Journal of The Korea Society of Computer and Information
Vol. 24 No. 7, pp. 101-108, July 2019
www.ksci.re.kr
https://doi.org/10.9708/jksci.2019.24.07.101

Design of the Scalable Naval Combat System Software using Abstraction and Design Pattern

Ki-Tae Kwon*, Ki-Pyo Kim*, HwanJun Choi*

Abstract

In this paper, we propose a new scalable and reliable combat system software in battleship which was developed procedurally in the past. Recently, combat system software is required to change frequently due to addition of new equipment and change of function. To solve those problems, this paper propose how to change combat system software into scalable software using class structure change and design pattern. Simulation results show that our scheme provides better performances and reliability than conventional scheme. Therefore proposed scheme can be efficiently used in Naval combat system.

▸Keyword: Scalable Software, Design Pattern. Reliability, Naval Combat System, Combat System, Warfare, Abstraction, Abstraction Development Method

I. Introduction

방위산업에서 총 개발비용 비중이 하드웨어 중심에서 점점 소프트웨어 중심으로 변화함에 따라 수정이 쉽고 유지보수비용이 적게 드는 소프트웨어에 대한 관심이 늘어나고 있다. 기존의 절차 지향적으로 개발된 함정 전투체계 소프트웨어들은 신규 장비의 추가, 하드웨어 기능 변경 및 업그레이드 등에 영향을 많이 받아 지속적인 수정과 그로 인한 유지보수 비용이 많이 발생하고 있다. 이에 함정 전투체계 소프트웨어를 보다 적은 코드 수정이 일어나고 더 쉽게 수정할 수 있도록 확장성 있는 소프트웨어로 발전시켜 변화에 잘 대처 할 수 있도록 할 필요성이 있다. [4]

본 논문에서는 함정 전투체계 소프트웨어 모듈 중에서 센서 및 무장을 통제하는 교전 소프트웨어를 확장성 있게 수정함으로써 수상함 전투체계의 무장, 센서 등의 추가 및 수정에 대한 변화에 보다 쉽고 적은 비용으로 유지보수 할 수 있도록 하는 방안을 제시하고 그에 따른 효과를 분석하였다. 본 논문의 구성은 다음과 같다.

II장에서 함정전투체계에 대한 설명과 기존 함정전투체계 소프트웨어에 대한 문제점을 분석하여 본 연구의 필요성을 서술하고 III장에서는 함정전투체계 중 교전 소프트웨어를 개선 대상으로 선정하여 클래스 구조 변경 및 디자인패턴 적용을 통한 수정 개발 과정을 제시하고 IV장에서는 제안된 교전 소프트웨어의 성능을 검증하며 V장에서는 디자인 패턴을 이용한 함정전투체계 교전 소프트웨어 수정 개발 연구결과와 앞으로 얻을 수 있는 효과와 향후 추가적인 연구방향에 대해 정리하였다.

• First Author: Ki-Tae Kwon, Corresponding Author: Ki-Tae Kwon
*Ki-Tae Kwon (kt0830.kwon@hanwha.com), SW Team(Naval), Hanwha Systems Co.
*Ki-Pyo Kim (kipyo.kim@hanwha.com), SW Team(Naval), Hanwha Systems Co.
*HwanJun Choi (hwanjun627@hanwha.com), SW Team(Naval), Hanwha Systems Co.
• Received: 2019. 05. 13, Revised: 2019. 07. 04, Accepted: 2019. 07. 04.
• This paper will expand the paper ("A Study on the Standardization of the combat system software in battleship using Object-oriented Design.") presented at the 56th Summer Conference Korea Society of Computer and Information 2017.

II. Preliminaries

1. Naval combat system Architecture

함정 전투체계는 다양한 적의 동시 다발적인 위협의 전투상황 하에서 함 탑재 센서와 외부로부터 획득된 정보를 종합 처리하여 최적의 전투능력을 제공할 수 있도록 지휘 및 무장을 통제하는 기능이 통합된 자동화 시스템이다.[5]

국산화과정을 거쳐 국산 함정전투체계가 실전배치 된 이래로 지속적으로 추가 개발 및 업그레이드되어 최신함정에 탑재되고 있으며 최근 무장, 센서의 기술개발 및 최신방산기술적용으로 추가 연동 및 기능 수정이 빈번히 일어나고 있으며 이에 전투체계 소프트웨어도 변경 또한 수시로 일어나고 있다.

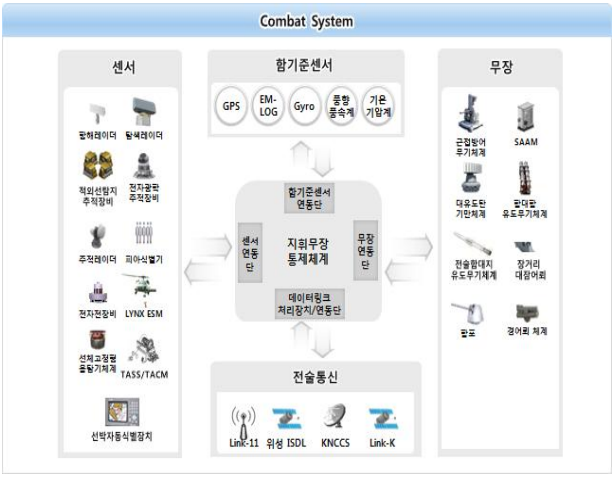

Fig. 1. Configuration of Combat System

기존 함정 전투체계는 설계 시에 추가 장비연동에 대한 확장성이 충분히 고려되지 않아 추가 연동 시 관련 소프트웨어 수정이 일어나고 관련패치를 위해 실전 배치된 함정 전체에 대한 전투체계 소프트웨어 패치도 같이 일어나게 되어 매우 번거로운 상황이 발생하고 있다. 또한 개발당시 절차중심의 코딩이 이루어져서 무장, 센서, 통신위성장비 등의 신규연동, 장비의 기능 변경 및 추가 등의 수정이 필요할 경우, 소프트웨어의 문제점 발견으로 수정이 필요할 경우 등에 노력과 시간이 많이 드는 문제점이 있다.

위의 기존 함정 전투체계의 문제점을 인식하고 개선하고자 기존 전투체계 소프트웨어를 확장성과 신뢰성이 높이도록 디자인 패턴과 클래스 객체화, 세분화를 통한 수정 개발을 진행하였다. 특히 여러 함정 전투체계 모듈 중 무장, 센서와 밀접한 연관이 있으면서 확장성 있도록 개선이 시급한 교전 모듈을 선정하여 선행 적용하였다. 국산화 전력화 당시 함정 전투체계 교전 소프트웨어는 지속적인 적 탐지 및 항시 교전 가능하도록 하여야 하는 특성과 적 위협에 대하여 빠른 대응이 필요한 특성이 있기 때문에 효율성이 강조되고 국산화 과정에서 객체지향에 대한 이해와 요구에 부족하여 절차지향적인 면이 강조되어 구현된 측면이 있다. 또한 앞서 말한 것처럼 추가 장비의 연동이 지속적으로 발생하고 있어 교전 소프트웨어의 변경이 수시로 일어나고 있다. 이에 확장성이 있는 소프트웨어로의 변화와 효율적이면서도 적은 시간과 노력으로 유지보수가 가능한 소프트웨어로의 변화가 요구되고 있으며 방위산업의 특성상 신뢰성이 높은 소프트웨어가 요구된다.

본 논문에서는 기존 함정전투체계를 디자인패턴 적용과 클래스 객체화를 통해 수정 개발하는 방안을 연구하고 함정전투체계 중 교전에 선행 적용하여 수정 개발된 교전이 얼마나 확장성과 신뢰성을 가지면서도 이전보다 얼마나 더 효율적으로 변화되었는지를 실험결과를 통해 확인하였다.

III. The Proposed Scheme

함정 전투체계에서 교전 소프트웨어는 함정에 탑재된 센서, 무장 등의 자원을 관리하고 통합하여 적 위협을 탐지, 위협분석, 무장할당 및 교전, 명중평가 등 지휘 및 무장통제를 자동화함으로써 위협표적에 대한 전투효과를 극대화 시키는 소프트웨어이다. 현재 국산 함정 전투체계에서의 교전 소프트웨어는 함정의 임무에 따라 대공전, 대함전, 대잠전 대지전 등 작전성분별로 설계 및 구현되어 있으며 센서, 무장 등의 추가 및 연동장비의 기능 변경에 영향을 받아 유지보수에 많은 시간과 비용이 투입되고 있어 확장성 있는 소프트웨어로 선행 개선하는 대상으로 선정하였다.

1. Class Design & Design Pattern applying

함정전투체계 교전 소프트웨어 수정개발과정은 기존의 절차지향적 설계에서 클래스를 객체 지향적으로 재설계하고 객체지향의 기본개념을 적용하여 세분화하고 객체별 클래스에 알맞은 설계 디자인 패턴을 적용하여 확장성을 확보하고 기존 자료구조를 검증된 STL(Standard Template Library)로 상황에 맞게 적용하여 안정적이고 더 나은 성능을 내도록 변경하였다. 개발단계는 아래의 Table 1 의 4단계 과정으로 진행하였다.

Table 1. Proposed Development Process of Combat system.

Step	Description
Requirement Analysis	Feature of The Warfare in Combat System Software
Design	Class Diagrams Design of Warfare in Combat System Software
	Application of Design Patterns
Implementation	Reuse Warfare Algorithm and make Calculation Class Library
	Application of Standard Template Library
Test & Debug	Unit Test of Proposed Warfare Software
	Software Reliability test using Commercial Software
	Functional Test in integrated development environment.

Step 1 요구사항 식별단계 : 요구분석 단계에서는 기존 전투체계의 교전 모듈들의 요구사항들을 통합 분석하고 클래스 추상화를 위한 공통속성과 고유속성을 식별하였다.

Step 2 클래스 식별 및 설계 단계 : 클래스 식별 및 설계 단계에서는 추상화를 통한 클래스를 확정하고 확장성 있는 전투체계 교전에 적합한 디자인 패턴을 적용 설계하였다.

Step 3 구현 단계 : 구현 단계에서는 식별된 클래스간의 독립성을 유지하고 클래스 간의 상호의존성은 낮추도록 구현하였다.

Step 4 시험 및 디버그 단계 : 시험 및 디버그 단계에서는 시뮬레이터를 활용한 유닛테스트, 정적/동적 신뢰성 시험을 수행하였고 최종 수상함 전투체계에 제안된 교전 SW를 탑재하여 동작여부를 확인하였다.

1.1 Step 1 : Requirement Analysis

요구사항 분석 단계에서는 먼저 여러 기존 함정전투체계 교전 요구사항을 취합하여 교전 고유의 기능에 대한 요구사항과 함정에 따른 센서, 무장에 특화된 요구사항, 추가 장비연동 및 기술 변경으로 수정, 추가된 요구사항을 식별하였다.

각 함정에 공통적으로 요구된 요구사항은 기능중심으로 요구사항을 나누었고 장비에 영향을 받아 수정이 일어날 수 있는 특화된 요구사항은 함정 별로 변경된 이유를 분석하고 분류하여 공통적인 특성을 식별하기 쉽도록 하였다.

1.2 Step 2 : Design

1.2.1 Class Abstraction

기존 교전 소프트웨어의 클래스 구조를 분석하여 작전성분별 기능중심으로 구현된 구조를 아래의 Fig. 2 와 같이 도식화하여 확인 하였다.

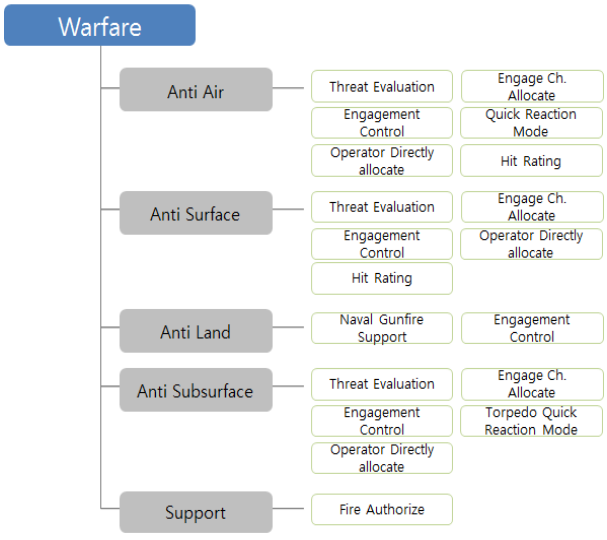

Fig. 2. Warfare SW Architecture

위의 교전 SW 구조의 문제점은 작전성분별로 구성되어 표적정보, 교전정보 등이 비효율적으로 나눠져 관리되고 유사한 기능의 클래스가 중복구현 되어있다는 것이다. 이러한 이유로 교전 클래스 구조를 추상화 과정을 통해 객체 지향적으로 변경하였다. 가장 큰 차이는 클래스구성이 기능과 객체중심으로 변경된 것이며 클래스를 보다 세분화하였고 기능별로 독립적성이 보장될 수 있도록 하였다. 변경된 전투체계 교전 모듈의 구조는 아래의 그림 Fig. 3 과 같다.

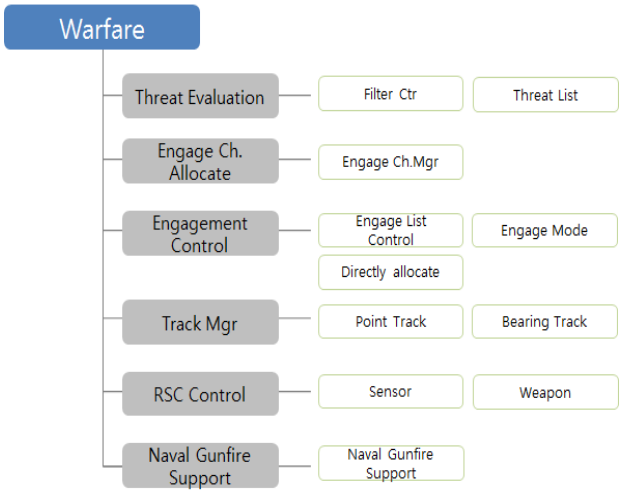

Fig. 3. Modified Warfare SW Architecture

클래스 작성과정에서는 아래의 클래스 작성 5대 원칙인 S.O.L I.D 원칙을 적용하고자 노력하여 가장 적합한 디자인 패턴을 선정 및 적용하고자 노력하였다.[3]

단일 책임 원칙(Single Responsibility Principle)을 적용하여 한 클래스안의 응집도를 높이고 다른 교전 클래스와의 결합도를 낮추어 독립성을 높였다. 이는 클래스를 기능별로 세분화하여 단일기능을 가지도록 하여 재사용 및 유지보수에 유리하도록 하였다.

개방 폐쇄 원칙(Open Close Principle)을 적용하여 센서, 무장의 확장성을 고려한 설계가 되도록 하였다. 하드웨어 추가 시 최소한의 변경으로 기능이 동작하도록 센서, 무장의 속성을 일반화 시키고 그 일반화 된 속성의 값을 파일로 읽어 들일 수 있게 설계하였다. 그리고 각 센서, 무장의 총 개수도 파일로 읽어 들이고 실행될 때 그 값에 따라 동적으로 생성되도록 하여 파일 수정만으로 신규 장비 추가, 기존장비 기능 변경, 장비 제거 등 에도 코드 수정되지 않도록 설계 하였다.

그 외 리스코프 교체의 원칙(Liskov Substitution Principle), 인터페이스 분리의 원칙(ISP : Interface Segregation Principle), 의존 관계 역전의 원칙(Dependency Inversion Principle)을 참고하여 지키고자 노력하였다.

아래의 Table 2 은 기능별 주요 클래스의 세부 역할을 정리한 표이다.

Table 2. Description of Class Diagram

Class Name	Class Description
CWFInterface	Warfare Main Interface Class
CFilterMgr	Threat Filter Management Class
CThreatMgr	Threat List Management Class
CRecChMgr	EngageChanel Recommend Class
CRscMgr	Resource Management Class
CTrackMgr	Track Management Class
CEngageMgr	Engagement Control and EngageList Management Class

작성된 클래스 설계를 바탕으로 만들어진 클래스 다이어그램은 fig. 4 와 같다.

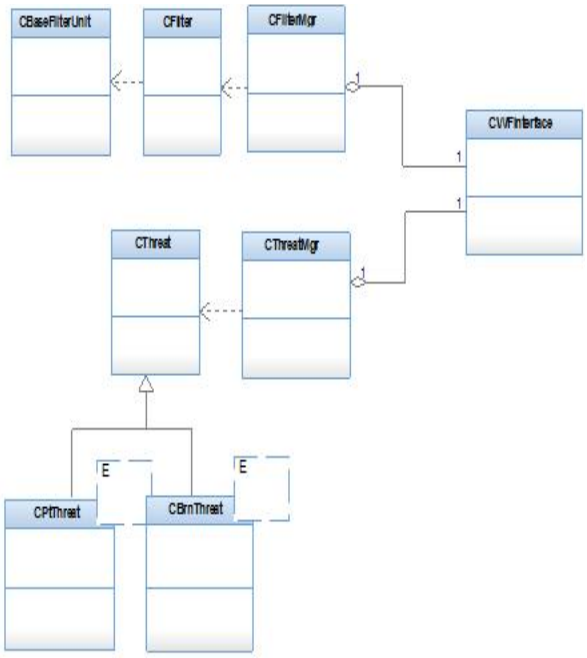

Fig. 4. Proposed Warfare SW Class Diagram
(Filter Management and Threat Management)

Fig. 4에서 기능으로 식별된 필터관리 CSC는 CFilterMgr가 대공, 대함, 대잠의 CFilter 클래스를 성분작전별로 생성하여 식별정보, 운동정보 등의 필터정보를 저장 및 관리하는 역할을 담당하도록 설계하였다. 위협관리 CSC는 CThreatMgr가 CThreat 클래스의 각 위협표적 중에 위협필터를 통과한 위협을 관리하고 위협도를 계산하여 순위를 결정하고 위협목록을 생성 및 전송하는 기능을 하도록 클래스 설계하였다.

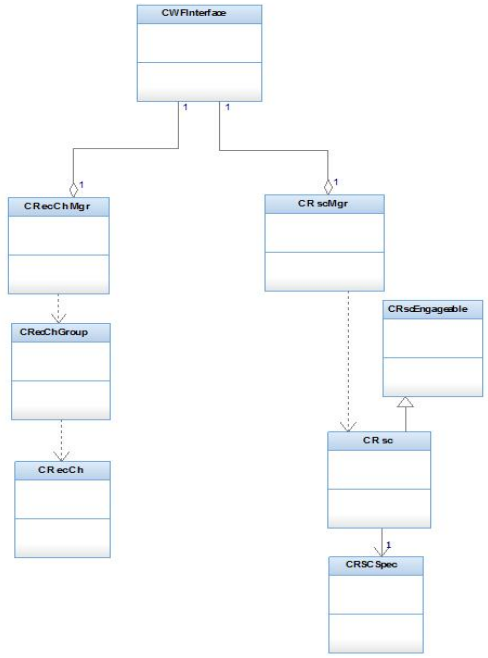

Fig. 5. Proposed Warfare SW Class Diagram
(Resource Management and EngageChanel Recommend)

Fig. 5에서 리소스관리 CSC는 CRscMgr가 각 센서 무장의 객체인 CRsc 클래스를 생성하여 사거리 제한각 등의 스펙을 저장하는 기능을 수행하며 교전채널관리 CSC는 CRecChMgr가 선택된 위협표적에 대한 할당 가능한 무장 또는 센서와 무장의 조합인 CRecChGroup 클래스 교전채널들을 생성 및 관리하고 목록 전송하여 운용자에게 해당위협에 대한 교전채널을 권고하도록 하는 기능을 수행하도록 설계하였다.

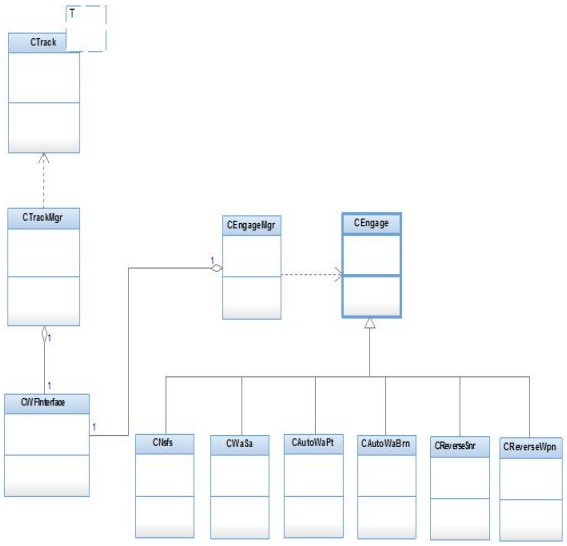

Fig. 6. Proposed Warfare SW Class Diagram
(Track Management and Engagement Control)

Fig. 6에서 표적관리 CSC는 CTrackMgr가 대공, 대함 등 성분작전별 표적정보와 점표적, 방위표적 등 표적 종류에 따라 표적을 저장 및 관리하고 교전관리 CSC는 CEngageMgr가 위

협에 대해 할당된 교전채널을 목록으로 관리하고 목록으로 전송하는 기능을 수행하도록 클래스 설계하였다.

1.2.2 Design Pattern applying

본 논문에서는 소프트웨어 개발 설계과정에서 공통으로 사용할 수 있는 유용한 여러 디자인 패턴들 중 본 과제 적합한 디자인패턴을 분석하고 식별 후 적용 가능한 디자인 패턴을 아래의 Table 3 과 같이 분석, 선정하여 클래스 설계 시에 반영하고자 하였다.

Table 3. Design Pattern of Proposed Warfare

Design Pattern	Description
Abstract Factory	Provide an interface for creating families of related or dependent objects without specifying their concrete classes
Singleton	Restricts the instantiation of a class to one object
Observer	one object changes state, all dependents are notified and updated automatically

적용한 디자인 패턴은 3가지로 생성패턴인 추상팩토리(Abstract Factory), 싱글턴(Singleton)과 행위패턴인 옵저버(Observer) 패턴이다.

추상팩토리(Abstract Factory) 패턴의 적용은 CEngageMgr 클래스에 적용하였다. CEngageMgr 클래스는 함정 전투체계에 연동되는 센서 및 무장을 이용하여 교전을 수행하는 메인 클래스이다. 이 클래스의 변경을 발생시키는 요인은 다음과 같다.

a. 환경정보(대공/대함...)
b. 표적타입(점표적/방위표적...)
c. 교전방법(하드킬/소프트킬)
d. 수동교전
e. 자동교전
f. 반자동 교전
g. 센서 우선 할당 교전
i. 해상화력지원
j. PAC(Pre Action Calibration)
k. 센서만 교전
l. 무장만 교전
m. 센서 무장 조합 교전
n. 센서의 개수
o. 무장의 개수

위의 요인들 즉 a×o의 조합만큼 함정 전투체계의 CEngageMgr 클래스를 변경시키는 요인이 되며 그리고 현재는 없지만 미래에 추가되거나 삭제되는 교전 개념은 CEngageMgr을 변경이 필요하게 한다. 교전 소프트웨어의 변경을 최소화하기 위해 CEngageMgr 및 하위 클래스들은 추상클래스 팩토리 패턴을 사용하여 변경을 최소화 하도록 하였다. 실제 함정의 목적(상륙함/호위함 등)에 따라 각 함정에 연동되어 운용되는 센서 및 무장의 조합은 각각 다르며 함정의 운용목적이 동일하더라고 함정의 크기에 따라 센서 및 무장의 조합도 다르게 설치된다. 이런 이유로 교전 소프트웨어의 변경을 최소화하기 위해 추상팩토리 디자인 패턴을 적용하였다.

자함 정보 COsd클래스처럼 오직 하나만 존재하도록 보장되어야 하는 객체들은 식별하여 싱클턴 디자인 패턴을 적용하여 단일성을 보장하였다.

무장과 센서 교전가능여부 관련 CRscEngageable 클래스처럼 특정 값 변경이 일어날 경우 다른 클래스에 즉시 알릴 필요가 있는 클래스는 옵저버 클래스들을 등록하여 값 변경 시 옵저버 등록된 추종 클래스에서 이벤트 처리가 일어나도록 하였다. 이러한 적용가능한 디자인 패턴을 식별 적용함으로써 코드를 간단명료하고 낮은 알고리즘 복잡도를 가지도록 쉽게 구현할 수 있었다.

1.2.3 Scalable SoftWare Design

제안된 교전 모듈은 추상화를 통한 클래스 식별과 디자인 패턴 적용을 통해 클래스를 세분화하고 서로 의존성을 떼어 냄으로써 신규 무장, 센서의 장비가 추가 되더라도 구동 시 무장, 센서의 속성 및 개수 등의 정보를 파일로 읽어 들이는 것으로 소프트웨어의 직접적인 수정이 일어나지 않도록 하였다. 가변적인 객체의 속성값을 파일로 읽도록 하고 공통요소를 뽑아낸 특성을 클래스 객체화 함으로써 신규장비가 추가되어도 실행 시 읽어 들이는 파일만 수정되도록 하였다. 이렇게 함으로써 확장성은 높이고 유지보수는 더욱 쉽도록 구현 할 수 있었다. 아래의 Fig 7. 의 파일은 제안된 교전이 읽어 들이는 센서, 무장의 속성이 쓰인 파일이다.

resource.ini | WFCalculator.h | WFCalculator.cpp | RscParser.h | RscParser.cpp | RecChParser.h

```
# gfeld     = SUBSYSTEM_ID_SRS, SUBSYSTEM_ID_TRS, ... see ICMSDefine.h
# resourceId = SENSOR_KIND_FC1, SENSOR_KIND_FC2,   ... see WFDefine.h
# Env : 1-Air, 2-Surf, 3-Land,4-Sub
# Kind : 1-Point, 2-Bearing, 3-SpecialPoint

# Sensor Resource ------------------------------------------------------
# SRS-TWS1(FC2)
[resource]
gfeld=127;
resourceId=102;
engEnv=0110; //Surf, Land
engKind=100; // Point
autoAssign=0;
reverseAssign=0;

# SRS-TWS2(FC3)
[resource]
gfeld=127;
resourceId=103;
engEnv=0110;
engKind=100;
autoAssign=0;
reverseAssign=0;
```

Fig. 7. Properties File of Proposed Warfare

1.3 Step 3 : Implementation

구현단계에서는 검증된 기존의 함정전투체계 교전의 위협계산 알고리즘, 표적의 속도와 방위, 교전 사거리 계산 알고리즘 등을 식별하고 이를 최대한 활용하여 구현 시간을 줄이고 오동작 리스크를 줄였다. 다만 계산모듈을 별도의 클래스로 모아 관리하도록 하여 상속 및 관리를 쉽도록 하여 구현에 도움이 되도록 하였다.

다음으로는 함정전투체계 교전에서 사용하는 자료구조 들을 식별하여 효율성을 분석하고 최대한 검증된 STL을 적용하여 구조와 사용에 맞으면서도 안정적이고 더 나은 성능을 내도록 적용하였다. 위협정보, 표적정보, 교전목록정보 같이 많은 정보들을 관리하며 키값으로 바로 접근할 필요가 있는 정보는 MAP

을 활용하여 관리하도록 변경하였다. 센서, 무장 등의 자원관리 자료구조는 MAP과 LIST를 활용하여 구현하여 효율을 높였다.

1.4 Step 4 : Test & Debug

테스트 & 디버그 는 제안된 함정 전투체계 교전 모듈에 특정 입력 값을 주고 출력 값을 확인 하는 단위테스트를 거친 후상용 신뢰성 툴을 이용하여 소프트웨어의 신뢰성을 검증하고 전체 시뮬레이터 환경에서 최종 동작함을 확인하는 순으로 진행하였다.

제안된 함정 전투체계 교전 모듈의 입출력 값을 확인하기 위해 메시지 시뮬레이터를 활용하여 입출력 기능이 제대로 동작하는지를 확인하였다. 시뮬레이터는 함정전투체계 교전과 관련 기능 확인 및 관련 정보가 전시 되도록 구현된 별로의 시뮬레이터 프로그램을 개발하여 사용하였다.

아래의 Fig. 8. 는 개발된 교전 시뮬레이터 프로그램 실행화면이다.

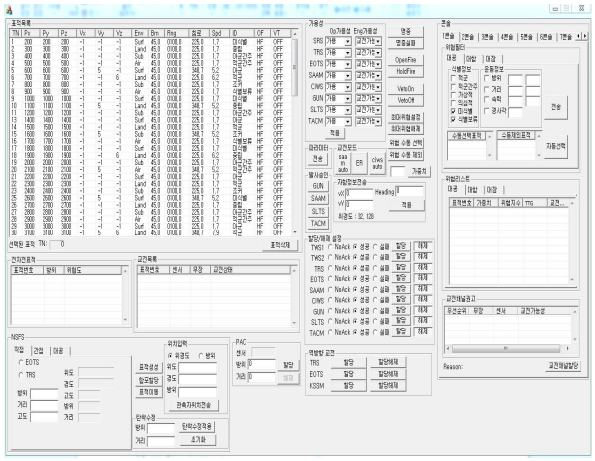

Fig. 8. Warfare Message Simulator

소프트웨어 신뢰성에 대한 검증을 위해 정적시험, 동적시험을 상용 SW 툴로 수행하였다. 사용된 툴은 정적시험 QAV, Codesonar 툴이고 동적시험에는 Codescroll 툴이다.

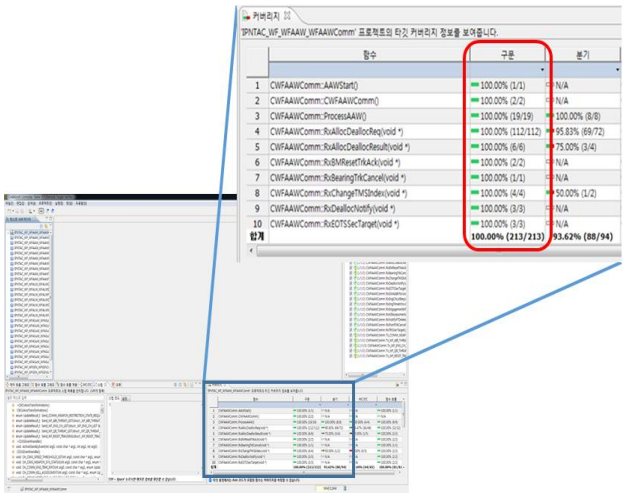

Fig. 9. Dynamic Test SW - codescroll

QAV와 Codesonar 툴로 검출된 소프트웨어 코딩 룰에 어긋나거나 잠재적 위험이 있는 오류항목은 수정 후 재빌드하여 완전제거 하였고 위의 Fig. 9. Codescroll 동적신뢰성 시험 툴을 통하여서 불필요한 코드 구문과 입출력 값 확인을 통한 실시간 발생할 수 있는 동적인 오류를 제거하였다.

제안된 함정전투체계 교전 소프트웨어는 최종적으로 센서 무장 등의 시뮬레이터 개발환경이 구축된 실제 특정 함정전투체계 개발 환경에서 테스트를 진행하였다. 기존 전투체계 시험 절차서를 그대로 시험에 활용함으로써 기존 교전 소프트웨어와 동일한 기능을 보유하고 있음을 검증하였다.

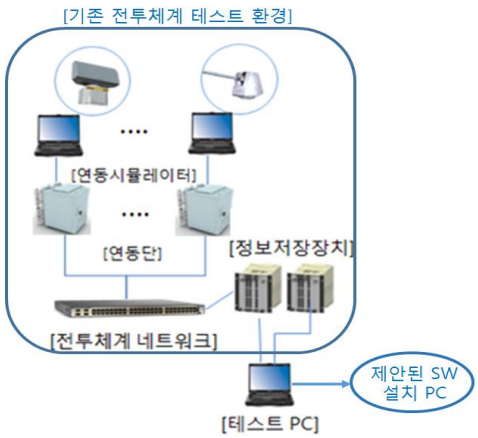

Fig. 10. Test environment

IV. Performance Analysis

1. Proposed Warfare performance Analysis

제안된 함정전투체계 교전 소프트웨어의 성능검증은 신규 장비 연동 시 수정이 필요한 클래스 수 비교, 신뢰성시험과 유지보수 등에 연관이 있는 코드의 총 라인수, 코드 복잡도 분석, 실행성능과 부하테스트를 위한 CPU 사용률로 확인하였다. 정확한 성능 비교를 위해 기존의 OOO사업 교전 전투체계를 비교 군으로 하고 제안된 교전모듈이 얼마나 향상 되었는가를 평가하였다.

1.1 Scalable and maintenance

코드 확장성 분석은 제안된 교전과 OOO 교전에 소요로 인하여 신규 센서가 추가 연동됨을 가정하고 코드 수정이 필요한 클래스 수를 비교하는 것으로 확인 하였다. 신규 장비는 향후 함정 전투체계에 추가 될 예정인 다기능 레이더 (Multi-

Function Radar)로 가정하고 기능분석 후 변경되는 내용에 따른 기존 전투체계 교전과 제안된 교전의 수정이 필요한 클래스를 비교하였다.

Table 4. Number of Modified Class by additional Equipment

	Warfare	Proposed Warfare
Number of Modified Class	7	0
Number of Modified file	0	1

다기능 레이더가 추가 될 경우 OOO 교전은 메시지 수신단의 교전 메인 클래스를 비롯하여 작전성분별로 메인 클래스, 교전채널을 권고하는 클래스, 센서 무장의 교전 가용성관리 클래스가 필수적으로 수정되어져야 한다. (MFR은 대공, 대함 모두 탐지 가능함으로 작전성분별 클래스 * 2) 하지만 제안된 교전은 Resource.ini 파일만 수정하면 해당 속성과 센서 개수의 추가가 이루어지기 때문에 클래스 수정이 필요 없다. 이러한 이점은 유지보수뿐만 아니라 코드 수정으로 신뢰성 시험을 다시 수행해야하는 번거로움도 줄여준다.

※ 제안된 교전은 위의 경우처럼 기 식별된 장비의 속성 외에 새로운 속성이 도입될 경우에는 해당 장비 클래스만 추가 수정이 일어나게 된다.

1.2 Code Line

먼저 제안된 교전과 OOO 교전의 코드 라인수를 비교하여 보았다. 코드 라인수는 주석과 빈 라인을 제외한 순수 코드 라인수만 측정하였다. 코드 라인수 결과는 아래의 Table 5 와 같다.

Table 5. Code Compare of Proposed Warfare and Warfare

SW	Class Num	Total line Num
Warfare SW	33	78386
proposed Warfare SW	33	10337

위의 표와 같이 제안된 교전은 기존의 OOO 교전보다 라인수가 86% 줄어들었고 총 클래스 수는 같다. 클래스 수는 작전성분별로 독립적으로 구성되어있던 유사기능을 하는 클래스가 통합되어 많이 줄어들었으나 기능별 객체중심으로 세분화되어 총 수에서는 변화가 없었다. 전체 라인수는 작전성분별 분리되어있던 메시지 송수신부, 유사기능이 통합되어 기존 교전에서 코드 길이가 14% 수준으로 줄어들었다.

1.3 Code Complexity

코드 복잡도 분석은 클래스 별 각 함수의 복잡도를 측정하여 수치로 나타낸 것으로 복잡도 수치별로 취합하여 비교하였다. 코드 복잡도는 낮을수록 성능 및 신뢰성 시험 등의 수행에 유리하다. 복잡도 측정은 QAC 상용 SW의 복잡도 측정기능으로 측정하였다. 복잡도 계산 공식은 “Cyclomatic Complexity = 분기문 개수 + 1” 이고 제안된 교전과 OOO 교전의 코드의 복잡도 측정 결과는 아래 Table 6 과 같다.

Table 6. Code Complexity of Warfare and Proposed Warfare

Complexity / Software	5~9	10~14	15~19	20 ↑
Warfare	127	37	29	16
Proposed Warfare	44	3	2	7

위의 표를 보면 기존의 교전의 보다 복잡도가 높은 함수의 수가 많이 준 것을 확인 할 수 있었다. 복잡도 20이상에서는 기존 교전의 44% 수준으로 낮게 나타났으며 해당 함수 검토결과 CRSC, CWFInterface 관련 클래스 에서 파일 읽어 들이는 함수와 장비별 체계파라미터 저장 함수, 메시지 콜백 함수 등 특정 식별자별로 작업이 반복 구현되어 함수길이가 불가피하게 길어진 함수에서 나타났다. 복잡도 15~20은 기존의 7% 수준10~14 사이에서는 기존의 8% 수준, 5~9 사이에서는 35% 수준으로 낮아져 제안된 교전이 OOO 교전의 복잡도와 비교해 상당히 개선되었다. 이는 객체화, 세분화, 디자인패턴적용, STL 적용 등으로 클래스 내의 함수 복잡도가 낮아지고 개선된 결과이다.

1.4 CPU Usage Rate

제안된 교전과 OOO 교전의 CPU 점유율 비교는 최대 표적 환경에서 측정하였다. OOO 교전은 5개의 모듈로 구성되어있어 실행파일이 5개이기 때문에 CPU 점유율 값은 5개의 모듈을 합친 값으로 표현하였다. 보다 정확한 측정을 위해서 10분 단위로 시간을 두고 측정하여 변화되는 수치를 확인 하였다. 테스트 환경과 측정값은 아래의 Fig. 11. 에 정리하였다.

※ 테스트 환경

CPU: 2.53GHz, RAM 4GB, HDD:128G, OS: Linux, 최대 표적 환경(약 4000개의 표적)

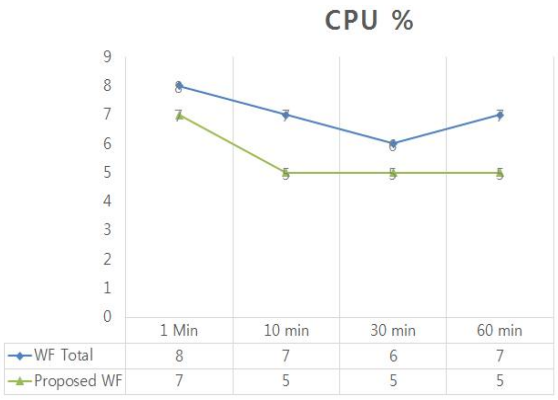

Fig. 11. Test Result for Processing Speed

OOO 교전과 제안된 교전의 CPU 점유율은 약 6%~8 와 5~7% 로 차이가 거의 없었으나 제안된 교전이 CPU를 조금 덜 점유함을 확인 할 수 있었다. 작전성분별로 나눠져 있었던 표적정보, 자함 정보, 장비상태 정보 수신 단이 통합되고 정보관리

를 STL 로 구현 등으로 통신 노드수가 줄어들고 효율이 좋아지면서 개선된 것으로 보인다.

제안된 교전은 OOO 교전과 비교하여 신규 장비연동에 대한 코드수정이 없거나 특정 클래스에 한정되어 확장성과 유지보수성이 뛰어나며 디자인 패턴 적용 및 객체화 설계로 코드 라인수도 14% 수준으로 줄어들었고 함수 복잡도도 20 이상이 44%수준으로 10이상은 15% 수준으로 낮아져 더욱 안정적이며 효율적이다. CPU 점유율도 1~2% 정도 소폭 개선되어 되었다. 이를 종합적으로 볼 때 제안된 함정 전투체계 교전은 확장성을 가지면서도 유지보수성 뿐만 아니라 성능 면에서도 향상된 것으로 확인되었다.

V. Conclusions

본 연구를 통해 제안된 함정전투체계 교전 소프트웨어를 통해 신규 무장, 센서의 추가, 장비의 기능변경 등에도 소프트웨어 수정을 최소화하면서 유지보수는 더욱 용이하게 할 수 있는 교전 소프트웨어를 개발할 수 있었다. 향후 함정전투체계 내의 다른 소프트웨어도 개선하는데 본 연구가 도움이 될 것이며 향후 유지보수 측면에서의 시간 및 비용 절약과 최근 중요성이 부각되고 있는 소프트웨어 신뢰성 시험수행과 관련된 측면에서의 효용성 대한 연구도 추가로 진행할 예정이며 앞으로 함정전투체계 개선에 대한 연구를 계속해서 해나갈 예정이다.

REFERENCES

[1] Gamma Erich “Design Patterns: Elements of Reusable Object-Oriented Software” Pearson Education India, 1995.

[2] Wolfgang, Pree. “Design Patterns for Object -Oriented Software development” Reading Mass, 1994.

[3] Robert Cecil Martin “Agile Software Development: Principles, Patterns, and Practice” Prentice Hall, 2002.

[4] Ki-tae Kwon “A Study on the Standardization of the combat system software in battleship using Object-oriented Design” Korea Society of Computer and Information 2017, 296-297

[5] SOON JOO KO “An Examination on Overseas Technology Trend and Domestic Development Pattern of the Naval Combat Management System”, Korea Association of Defence Industry Studies 2009.

[6] Bohner S. A. and Arnold R. S., "Software Change Impact Analysis," IEEE Computer Society Press, 1996.

[7] IEEE Computer Society SWEBOK Team, Guide to the Software Engineering Body of Knowledge (SWEBOK), IEEE, 2004.

[8] Aspect-oriented programming, Gregor KiczalesJohn LampingAnurag MendhekarChris MaedaCristina LopesJean-Marc LoingtierJohn Irwin, ECOOP 1997: ECOOP'97 Object-Oriented Programming pp 220-242, 23 May 2006

[9] Kyu-Jung Han, Chi-Su Kim and Kyung-Whan Lee “A Design of OOPT system for object oriented program testing”, KOREA INFORMATION SCIENCE SOCIETY 1993.10, 859-862 (4 pages)

[10] Dae-Yeob Kim, Cheong Youn “Methodology for Traceability Management and Impact Analysis for Efficient Change Management in Object-Oriented Development”, KOREA INFORMATION SCIENCE SOCIETY 2015.03, 328-340 (13 pages)

[11] Won Young Lee and Eun Man Choi “A Tool generating Object-Oriented Models from C++ Programs”, KOREA INFORMATION SCIENCE SOCIETY 1997.9, 948-957 (10 pages)

[12] Kim, Young-Gyu, Yang, Hae-Sool and Choi, Hyung-Jin “Framework Model for Software Productivity Enhancement In Object-Oriented Environment”, Korea Academy Industrial Cooperation Society 2008.12, 1678-1689 (12 pages)

Authors

Ki-Tae Kwon received the B.S. degrees in Computer Engineering, Kyungpook National University, Korea, in 2009 He is currently working in Hanwha Systems Co. from 2009. He is interested in Combat System Software, weapon system and Information processing Algorithm.

Ki-Pyo Kim received the M.S degree in Electrical Engineering and Computer Science from Gwangju Institute Science Technology, Korea in 2004. He is interested in warfare and track management software of the naval combat management system, parallel computing, interactive genetic algorithm and graph theory.

HwanJun Choi received the B.S and M.S. degrees in Information and communication system from Sejong University, Korea, in 2013 and 2015 respectively He is interested in Combat System Software, warefare module.