JKSCI

한국컴퓨터정보학회논문지
Journal of The Korea Society of Computer and Information
Vol. 26 No. 11, pp. 101-110, November 2021
https://doi.org/10.9708/jksci.2021.26.11.101

A Study on the Standard Architecture of Weapon Control Software on Naval Combat System

Jae-Geun Lee*

*Engineer, Naval R&D Center, Hanwha Systems, Gumi, Korea

[Abstract]

The Weapon Control Software performs the function of supporting weapon operation within the Naval Combat System in connection with the Weapon System. As Weapon Control Software depends on an Weapon System, it has the characteristic that software modification is unavoidable with the change in Interface information. Modification of software causes an increase in development costs since it must take verification step such as software reliability test. In this paper, We design the standard architecture of weapon control software to minimize the modification elements of existing weapon control software. For Interface information management, Feature Model were applied to make a division between common factor and variable factor. In addition, Strategy Pattern were applied to improve the software design. Software evaluation test results show that new architecture provides better modifiability and reuse than existing software as well as the cost of development decrease.

▸ **Key words**: Architecture, Naval Combat System, Weapon Control Software, Software Reuse, Design Pattern, Strategy Pattern, Feature Model

[요 약]

무장통제 소프트웨어는 무장체계와 연동하여 함정 전투체계 내 무장 운용을 지원하는 기능을 수행한다. 무장통제 소프트웨어는 무장체계에 의존적이므로 연동 정보 변화에 따라 소프트웨어 수정이 필연적으로 이뤄지는 특성을 가진다. 소프트웨어의 수정은 신뢰성 시험과 같은 검증 단계를 반드시 거쳐야 하므로 개발 비용의 상승을 초래한다. 본 논문에서는 기존 무장통제 소프트웨어의 수정 요소를 최소화하기 위해 무장통제 소프트웨어 표준 아키텍처를 설계하였다. 연동 정보 관리를 위해 휘처 모델(Feature Model)로 공통요소와 가변요소를 구분하였고 디자인패턴 중 전략 패턴(Strategy Pattern)을 적용하여 소프트웨어 구조를 개선하였다. 소프트웨어 평가 실험을 통해 제안한 아키텍처가 기존 무장통제 소프트웨어보다 개발 비용이 감소하고 변경용이성과 재사용성이 향상된 것을 확인하였다.

▸ **주제어**: 아키텍처, 함정전투체계, 무장통제소프트웨어, 소프트웨어재사용, 디자인패턴, 전략패턴, 휘처모델

• First Author: Jae-Geun Lee, Corresponding Author: Jae-Geun Lee
*Jae-Geun Lee (jg5250.lee@hanwha.com), Naval R&D Center, Hanwha Systems
• Received: 2021. 10. 01, Revised: 2021. 11. 22, Accepted: 2021. 11. 22.

http://www.ksci.re.kr pISSN:1598-849X | eISSN:2383-9945

I. Introduction

함정 전투체계는 함정에 탑재된 각종 센서들을 이용하여 표적을 탐지하여 추적하고, 무장을 통제하여 교전을 수행하는 함정 작전 수행의 핵심 역할을 담당하고 있다[1]. 또한, 무장체계는 발사관을 여러 개 배치하고 각 발사관에 무장을 적재하여 무장이 이탈하도록 하는 장치를 뜻하며, 함정 전투체계의 소프트웨어 중 무장통제(Weapon Control) 소프트웨어는 전투체계와 무장체계를 이어주는 역할로 전투체계에서 식별한 위협 상황에 대해 무장 운용을 통한 교전 수행에 직접적인 도움을 주는 소프트웨어이다. 그러므로 함정 전투체계에서 복합적이고 동시다발적인 전장 상황에 대응하기 위한 무장통제 능력은 필수적이다.

그러나 체계 개발단계에서는 체계 요구사항 분석부터 하드웨어와 소프트웨어 설계 및 구현, 최종적으로 체계통합 및 시험평가(개발시험평가, 운용시험평가)등의 개발관리를 거치고 있지만 전력화 공백을 방지하기 위한 제한된 개발기간 및 예산으로 인하여, 모든 운용환경을 반영하지 못한 설계내용, 예기치 못한 운용자의 실수 등으로 인한 무기체계 양산 이후 운용단계에서 많은 문제점이 발생되고 있다[2]. 또한, IT 기술발전에 따라 2~3년 주기로 새로운 부품이 개발되기 때문에 기술변화에 적시적으로 대응하지 못하면 부품단종과 대규모 성능개량으로 인해 관리비용이 급증할 수 있다[3].

함정에 진화된 기술을 제공함으로써 무장체계의 성능도 향상된다. 이에 무장통제 소프트웨어는 무장체계의 연동정보에 의존적일 수밖에 없다. 이런 의존성이 높은 조건은 비용적으로 높게 측정되는 단점이 있다. 특히, 소프트웨어의 소스코드가 직접적으로 수정이 되어 신뢰성 검증을 위한 정적시험 및 동적시험(LDRA[4,5])을 수행해야 하며 개발 기간이 증가하여 개발 비용을 상승 시키는 요인이 된다. 더불어 SW 개발 및 제품시험의 특성 상 빈번한 SW의 수정과 그에 따른 기능 만족성을 재검증하기 위해 수행되는 회귀시험 등으로 인해 품질향상을 위해 요구되어지는 SW 신뢰성 시험을 수행할 시간적 여건이 부족한 실정이며, 여러 차례의 SW 신뢰성 시험을 수행하면서 개발자의 피로도가 증가되고 개발자의 신뢰성 시험 전문성 부족으로 인해 인적오류가 다소 발생하고 있는 실정이다[6].

본 논문에서는 새로운 함정 전투체계 혹은 무장체계가 설계 및 구현되면서 기존 대비 탑재되는 무장의 연동 정보가 변경될 경우 최소한의 소스코드 변경을 통한 유지보수 효율을 높이기 위해 기존 무장통제 소프트웨어에 휘쳐 모델링 방법과 전략 패턴(Strategy Pattern)을 적용하여 재설계하였으며 기존 소프트웨어와 비교한 결과를 기술하였다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구를 통해 함정 전투체계를 설명하고 무장통제 소프트웨어 표준 아키텍처에 필요한 휘처 모델 및 SOLID 원칙을 서술하고, 3장에서는 개선 항목을 식별하여 클래스 구조 변경 및 디자인패턴을 적용하여 새로운 무장통제 소프트웨어 표준 아키텍처를 설명한다. 4장에서는 본 논문에서 새롭게 제안된 무장통제 소프트웨어와 기존 소프트웨어의 개발 소요를 비교하여 검증한다. 마지막으로 5장에서는 결론으로 논문을 마무리한다.

II. Preliminaries

1. Naval Combat System

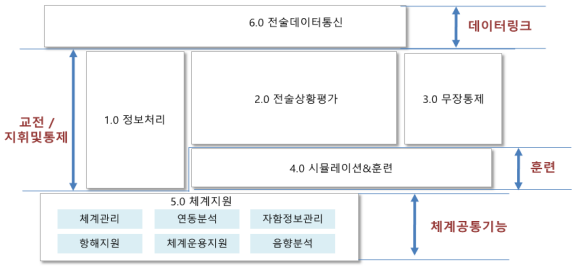

Fig. 1. Function of Combat Management System

함정 전투체계의 기능은 크게 데이터링크, 교전/지휘 및 통제, 훈련 그리고 체계공통기능으로 이루어져 있으며 함정 전투체계 소프트웨어의 전체 구성도는 Fig. 1과 같다[7].

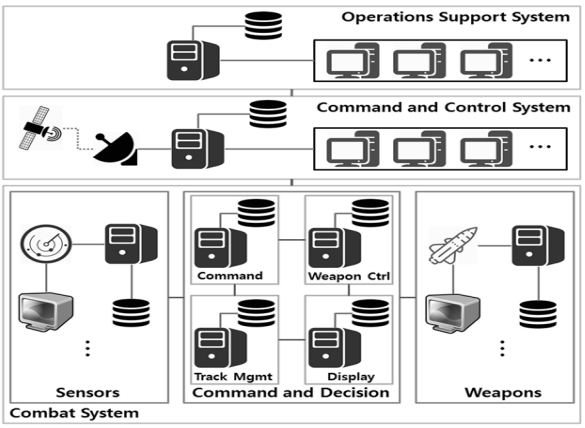

Fig. 2. Structure of legacy naval combat systems[8]

Fig. 1.에서 무장통제 기능은 Fig. 2.의 무장체계와 연동하여 함정 전투체계에서 탐지한 표적과의 교전을 수행할 수 있도록 지원한다. 적 위협에 대응할 수 있도록 무장체계를 통제하는 핵심 소프트웨어이다.

2. Feature Model

소프트웨어 프로덕트 라인에서 공통성과 가변성을 분석하기 위해 사용되는 도메인 분석 방법들로는 휘처 중심의 방법이 있다[9]. 그 중 휘처의 재사용에 초점을 맞춘 방법인 FORM(Feature-Oriented Reuse Method)이 있다[10]. FORM 방법론은 FODA[11]에서 제안한 휘처 모델로서 특정 도메인의 가변성을 식별하고, 그것을 반영한 재사용 가능한 컴포넌트를 만듦으로써, 재사용 컴포넌트의 조합으로 개별 제품을 빠르게 생성해 낼 수 있는 특징이 있다[12]. 휘처 모델을 바탕으로 AND/OR 그래프를 이용하여 공통성을 추출하는데, AND 노드는 제품의 필수적인 휘처를 말하고, OR 노드는 다른 제품으로 선택될 수 있는 양자 택일의 휘처를 말한다[13].

본 논문에서는 기존의 함정 전투체계 내 무장통제 소프트웨어의 도메인을 휘처 모델링하여 공통요소와 가변요소를 구분하였다. 공통요소로 식별된 휘처를 재사용 가능토록 설계하였으며, 가변요소의 선택에 따라 재조합이 가능하도록 설계하였다. 이러한 과정을 통해 무장통제 소프트웨어 표준 아키텍처의 수정 요소를 도출하였고 가변요소를 기존 클래스와 분리되도록 설계하였다.

3. S.O.L.I.D

컴퓨터 프로그래밍에서 S.O.L.I.D란 Robert C. Martin이 2000년대 초반에 명명한 객체 지향 프로그래밍 및 설계의 다섯 가지 기본 원칙이다[14,15]. 프로그래머가 시간이 지나도 유지 보수와 확장이 쉬운 시스템을 만들고자 할 때 이 원칙들을 함께 적용할 수 있다. SOLID 원칙들은 소프트웨어 작업에서 프로그래머가 소스 코드가 읽기 쉽고 확장하기 쉽게 될 때까지 소프트웨어 소스 코드를 리팩터링하여 코드 스멜을 제거하기 위해 적용할 수 있는 지침이다. 이 원칙들은 애자일(Agile) 소프트웨어 개발과 적응적 소프트웨어 개발의 전반적 전략의 일부이다[16].

본 논문에서는 무장통제 소프트웨어 표준 아키텍처를 위해 기존 무장통제 소프트웨어 클래스의 일부 기능을 새로운 클래스로 추가하는 과정에서 S.O.L.I.D 원칙을 적용하고자 노력하였다.

III. The Proposed Scheme

함정 전투체계에서 무장통제 소프트웨어는 적 위협에 대한 교전을 지원하기 위해 무장체계를 통제하는 소프트웨어이다. 무장체계가 발전함에 따라 무장통제 소프트웨어는 무장체계와의 연동 정보에 따라 수정이 필요하다. 기존의 함정 전투체계 내 무장통제 소프트웨어는 무장체계와의 의존성이 높아 유지보수를 위해 많은 개발 비용 및 시간이 투입되고 있어 무장 연동 정보 변경에 유연한 소프트웨어로 개선이 필요하다. 무장통제 소프트웨어 표준 아키텍처로의 수정과정은 Table 1.와 같이 3단계로 진행하였다.

Table 1. Development process of the Standard Architecture of Weapon Control Software

Step	Description
Identification	Identify function of software in service
Classification	Applying of Feature Model
Design	Design the Class Diagram
	Applying of Design Pattern

Step 1 : 기존 무장통제 소프트웨어의 동작 범위를 통해 기능을 식별 및 나열하였다.

Step 2 : 휘처 모델을 통해 식별된 기능들을 공통요소와 가변요소로 구분하였다.

Step 3 : 가변요소를 새로운 클래스로 디자인하고 의존성을 감소시키기 위한 적합한 디자인패턴을 적용하였다.

Step 1 : Identification

현재 운용 중인 무장통제 소프트웨어의 작동 범위를 분석하여 무장통제 소프트웨어 표준 아키텍처를 위한 기능을 식별하였다.

Step 1.1 : Working range of Weapon Control Software

함정 전투체계에서 운용 중인 무장통제 소프트웨어의 동작은 아래 Fig 3.와 같다.

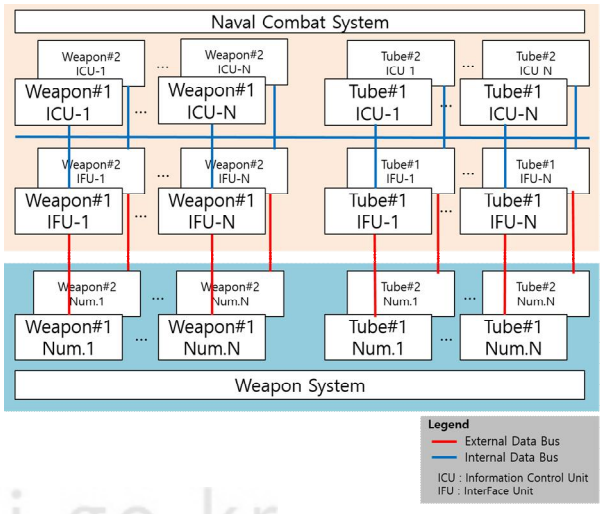

Fig. 3. Behavior of existing Weapon Control Software

무장통제 소프트웨어 내 인터페이스 모듈은 무장체계와의 직접적인 연동의 결과를 내부 데이터버스로 전달하며, 무장통제 소프트웨어 내 정보통제 모듈로부터의 명령을 내부 데이터버스를 통해 수신받아 무장체계로 전달하는 기능을 수행한다. 또한 정보통제 모듈은 인터페이스 모듈로부터 수신받은 연동 결과를 통해 전투체계에 필요한 정보를 제공하며, 전투체계에서 무장 운용을 위해 필요한 정보를 인터페이스 모듈로 전달하는 기능을 수행한다.

특이점으로는 무장체계와의 연동 개수가 1개 이상 존재하기에 하나의 응용프로그램이 여러 장비에 설치되며, 하나의 응용프로그램에서 조건문을 통해 각 연동을 처리해야 하는 1:N 구조이다. Fig 3.에서와 같이 무장체계 Weapon#1에 대해 전투체계의 Weapon#1 인터페이스 모듈 응용이 N개의 외부 연동을 수행하고 있으며, 인터페이스 모듈과 내부 통신하는 정보통제 모듈도 같은 구조로 되어 있음을 확인할 수 있다.

Step 1.2 : Identify function of weapon control software in naval combat system

무장통제 소프트웨어 작동 범위 중 무장체계와 직접적인 관련이 있는 인터페이스 모듈의 표준화를 진행하기 위해 기존 무장통제 소프트웨어 중 인터페이스 모듈의 Class Diagram을 분석하였다. Class Diagram의 설명은 아래 Fig. 4. 및 Table 2.과 같다.

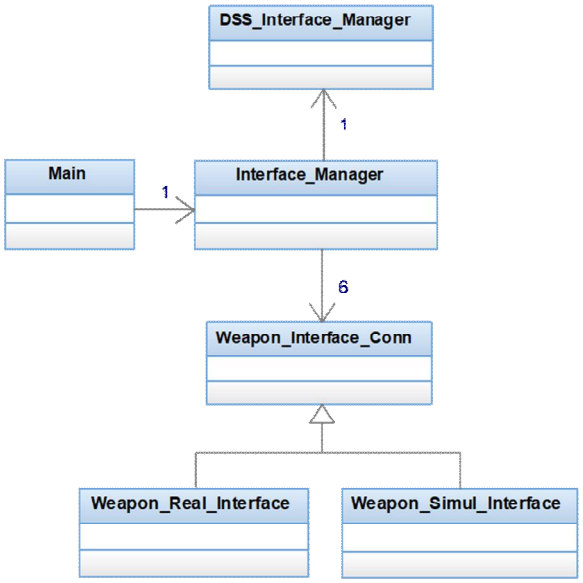

Fig. 4. Existing Class Diagram of Interface Unit Module in Weapon Control Software

Table 2. Interface Unit Module Class Description

Class Name	Description
Main	Main Class
InterfaceManager	Class for Weapon Interface Information management
WeaponInterfaceConn	Virtual class for Weapon Interface
WeaponRealInterface	Class for Weapon Interface Real Connection
WeaponSimulInterface	Class for Weapon Interface Simulation Connection
DSSInterfaceManager	Class for Internal Interface management

무장통제 소프트웨어 중 인터페이스 모듈은 총 6개의 클래스로 구성되어 있으며 자세한 설명은 다음과 같다. Main 클래스는 소프트웨어의 시작 부분을 담당하고 있으며 DSS_Interface_manager는 내부 통신을 위한 클래스로 내부 메시지의 Send 및 Callback을 위한 처리가 구현되어 있다. Interface_manager 클래스는 외부 연동을 위한 WeaponRealInterface 및 WeaponSimulInterface 클래스를 무장 연동 정보에 따라 정적으로 생성한다. WeaponInterfaceConn 클래스는 WeaponRealInterface 클래스 및 WeaponSimulInterface 클래스의 부모 클래스이며 연동 처리를 위한 Virtual 함수가 구현되어 있다. WeaponRealInterface 클래스는 실질적인 무장체계와의 연동을 지원하는 클래스이다. 실질적인 연동 개시 및 Write 및 Recv 위한 처리가 구현되어 있다. WeaponSimulInterface 클래스는 실질적인 연동이 아닌 가상의 연동을 지원하는 클래스이다. 클래스가 지원하는 기능은 WeaponRealInterface 클래스와 동일하다.

기존 무장통제 소프트웨어의 클래스 다이어그램을 분석하여 아래 Fig 5.와 같이 기능을 식별하여 도식화하였다.

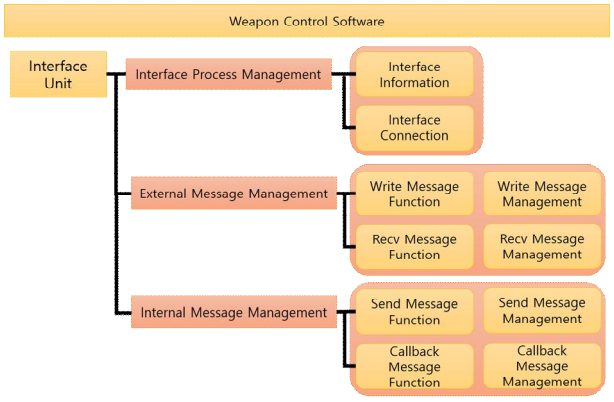

Fig. 5. Function of Weapon Control Software in service

무장통제 소프트웨어 중 인터페이스 모듈에서 처리하는 기능은 아래와 같이 분류하였다.

- 연동 정보 관리 기능
- 연동 개시 기능
- 무장체계로 송신할 외부 연동 메시지 처리 기능
- 무장체계로부터 수신할 외부 연동 메시지 처리 기능
- 함정 전투체계로 송신할 내부 메시지 처리 기능
- 함정 전투체계로부터 수신할 내부 메시지 처리 기능

Step 2 : Classification

앞서 식별된 기능을 휘쳐 모델을 통해 공통요소와 가변요소로 구분하였다. 또한 가변요소에 해당하는 기능의 소스코드를 분석하여 표준 아키텍처를 위한 구조를 설계하였다.

Step 2.1 : Analysis with Feature Model

앞서 식별된 기존 무장통제 소프트웨어의 기능을 휘쳐 모델에 적용하였다. Fig 6.은 휘쳐 도형을 통해 기존 함정 전투체계의 무장통제 인터페이스 소프트웨어의 기능을 휘쳐 모델로 구성한 그림이다.

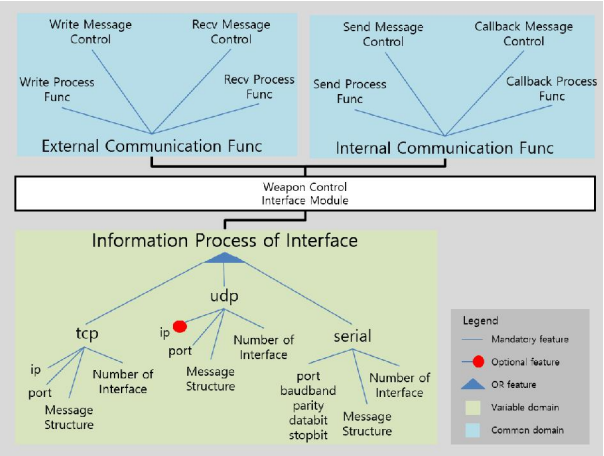

Fig. 6. Feature Model of Weapon Control Interface Module

Common domain으로는 연동 개시 기능과 메시지 처리 기능이 포함되었고, Variable domain으로는 연동 정보 관리 기능이 포함되었다. 통신 방법과 통신 방법에 따른 연동 정보를 식별하였다. 무장통제 소프트웨어는 다양한 연동 방식을 지원하기에 연동 방식에 따라 필요한 무장 연동 정보를 관리하여야 한다. Table 3.과 같이 가변요소인 연동 정보 관리 기능의 Primitive Feature를 정리하였다.

Table 3. Primitive Feature of Variable domain in Weapon Control Interface Module

Subcategory	Primitive Feature
tcp	ip, port, Message Structure, Number Of Interface
udp	ip (Optional feature), port, Message Structure, Number Of Interface
serial	port, baudband, parity, databit, stopbit, Message Structure, Number Of Interface

Step 2.2 : Analysis with Exist source code

휘처모델링의 결과로 구분된 가변요소인 연동 정보 관리 기능의 독립성을 위해 기존의 소스코드를 분석하였다.

Table 4. Exist Interface_Manager Class Source Code

Exist Interface_Manager.cpp

```
...
CWeapon_Real_Interface *apcRealIF[6] = {NULL};
for(int sub=0; sub<6; sub++) {
  apcRealIF[sub] = new CWeapon_Real_Interface (sub);
  ...
  apcRealIF[sub]->initialize();
...
```

Table 4.과 같이 정적으로 생성하는 연동 개수 처리는 무장 연동 정보가 변경되었을 경우 반드시 수정되어야 할 부분이다. Table 5.는 Interface_Manager 클래스로부터 수신받은 index에 따라 처리되는 방식을 보여주며, index에 따라 매핑되는 무장 연동 정보가 define 된 고정값을 사용하는 것을 확인할 수 있다.

Table 5. Exist Weapon_Real_Interface Class Source Code

Exist Weapon_Real_Interface.cpp

```
...
switch(index) {
case 1:
  struct_ip_mrq.imr_multiaddr.s_addr=inet_addr(WEAPON
  _FIRST_IP);
  struct_sockaddr_receive.sin_port=htons(WEAPON_FIRS
  T_PORT)
break;
case 2:
  struct_ip_mrq.imr_multiaddr.s_addr=inet_addr(WEAPON
  _SECOND_IP);
  struct_sockaddr_receive.sin_port=htons(WEAPON_SECO
  ND_PORT)
break;
...
```

기존의 구조는 무장체계와의 연동 정보에 따라 연동 개시를 위한 정보 혹은 외부 연동 메시지 처리 부분이 변경되어야 하는 의존적인 관계이므로 무장통제 소프트웨어는 무장체계와의 연동 정보에 영향을 직접적으로 받는다. 현재 설계되어있는 소프트웨어 구조로는 무장 연동 정보와의 의존성이 크므로 무장 연동 정보가 변경되었을 때 소스코드 수정 범위가 넓어질 수밖에 없다. 아래 Table 6.는 기존 무장통제 소프트웨어에서 연동 정보가 변경되었을 경우 수정이 필요한 클래스를 나타내었다.

Table 6. Modification of Existing Class

Class Name	Modification
Main	X
InterfaceManager	O
WeaponInterfaceConn	X
WeaponRealInterface	O
WeaponSimulInterface	O
DSSInterfaceManager	O

연동 정보가 변경되었을 경우 연동 정보를 관리하는 Interface_manager 클래스의 수정이 불가피함에 따라 WeaponRealInterface 및 WeaponSimulInterface 클래스 역시 수정되어야 하는 비효율적인 구조를 확인할 수 있다. 그러므로 휘처모델에서 분석한 것과 같이 가변요소인 연동 정보 관리 기능을 기존 클래스에서 분리한다면 소프트웨어 수정 요소를 상당히 감소시킬 수 있다. 무장통제 소프트웨어 표준 아키텍처를 위해 연동 정보 관리 클래스를 추가하였고 기존 클래스에서의 정적으로 처리되는 로직을 동적으로 변경하였다. 또한, 소스코드 수정 최소화를 위해 디자인패턴 중 전략 패턴을 적용하여 무장 연동 정보 관리 클래스 구조를 제안하였다. 소스코드 수정의 최소화는 소프트웨어 신뢰성 시험의 범위를 축소해 개발자 실수율을 낮추며 개발 시간을 대폭 감소시키므로 개발 비용을 낮추는 효과가 있다.

Step 3 : Design

무장통제 소프트웨어 표준 아키텍처를 위해 가변요소를 새로운 클래스로 확정하여 독립성을 높이고 의존성을 감소시키기 위해 적합한 디자인패턴을 적용하였다. 또한, 상호의존성을 낮추기 위해 정적으로 구현된 로직을 동적으로 변경하였다.

Step 3.1 Add the New Class for Variable Domain

휘처모델의 가변요소로 식별된 연동 정보 관리 기능을 위한 클래스를 추가하였다. 새로운 클래스를 추가하면서 클래스 작성 5대 원칙을 준수하고자 노력하였다. 단일책임원칙(SPR, Single Responsibility Principle)을 통해 클래스의 단일 책임을 수행하도록 하여 다른 클래스와의 독립성을 높였다. 개방폐쇄원칙(OCP, Open Close Principle)을 통해 무장 연동 정보에 확장성을 고려하여 설계하였다. 무장 연동 정보가 변경되더라도 최소한의 변경으로 기능이 동작하였고, 변경된 클래스에 따라 관련 정보를 동적으로 생성되도록 하여 기존 클래스가 수정되지 않게 설계하였다. 리스코프치환원칙(LSP, Liskov Substitution Principle)을 통해 무장 연동 정보를 쉽게 교체할 수 있도록 상위 클래스를 갖도록 설계하였다. 인터페이스분리원칙(ISP, Interface Segregation Principle)을 통해 무장 연동 정보에 필요한 인터페이스를 제공하도록 설계하였으며, 의존관계역전원칙(DIP, Dependency Inversion Principle)을 통해 기존 클래스가 추상 클래스가 아닌 구체 클래스를 의존토록 설계하였다.

Step 3.2 Architecture Design with Design Pattern

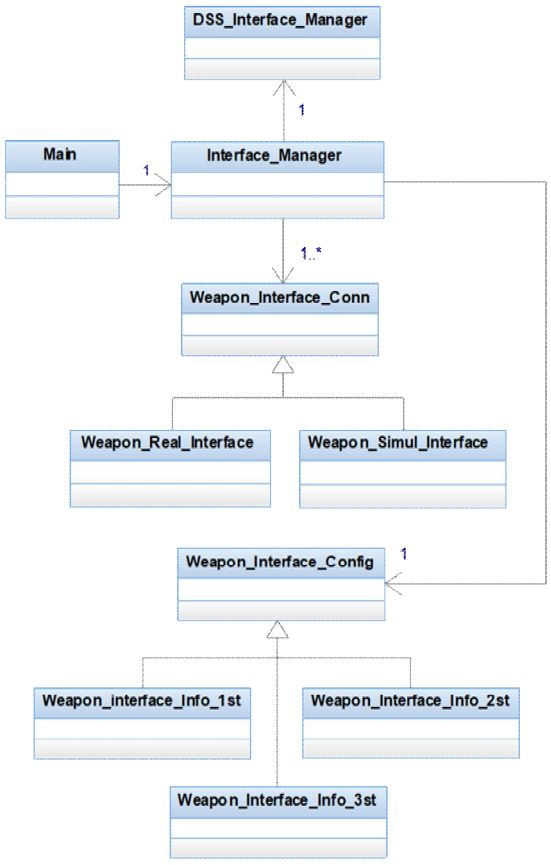

Fig. 7. New Class Diagram of Interface Unit Module With Strategy Pattern

본 논문에서는 기존 무장통제 소프트웨어의 클래스 구조에 디자인패턴 중 전략 패턴(Strategy Pattern)을 적용하였다. 전략 패턴은 소프트웨어에서 달라지는 부분과 그렇지 않은 부분을 분리하여 별도로 캡슐화 시키는 기법이다. 코드에서 달라지는 부분을 별도의 인터페이스로 캡슐화하는 방법으로 구현의 관점보다는 인터페이스에 맞추는 프로그래밍 기법이다[17]. 새롭게 추가된 클래스는 무장체계의 무장 연동 정보를 포함하고 있으므로, 각 무장체계별 무장 연동 정보를 쉽게 교체할 수 있도록 아래 Fig 7.과 같이 전략 패턴을 적용하였다.

무장체계와의 연동 정보에 대해 의존성을 줄이기 위해 무장통제 소프트웨어 내 연동 정보를 새로운 클래스로 분리하였다. 부모 클래스인 WeaponInterfaceConfig 클래스를 추가하였으며, 함정 전투체계별로 매칭되는 무장체계의 다양한 연동 정보 관리를 위해 자식 클래스로 WeaponInterfaceInfo1st, WeaponInterfaceInfo2st, WeaponInterfaceInfo3st 등과 같은 무장 연동 정보 관리 클래스를 추가하였다. 추가된 클래스의 요약은 Table 7.과 같으며, Table 8.과 같이 무장 연동 정보 관리 클래스를 Return 하도록 설계 및 구현하였다.

Table 7. New Class Description

Class Name	Description
WeaponInterfaceConfig	Interface Info Configuration Class
WeaponInterfaceInfo1st	Interface Info of First Naval Combat System
WeaponInterfaceInfo2st	Interface Info of Second Naval Combat System
WeaponInterfaceInfo3st	Interface Info of Thrid Naval Combat System

Table 8. New Weapon_Interface_Config Class Source Code

New Weapon_Interface_Config.cpp

```
...
CWeapon_Interface_info*      CWeapon_Interface_Config::
weapon_interface_info() {
   return new CWeapon_Interface_Info_1st();
}
...
```

이를 통해 Interface_Manager 클래스에서의 연동 정보는 WeaponInterfaceConfig의 실행 시점에 결정되면서, 연동 정보 변경에 유연해진 구조를 가질 수 있게 되었다. 위 구조를 통해 InterfaceManager 클래스의 연동 정보 관리는 WeaponInterfaceConfig를 통해 얻은 연동 정보에 따라 동적으로 WeaponInterfaceConn을 생성하여 완전한 분리 구조를 갖도록 개선하였다.

Table 9. New Interface_Manager Class Source Code

New Interface_Manager.cpp

```
...
iCount = mpcWPIF_Info->get_IF_Count();
stInterfaceInfo = mpcWPIF_Info->get_IF_Info();
CWeapon_Real_Interface *apcRealIF[iCount] = {NULL};
for(int sub=0; sub<iCount; sub++) {
   apcRealIF[sub] = new CWeapon_Real_Interface(sub);
   apcRealIF[sub]->setinterfaceinfo(stInterfaceInfo);
   apcRealIF[sub]->initialize();
...
```

무장체계의 무장 연동 정보가 변경되었을 경우 WeaponInterfaceInfo1st 클래스 내 연동 정보만 변경하면 되기에 Table 9.과 같이 Interface_Manager 클래스는 수정될 필요가 없다. 더불어 새로운 무장 연동 정보가 추가되었을 경우 WeaponInterfaceInfo3st와 같은 클래스를 추가하고 WeaponInterfaceConfig 클래스에서 Return을 한다면, 연동 정보 구성 및 연동 정보 관리 클래스 외 나머지의 기존 클래스는 변경 없이 사용 가능하기에 무장체계와의 의존성을 감소시킬 수 있다. 이에 소스코드 수정 및 신뢰성 시험 시간을 단축하여 개발기간을 대폭 축소할 수 있다.

무장 연동 정보가 변경될 때, 무장통제 소프트웨어를 수정하는 프로세스는 4단계이며, 각 단계의 설명은 다음과 같다.

1. Modify Interface Information
   - 변경된 연동 정보에 따라 관리 중인 무장 연동 정보 수정
2. Modify Source Code
   - 변경된 연동 정보에 대응할 연동 개시 소스코드 수정
   - 변경된 연동 정보에 대응할 외부 연동 메시지 처리 관련 소스코드 수정
3. Reliability Test
   - 변경된 연동 정보에 대응할 소스코드 수정에 따른 정적분석 및 동적분석 신뢰성 시험 수행
4. Build and Verify
   - 수정된 코드 build 및 side effect 검증

Step 3.3 : Expectation Effectiveness

본 논문에서는 무장 연동 정보 변화에 대한 무장통제 소프트웨어의 영향성을 최소화하기 위한 연구를 수행하였고

기대효과는 다음과 같다.

- 무장 연동 정보 관련 무장통제 소프트웨어 소스코드 수정 최소화
- 무장통제 소프트웨어 신뢰성 시험 범위 최소화

IV. Software Evaluation

본 논문에서 함정 전투체계 무장통제 소프트웨어의 개선 정도를 확인하기 위해 무장 연동 정보 변경에 대한 수정 요소와 신뢰성 시험 수행의 총 시간을 비교하여 평가하였다. 평가에 대한 실험 환경은 아래 Fig. 8. 및 Table 10.와 같다.

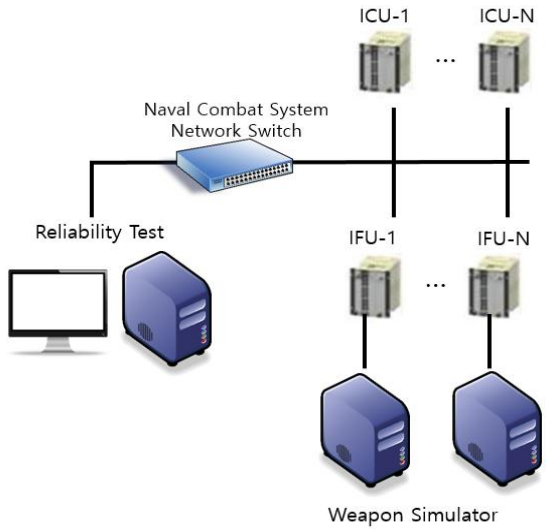

Fig. 8. Structure of Test Environment

Table 10. Performance of Test Environment

Item	Reliability Test	ICU, IFU	Weapon Simulator
CPU	Intel Core i9-10900 @5.20GHz	Intel Core i7-4700 EQ @2.40GHz	Intel Xeon E3-1231 @3.40GHz
Memory	64 GB	8 GB	64 GB
OS	Windows 10	WindRiver Linux	Windows 7

제안한 무장통제 소프트웨어 표준 아키텍처와 기존의 무장통제 소프트웨어의 평가를 위해 연동 정보 변경에 대한 의존성을 비교하는 실험을 하였다. 개발 비용이 가장 많이 투입되는 소스코드 수정이 필요한 클래스의 수, 소스코드 라인 수를 통해 재사용률을 비교하였으며, 신뢰성 시험은 정적분석 및 동적분석 모두 LDRA 도구를 이용하여 실제 수행시간을 측정하였고 신뢰성 시험을 수행한 시간을 비교하였다. 실험 시나리오는 현재 개발단계의 OOO함정 전투체계에 적용한 CASE로 총 3가지로 설정하였다.

- CASE-1 : 무장 연동 개수가 변경된 CASE
- CASE-2 : 무장 연동 Port Number가 변경된 CASE
- CASE-3 : 새로운 함정 전투체계가 적용된 CASE

CASE-1. Analysis of the change in the number of Interface

Table 11. Analysis of Code Reuse Rate (CASE-1)

The Number of	Total Count	Changeable Count	Reuse Rate
Exist Classes	6	4	33.3%
New Classes	10	1	90.0%
Exist Lines	6,411	1,099	82.9%
New Lines	6,731	2	99.9%

Table 12. Compare time to perform Reliability Test (CASE-1)

Reliability Test	Exist Software	New Software	Diff.
Static Test	27 hours	0.5 hours	-26,5 hours
Dynamic Test	60 hours	1 hours	-59 hours

CASE-1의 경우 Table 11.과 같이 연동 개수 변경에 따라 수정이 필요한 클래스 수는 새롭게 제안한 무장통제 소프트웨어에서 총 1개, 소스코드 라인 수는 2 Lines로 나타났다. 기존과 비교하여 클래스 재사용률은 56.7%, 소스코드 라인 수의 재사용률은 17.0% 증가함을 확인하였다. 또, 신뢰성 시험을 수행한 시간을 비교한 Table 12.과 같이 정적분석 최대 26.5시간, 동적분석 최대 59시간이 절약됨을 확인하였다. 무장 연동 개수가 증가 및 감소하는 상황이 많아질수록 더욱 효과적일 것으로 기대한다.

CASE-2. Analysis of the change in the Port Number

Table 13. Analysis of Code Reuse Rate (CASE-2)

The Number of	Total Count	Changeable Count	Reuse Rate
Exist Classes	6	1	83.3%
New Classes	10	1	90.0%
Exist Lines	6,411	1	99.9%
New Lines	6,731	1	99.9%

Table 14. Compare time to perform Reliability Test (CASE-2)

Reliability Test	Exist Software	New Software	Diff.
Static Test	4 hours	0.5 hours	-3.5 hours
Dynamic Test	11 hours	1 hours	-10 hours

CASE-2의 경우 Table 13.과 같이 연동 Port Nubmer의 변경에 따라 수정이 필요한 클래스 수는 새롭게 제안한 무장통제 소프트웨어에서 총 1개, 소스코드 라인 수는 1 Lines로 나타났다. 기존과 비교하여 클래스 수 및 소스코드 라인 수는 큰 차이가 없었다. 그러나, Table 14.과 같이 신뢰성 시험을 수행한 시간에는 큰 효과를 보였다. 정적분석 최대 3.5시간, 동적분석 최대 10시간이 절약됨을 확인하였다.

기존 무장통제 소프트웨어는 조금의 연동 정보 변경에도 본 클래스가 변경되는 구조이기에 본 클래스 전체에 대한 재검증이 필요하므로 단순한 변경이 발생할 때에도 신뢰성 시험 수행을 위한 많은 시간을 투입할 수밖에 없다. 그렇기에 디자인 패턴 중 전략 패턴에 맞춰 연동 정보 구성 클래스를 도입하여 메인 클래스에 영향을 주지 않도록 개선하였기에 신뢰성 시험 수행 시간이 대폭 감소함을 확인하였다. 단순한 설정 정보 값의 변경 상황이 빈번히 발생하므로 제안한 아키텍처의 효과는 더욱 클 것으로 기대한다.

CASE-3. Analysis of the change by the introduction of new Naval System

Table 15. Analysis of Code Reuse Rate (CASE-3)

The Number of	Total Count	Changeable Count	Reuse Rate
Exist Classes	6	4	33.3%
New Classes	11	2	80.0%
Exist Lines	6,411	1,099	82.9%
New Lines	6,731	87	98.7%

Table 16. Compare time to perform Reliability Test (CASE-3)

Reliability Test	Exist Software	New Software	Diff.
Static Test	27 hours	1 hours	-26 hours
Dynamic Test	60 hours	1 hours	-59 hours

CASE-3는 새로운 함정 전투체계가 도입되면서 그에 맞춘 무장 연동 정보가 추가된 경우이다. Table 15.와 같이 새롭게 제안한 무장통제 소프트웨어에서 클래스 수는 총 2개, 소스코드 라인 수는 87 Lines로 나타났다. 새로운 함정의 무장 연동 정보가 포함된 새로운 클래스를 1개 생성하였다. Table 16.와 같이 신뢰성 시험을 수행한 시간은 정적분석 최대 26시간, 동적분석 최대 59시간의 시간 절약 효과가 있음을 확인하였다. 제안한 무장통제 소프트웨어 표준 아키텍처의 재사용률은 기존 대비 클래스 수 46.7%, 소스코드 라인 수 15.8%가 증가하였다. 또한 신뢰성 시험 수행 시간은 CASE-1의 효과와 유사하게 나타났다.

제안한 무장통제 소프트웨어 표준 아키텍처에서는 무장 연동 정보와의 의존을 최소로 하기 위해 기존 클래스의 기능을 분리하여 새로운 클래스로 확정하였다. 이에 독립적으로 설계 및 개선하였기에 새로운 무장 연동 정보 증감에도 유연한 대응을 할 수 있으므로 추후 진행될 사업의 개발 비용을 대폭 축소시킬 수 있을 것으로 기대한다.

V. Conclusions

무장통제 소프트웨어는 함정 전투체계에서 무장을 통한 교전을 위한 필수 소프트웨어이지만, 무장체계의 연동 정보에 의존적이다. 무장체계의 형상이 변경된다면 반드시 무장통제 소프트웨어도 변경되어야 한다. 소프트웨어의 수정은 소스코드 수정, 기능시험, 신뢰성 시험 수행이 필요하며, 이는 개발기간 투입에 비례하므로 개발 비용 증가의 핵심 요인으로 적용한다.

본 논문에서는 기존 무장통제 소프트웨어를 외부 환경에 대해 최소한의 대응을 위한 무장통제 소프트웨어 표준 아키텍처로 개선하는 연구를 진행하였다. 휘처모델을 통해 가변 요소를 구분하여 연동 정보 관리 클래스를 설계하고 이를 동적으로 생성하여 무장 연동 정보를 독립적으로 설계하여 불필요한 소스코드 수정 없이 재사용 가능하도록 하였다. 또한 디자인패턴 중 전략 패턴(Strategy Pattern)을 이용한 연동 정보 구성 클래스를 도입하여 연동 정보 관리 클래스의 최적화를 통해 무장 연동 정보의 의존성을 감소시켰다. 새로운 무장통제 소프트웨어의 클래스 및 소스코드의 재사용률과 신뢰성 시험 수행 시간 비교를 통해 변경용이성과 재사용률의 향상으로 개발 비용이 크게 감소됨을 확인하였다. 더불어, 본 논문에서 제안한 방법은 신규 사업 시 즉시 적용할 수 있으므로 앞으로의 사업에서 개발 비용 산출에도 도움이 될 수 있을 것으로 기대한다.

REFERENCES

[1] Joon-Ho Lee, Ki-Hyun Jung, Kee-Young Yo, “Hybrid Information Hiding Method Based on the Characteristics of Military Images on Naval Combat System” Journal of Korea Multimedia Society Vol. 19, No. 9, September 2016, pp. 1669-1678

[2] Jin-Woo Oh, Jong-Kyu Kim, Ji-seon Yu, Jae-Hyeong Yun, Chi-Hoon Song, “Research on DB Construction and Utilization Measure to Analyze the Cause of Weapon System Software Engineering Change and Derive Improvement Plan” Journal of the Korea Academia-Industrial cooperation Society Vol. 22, No. 4, pp. 331-337, 2021.

[3] Jin-kook Im, Yong-Seok Choi, Su-Beom Kim, “Naval combat system life time support (LTS) performance and development” Defense & Technology , Vol. 497, pp. 84-95, 2020

[4] https://ldra.com/aerospace-defence

[5] https://moasoftware.co.kr/case_study/ldra

[6] Hyoung-Kweon Kim, “A study for the reduction of the SW reliability test time and human errors using the SW reliability test automation” Journal of The Korea Society of Computer and Information, Vol. 20, No. 10, pp. 45-51, October 2015.

[7] Young-Dong Heo, “A Study on the Standardization of System Support Software in the Combat Management System” Journal of The Korea Society of Computer and Information, Vol. 25 No. 11, pp. 147-155, November 2020.

[8] Jin Yong Im and Dong Seong Kim, "Performance Evaluation of Virtualization Solution for Next Generation Naval Combat Systems" Journal of The Institute of Electronics and Information Engineers Vol.56, NO.2, February 2019.

[9] Minseong Kim, Sooyong Park, "A Domain Analysis Method for Software Product Lines Based on Goals, Scenarios, and Features", Journal of KISS : Software and Applications 33(7), pp. 589-604, 2006

[10] Kyo C. Kang, Sajoong Kim, Jaejoon Lee, Kijoo Kim, Euiseob Shin, and Moonhang Huh, "FORM: A Feature-Oriented Reuse Method with Domain-Specific Reference Architectures," Annals of Software Engineering, pp. 143-168, May 1998.

[11] K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak, and A. S. Peterson, "Feature-Oriented Domain Analysis(FODA) Feasibility Study," Technical Report CMU/SEI-90-TR-21, Pittsburgh, PA., Software Engineering Institute, Carnegie Mellon University, 1990.

[12] Kyungmo Yang, YoonHo Jo, Kyo Chul Kang, "Modeling FORM Architectures Based on UML 2.0 Profiling", Journal of KISS : Software and Applications 36(6), 2009.6, 431-442

[13] Chee-Yang Song, Eun-Sook Cho, Chul-Jin Kim, "A Formal Specification and Checking Technique of Feature model using Z language", Journal of the Korea Society of Computer and Information 18(1), 2013.1, 123-136(14 pages)

[14] Robert C. Martin, "Agile Software Development, Principles, Patterns, and Practices," Prentice Hall, New Jersey, pp. 95-145, 2002.

[15] Robert C. Martin, "Clean Code: A Handbook of Agile Software Craftsmanship," Prentice Hall, New Jersey, pp. 138-140, 2008.

[16] Chi-Sun Baek, Jin-Hyang Ahn, “A Study of the Standard Interface Architecture of Naval Combat Management System”, Journal of The Korea Society of Computer and Information, Vol. 26 No. 1, pp. 147-154, January 2021

[17] Eric Freeman, Elisabeth Freeman, Kathy Sierra, Bert Bates, “Head First Design Patterns”, O’REILLY, 2005.

Authors

Jae-Geun Lee received the B.S. degree in Computer Engineering from Yeungnam University, Korea, in 2014. He is currently working in Hanwha Systems Co. from 2014. He is interested in Weapon Control Software of the naval combat management system, Software Reuse, Design Pattern and so on.