시스템

무장관리 시스템은 항공기에 장착되는 무장을 제어하기 위한 관리시스템으로 무장의 발사, 상태관리, Jettison과 같은 기능을 수행하는 시스템이다. MIL-

HDBK-1760은 항공기와 무장들 사이의 전기적, 물리적, 논리적 연동을 위한 군 표준이다[2]. 위 표준에서는 주로 전기적, 물리적 연동을 위한 신호, 구성 형태에 관련된 내용들이 주로 언급된다. 무장관리 시스템은 구성형태에 따라서 중앙집중형 구조와 분산형 구조로 구분되며 구성형태에 따라서 무장관리시스템의 기능 복잡도 및 모듈 구성이 변경된다[3].

2.2.1 무장관리 시스템 탑재 소프트웨어

무장관리 시스템에 탑재되는 소프트웨어는 탑재되는 무장별로 무장관리 프로토콜을 처리해야 하며, 시스템의 구성형태에 따라서 중앙집중형 구조와 분산형 구조에 따라 다른 형태로 적용되어야 한다. 무장별 프로토콜의 처리 기능으로는 개별 무장들의 상태를 확인하고, 임무컴퓨터와 같은 임무 연동장비들을 통해 무장 제어를 위한 입력을 처리할 수 있어야 한다.

3. TMO 모델 기반의 무장관리 소프트웨어 설계

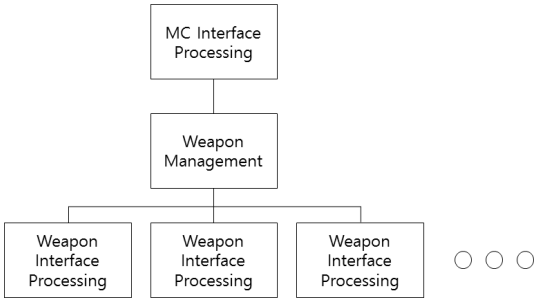
3.1 무장관리 시스템 구조에 따르는 객체 설계

무장관리 시스템의 구조에 따라 중앙집중형과 분산형에 대한 소프트웨어 설계가 필요하다. 두 가지 구조 모두 TMO 모델을 기반으로 설계하며, 설계를 위해 필요한 요건들을 분석하여 객체 기반으로 설계를 진행하였다.

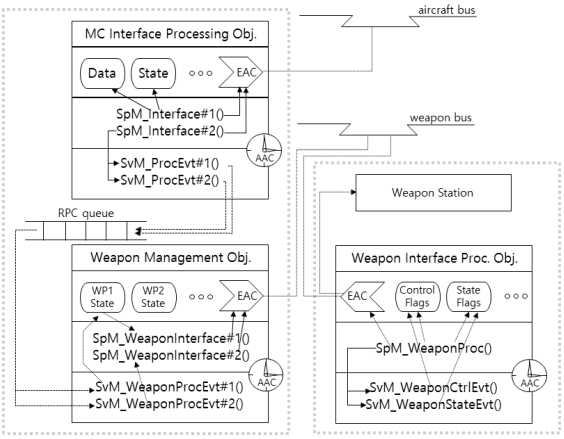
3.1.1 중앙집중형 무장관리 소프트웨어 설계

중앙집중형 무장관리 시스템은 임무컴퓨터 등의 외부 장비와 연동되고, 내부의 무장 연동 인터페이스를 관리하기 위한 SMCU(Stores Management Control Unit)를 중심으로 운영되는 방식이다. 따라서 SMCU에 탑재되는 소프트웨어는 외부와의 연동 및 내부 인터페이스 관리를 모두 수행할 수 있어야 한다.

Figure 2는 중앙집중형 무장관리 시스템의 SMCU



[Figure 2] SMCU 소프트웨어 구성 모듈



[Figure 3] SMCU 소프트웨어 TMO 모델 구성도

탑재 소프트웨어 구성 모듈들을 나타낸다. 각 구성 모듈은 TMO 모델을 이용하여 표현이 가능하며 위 Figure 3과 같이 나타낼 수 있다.

SMCU 소프트웨어는 각 모듈별 TMO 모델 기반의 객체들로 구성된다. MC Interface Processing 객체는 Aircraft bus를 통해 입력되는 패킷들을 처리하는 SpM 메소드들을 정의하며, SpM 처리 중 발생하는 이벤트들을 처리하기 위한 SvM들로 정의된다. SvM에서 처리된 무장관련 이벤트들은 Weapon Management 객체로 RPC Message Queue를 전송하여 무장제어를 수행한다. 외부 인터페이스 처리를 위한 I/O는 EAC(Environment Access Capability) 포트를 통해 처리되며, SpM과 SvM 사이의 데이터 공유는 ODSS(Object Data Store Segment)를 통해 이루어진다. SpM과 SvM은 AAC에 의해 Deadline 제어가 이루어져 실시간 처리를 수행한다.

Weapon Management 객체는 MC Interface Processing 객체로부터 입력받은 RPC Message

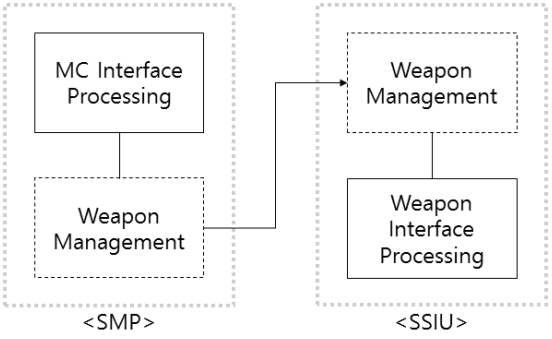
Queue를 SvM_WeaponProcEvt 메소드에서 처리한다. 처리 과정에서 ODSS에 무장 상태 정보등이 갱신되어야 하며, 갱신된 정보는 무장 인터페이스 처리를 위한 SpM_WeaponInterface 메소드에서 사용된다. SpM_WeaponInterface 메소드들은 EAC를 통해 Weapon bus에 접근하여 프로토콜에 대한 처리를 수행한다. 실제 무장 스테이션과 연결되어 하위 수준의 제어를 수행하는 Weapon Interface Processing 객체는 EAC를 통해 프로토콜 입력 및 처리를 수행하는 SpM_WeaponProc 메소드와 SpM 메소드에서 발생한 제어 또는 상태에 대한 이벤트 처리를 위한 SvM_WeaponCtrlEvt, SvM_WeaponStateEvt 메소드를 포함한다. SpM 메소드는 프로토콜 처리 중에 필요한 상태 정보들을 ODSS에 저장하여 SpM과 SvM 사이에 공유한다.

3.1.2 분산형 무장관리 소프트웨어 설계

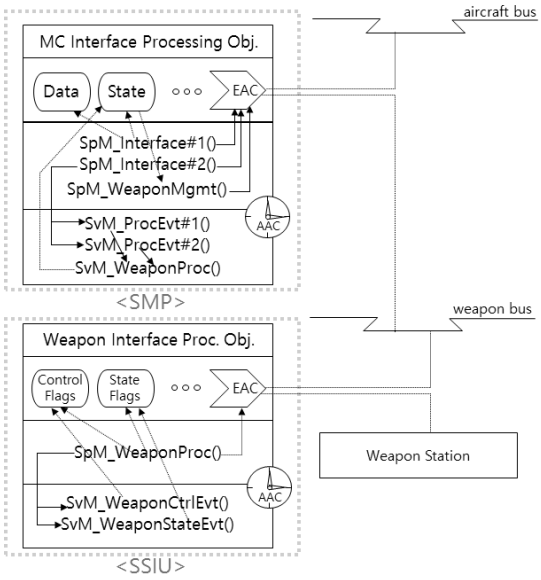
분산형 무장관리 시스템은 기존의 중앙집중형 무장관리 시스템에서 SMCU를 상위 장비간 연동을 위한 SMP(Stores Management Processor)와 무장 스테이션 연동을 위한 SSIU(Store Station Interface Unit)로 분리하여 운영하는 방식으로, SMP에 집중되는 프로토콜 처리가 SSIU로 분산되는 특징이 있다.

분산형 무장관리 시스템은 SMP와 SSIU로 기능이 구분되며, SMP에는 기존의 중앙집중형 무장관리 소프트웨어의 모듈 중, MC Interface Processing 모듈과 Weapon Management 모듈의 일부 기능(무장별 처리 분산)이 포함된다. SSIU에는 기존 Weapon Management 모듈의 프로토콜 처리 기능과 Weapon Interface Processing 저수준 무장 제어 기능이 포함된다. Figure 4의 SMP 및 SSIU 구성 모듈들은 다음 Figure 5와 같이 TMO 모델들로 표현된다.

위 Figure 5와 같이, 분산형 소프트웨어는 SMP 객체와 SSIU 객체로 분산되며, 기존 중앙집중형 TMO 모델 구성도에서의 Weapon Management 객체의 기능이 MC Interface Processing 객체와 Weapon Interface Processing 객체로 분산되고,



[Figure 4] 분산형 소프트웨어 구성 모듈



[Figure 5] 분산형 소프트웨어 TMO 모델 구성도

각각 1개의 객체로 설계가 단순화 된다. MC Interface Processing 객체에는 주기적으로 weapon bus를 통해 SSIU를 관리하기 위한 SpM_WeaponMgmt 메소드와, SSIU 관리를 위한 상태 및 데이터 관리용 SvM_WeaponProc 메소드가 추가된다. Weapon Interface Processing 객체의 SpM_WeaponProc 메소드는 MC Interface Processing으로부터 무장 관련 이벤트를 받아 SvM들이 처리하고, 처리 결과를 이용하여 무장 스테이션들을 제어하기 위한 프로토콜 처리 기능이 추가된다.

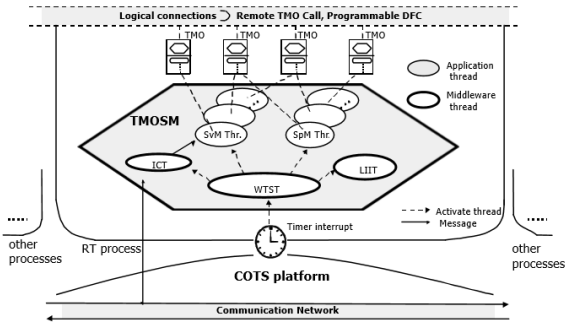
4. TMO 모델 지원 미들웨어 설계

4.1 TMO 모델 지원 미들웨어 참조 모델

TMO 모델을 지원하기 위한 미들웨어 구조는 TMOSM(TMO support middleware)로 Windows NT와 Linux 상에서 적용 가능한 형태로 제시되었다[4][5]. TMOSM은 SpM과 SvM의 실시간 제어를 위한 WTST(Watchdog Timer and Scheduler Thread)와 외부 입력 데이터 처리를 위한 ICT (Incomming Communication Thread), 내부 I/O 처리를 위한 LIIT(Local I/O Interface Thread) 등으로 구성된다.

4.2 무장관리 시스템 소프트웨어를 위한 TMO 미들웨어 설계 방안

위 Figure 6의 TMO 참조모델을 기반으로 무장관리 시스템을 위한 TMO 지원 미들웨어 설계 방안을 크게 태스크 관리, I/O 처리, 실시간 스케줄링 관점에서 기술하고자 한다.



[Figure 6] TMOSM 참조 모델

4.2.1 태스크 관리

SpM과 SvM은 AAC와 같은 Trigger 및 Deadline 조건에 의해 제어되며, 실시간의 정도(경성, 연성)에 따라 overrun에 대한 예외 처리가 가능해야 한다. TMO 모델은 Design-Time Guarantee 정책으로 설계 단계에서의 시간 요건이 정확해야 실제 수행시의 실시간성이 보장될 수 있다는 기본 원칙을 갖고 있으며 예외에 대해서는 사용자 처리가 가능하도록 설계한다.

SpM과 SvM은 운영체제에서 제공하는 사용자 스레드들로 구성하며, 하위 수준의 WTST와 ICT에 의해 Trigger될 수 있다. 각 SpM/SvM은 Blocking 조건으로 진입하였다가, Trigger 시 시그널 또는 메시지에 의해 활성화 되도록 설계한다.

4.2.2 I/O 처리

TMO 지원 미들웨어의 ICT 역할을 수행하는 I/O 전담 스레드를 운영하며, 주로 통신(Ethernet, MUX BUS)의 타 TMO 객체의 RPC Message들을 Distribution 하여 타켓 SvM으로 전달해 주도록 설계한다. 메시지에 대한 안정적인 처리를 위해 Queue를 구현해야 하며, 이를 위해서는 Non-Blocking Buffer 기술[6]을 적용한다.

4.2.3 실시간 스케줄링

특정 시작시간부터 주기적으로 수행되어야 하는 SpM의 경우 여러 SpM들의 동작시간을 고려해야 한다. 이를 위해서는 각 SpM들의 동작 주기에 대한 최대공약수 기법[7]을 사용하여 스케줄링의 Time Window를 계산하고, 해당 Window 크기만큼을 SpM에 할당한다. 이를 통해 SpM들의 실시간 스케줄링을 지원하게 되며, overrun을 체크하는 시간 단위가 된다.

5. 결 론

TMO 모델은 제안된 지 오래된 모델이며, 많은 연구를 통해 수정과 보완을 거쳐 최적화되어있고, 미들웨어 또한 기존에 상용 플랫폼 기반으로 기 개발되었기 때문에 기존 정보들을 활용하여 최적의 무장관리 소프트웨어 및 미들웨어 설계 방안을 제시할 수 있었다. 본 논문에서는 무장관리 시스템의 두 가지 시스템 구성 방식에 따르는 TMO 모델 기반의 설계 방안을 고안하고, 설계된 S/W를 지원하기 위한 TMO 지원 미들웨어의 설계 방안에 대해 연구를 진행하였다. 이 연구를 통해 실시간 설계 모

델을 무장관리 소프트웨어에 적용하여 설계 및 구현에 대한 효율을 높일 수 있을 것으로 판단된다. 향후 연구로는 제시된 설계 방안을 테스트베드에 적용하여 설계 시 요구된 실시간성을 만족하는지를 시험해봐야 하겠다.

References

1. Kim, K.H. and Kopetz, H., "A Real-Time Object Model RTO.k and an Experimental Investigation of Its Potentials", Proc. COMPSAC'94, pp.392-402, Nov. 1994.
2. MIL-HDBK-1760
3. D.H. Lee and H.J. Park, "The Design of a Fault Tolerant Stores Management System", J. of The Korea Society of Computer and Information, Vol. 20 No. 10, October 2015.
4. K. H. (Kane) Kim, Masaki Ishida, and Juqiang Liu, "An efficient middleware architecture supporting time-triggered message-triggered objects and an NT-based implementation", Proceedings 2nd (ISORC'99) (Cat. No.99-61702), 1999.
5. Jenks, Stephen F, Kim, Kane, et al. "A middleware model supporting time- triggered message-triggered objects for standard Linux systems", Real-Time Systems, 07/2007, Volume 36, Issue 1.
6. Kane Kim, "A Non-Blocking Buffer Mechanism for Real-Time Event Message Communication", Real-Time Systems, 32, 197-211, 2006.
7. Ho-Joon Park, C.H Lee, "A Study on the Scheduling Improvement for Periodic Real-Time Tasks on Middleware based on Linux (TMOSM/Linux)", Proceedings 7th 11-A, 7th, 2014.12.