JKSCI

한국컴퓨터정보학회논문지
Journal of The Korea Society of Computer and Information
Vol. 25 No. 2, pp. 133-140, February 2020
https://doi.org/10.9708/jksci.2020.25.02.133

A Study on IISS Software Architecture of Combat Management System for improving modifiability

Ji-Yoon Park*, Moon-Seok Yang*, Dong-Hyeong Lee*

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

[Abstract]

The IISS(Integrated Interface Storage System) software uses communication methods such as DSS(Data Sharing Service), UDP to perform the function of sending all messages from the Combat Management System to the analytical computer. Because IISS software handles all message used in the Combat Management System, the source code is large and has a highly dependent feature on message changes. Modification of software is a task that requires a lot of labor, such as series of software reliability test. so research has been conducted to reduce software development costs, including minimizing software modifications. In this paper, We study the method of messages receiving and architectural structure improvement to minimize reliance on message changes in the Combat Management System and improve the modifiability. Reduced message dependency by changing the way DSS and UDP protocols are communicated to Packet Sniffing. In addition, Factory Method Pattern were applied to improve the software design. Test comparing existing software and development elements have confirmed that the software has improved its modifiability and reuse.

▸ **Key words**: Combat Management System, Software modifiability, Design Pattern, Factory Method Pattern, Packet Sniffing

[요 약]

정보저장 소프트웨어는 DSS(Data Sharing Service), UDP와 같은 통신 방식을 사용하여 전투관리체계에서 송/수신되는 모든 메시지를 분석컴퓨터로 전송하는 기능을 수행한다. 정보저장 소프트웨어는 전투관리체계에서 사용되는 모든 메시지를 처리하기 때문에 소스코드의 규모가 크며 메시지 변화에 의존성이 강한 특성을 가진다. 소프트웨어의 수정은 연쇄적으로 소프트웨어 신뢰성 시험과 같은 많은 노동력을 요구하는 작업이 발생하기 때문에, 소프트웨어 수정 최소화를 비롯한 소프트웨어 개발 비용 절감을 위한 연구가 지속적으로 이루어지고 있다. 본 논문에서는 전투관리체계 정보저장 소프트웨어의 메시지 변화에 대한 의존성을 최소화하고 수정 용이성의 향상을 위한 메시지 수신 방식 및 아키텍처 구조 개선 방안을 연구하였다. DSS와 UDP 프로토콜을 통하여 메시지를 송/수신하던 기존의 방식을 Packet Sniffing으로 변경함으로써 메시지에 대한 의존성을 줄였으며 팩토리 메소드 패턴(Factory Method Pattern)을 적용하여 소프트웨어 설계를 개선하였다. 기존 소프트웨어와 개발 요소를 비교하는 시험을 통해 소프트웨어의 수정 용이성과 재사용성이 향상 된 것을 확인하였다.

▸ **주제어**: 전투관리체계, 소프트웨어 재사용, 디자인패턴, 팩토리 메소드 패턴, 패킷 스니핑

• First Author: Ji-Yoon Park, Corresponding Author: Ji-Yoon Park
*Ji-Yoon Park (jiyoon15.park@hanwha.com), Naval R&D Center, Hanwha Systems
*Moon-Seok Yang (moonseok.yang@hanwha.com), Naval R&D Center, Hanwha Systems
*Dong-Hyeong Lee (dh0104.lee@hanwha.com), Naval R&D Center, Hanwha Systems
• Received: 2019. 11. 27, Revised: 2020. 01. 16, Accepted: 2020. 01. 19.

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

I. Introduction

함정 전투관리체계(CMS, Combat Management System)는 전장상황인식을 위해 센서체계로부터 수신한 다량의 정보를 정보처리장치에 전달하여 위협을 식별하고 무장에 전술 명령을 전달하여 교전을 수행하고 있다.[1] 복합적이고 동시 다발적으로 발생하는 전장 상황에 대한 대응 능력의 요구가 증대됨에 따라 네트워크를 통해 전달되는 정보가 폭발적으로 증가되었고 전장상황을 분석하고, 강평하기 위한 데이터량도 동반하여 증가하였다.

함정 전투관리체계의 소프트웨어 중 정보저장(IISS, Integrated Interface Storage System) 소프트웨어는 전투관리체계 데이터버스(CSDB, Combat System Data Bus)망을 통해 전달되는 모든 DSS(Data Sharing Service) 메시지와 연동장비 메시지를 수신하여 저장하고, 상황을 분석하고 강평하기 위하여 메시지들을 분석컴퓨터로 송신하는 소프트웨어이다. 정보저장 소프트웨어는 DSS 통신을 통해 전달되는 모든 메시지를 수신하고 처리하기 때문에 DSS 메시지 및 연동 메시지의 변경에 따라 소스코드 수정이 빈번하게 발생하며, 이러한 의존성은 개발 비용의 증가를 야기하게 된다. 특히, 소스코드의 수정은 소프트웨어 정적시험(MISRA[2], Code Sniper[3]) 및 동적시험(Quality Scroll Cover[4])이라는 소프트웨어 신뢰성 시험을 수행해야한다. 소프트웨어 수정 및 신뢰성 시험의 수행은 소프트웨어 개발자의 숙련도, 소스코드 규모 및 난이도, 보조도구 사용여부, 신뢰성시험 검증 툴의 라이센스 수 등 업무 수행에 영향을 미치는 요소를 지니고 있으며 사람에 의한 작업으로 인적 오류의 잠재적인 요소를 가진다.[5]

본 논문에서는 메시지의 변경에도 소스 코드 수정을 최소화하고 수정 용이성을 향상시키기 위하여 정보저장 소프트웨어에 Packet Sniffing 방식과 팩토리 메소드 패턴(Factory Method Pattern)[6]을 적용하여 재설계하였으며 기존 소프트웨어와 비교한 결과를 기술하였다.

본 논문의 구성은 다음과 같다. 2장에서는 관련연구로써 메시지 네트워크 기반의 함정 전투관리체계와 기존 정보저장 소프트웨어 문제점에 대해 설명하여 연구의 목적을 언급하고, 3장에서는 Packet Sniffing 방식과 디자인 패턴을 적용한 정보저장 소프트웨어 아키텍처 설계에 대해 설명한다. 4장에서는 본 논문에서 제안하는 정보저장 소프트웨어와 기존 소프트웨어의 개발 요소 비교를 확인한다. 마지막으로 5장에서 결론으로 논문을 마무리한다.

II. Preliminaries

1. Related works

1.1 Naval Combat Management System

전투체계는 전투관리체계를 중심으로 무장체계, 센서체계 및 통신체계로 구성되어 있으며, 본 논문에서 개선 연구를 실시한 정보저장장치는 전투관리체계의 구성품의 일부이다. 전투체계의 전체 구성도는 Fig. 1과 같다.[6]

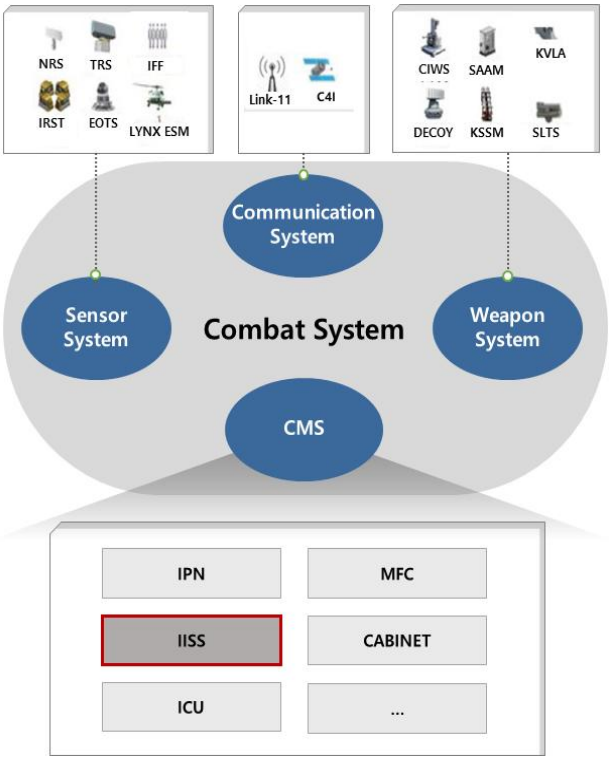

Fig. 1. Combat System

무장체계와 센서체계 및 통신체계는 전투관리체계와 직접적으로 연결되어 있으며, 각 무장, 센서 장비 및 통신 체계에서는 이더넷, 시리얼 등 다양한 프로토콜을 통하여 전투관리체계의 연동단과 통신한다. 정보저장 소프트웨어는 이 모든 통신 메시지와 전투관리체계 내에서 발생하는 DSS 및 UDP 프로토콜을 통한 메시지를 모두 수신하여 분석컴퓨터로 송신한다.

무장이나 센서의 변경, 통신체계의 변경 및 기존 장비의 기능 개선에 의한 메시지 변경이 발생하게 되는 경우, 정보저장 소프트웨어에 수정 작업이 발생하게 된다.

1.2 Software development with DSS

DSS 통신은 Fig. 2와 같은 DDS(Data Distribute Service) 미들웨어의 Publication/Subscribe 형식을 사용한다.[8-9]

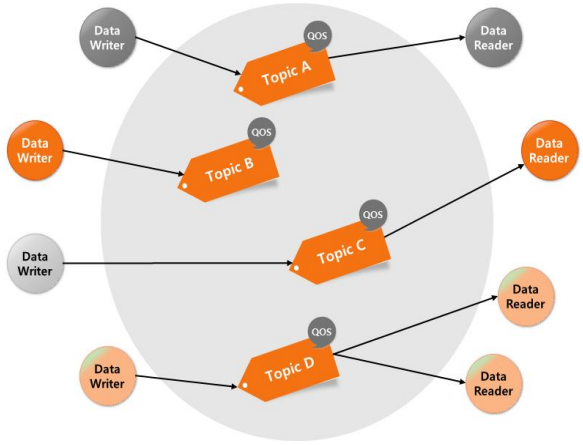

Fig. 2. Publication/Subscribe of DDS Middleware

함정 전투관리체계의 소프트웨어는 DSS 통신을 기반으로 구현되었으며 이를 사용하기 관련된 클래스를 기본적으로 포함하고 있다. 그러므로 메시지 변경의 발생 시 이와 DSS 관련 클래스는 반드시 수정되어야한다. 또한 DSS 통신을 사용하기 위해서는 별도의 NodeAgent라는 응용프로그램이 설치되어야 하며 메시지 변경 시 NodeAgent도 재빌드 및 설치가 필요하다.

1.3 Packet Sniffing

Packet Sniffing은 네트워크 상에 전송되는 각 패킷을 수집하는 방법이며 구조는 Fig 3.와 같다.

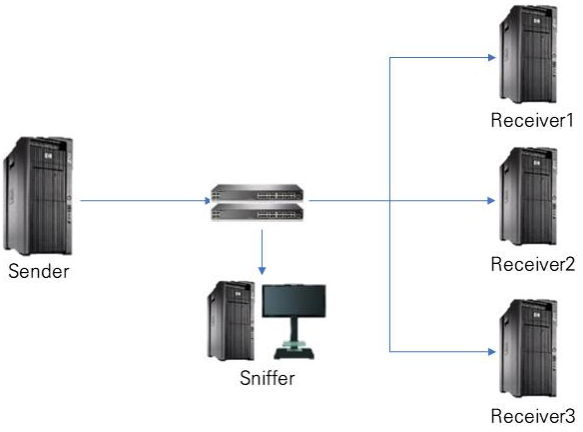

Fig. 3. Packet Sniffing

Packet Sniffing은 사용자가 속한 네트워크에서 다른 사용자가 송수신한 데이터를 Sniffing 하기 위한 기술이다. 이런 Sniffing을 할 수 있는 도구를 Sniffer라고 하며 네트워크 상의 실제 데이터를 수집하고 해석하는 과정이다. 아래 Fig 4는 이더넷 프레임의 포맷을 나타낸다.[10-12]

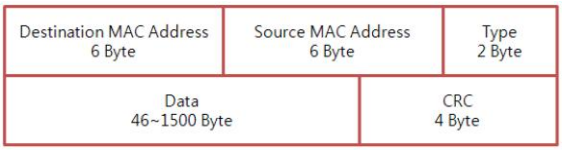

Fig. 4. Format of Ethernet Frame

정보저장 소프트웨어에서는 캡처한 Packet의 Data 영역에 별도의 메시지헤더를 추가하여 분석컴퓨터에 전송하도록 한다.

2. Purpose of Study

2.1 Behavior of Exist IISS software

함정 전투관리체계에서 운용하고 있는 기존 정보저장 소프트웨어의 동작은 Fig. 5와 같다.

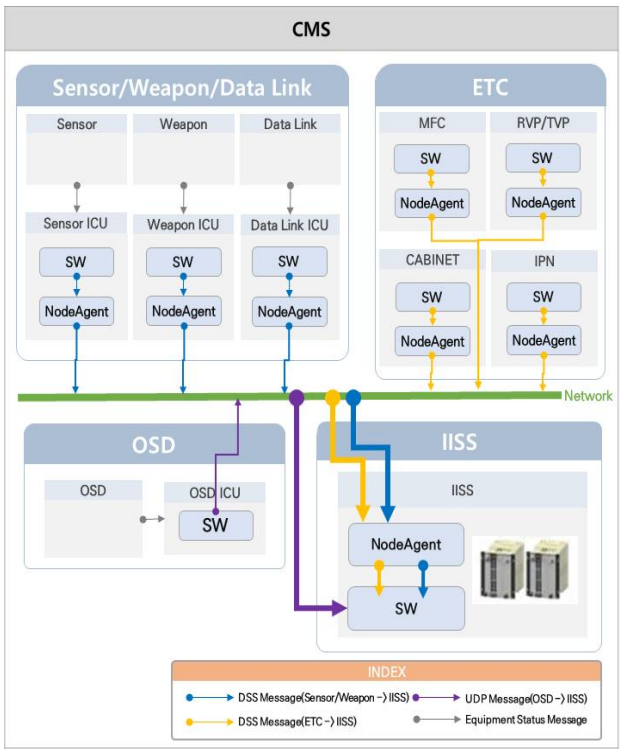

Fig. 5. Behavior of existing IISS Software

기존 정보저장 소프트웨어에서 처리하는 메시지는 아래와 같이 분류할 수 있다.

- 연동장비로부터 받은 장비상태를 DSS로 송신한 메시지
- 전투관리체계의 운용, 전술 지휘를 위해 DSS로 송신한 메시지
- 함기준센서에서 연동장비로부터 수신한 장비상태를 UDP로 송신한 메시지

DSS 메시지는 NodeAgent를 통하여 전투관리체계 데이터버스에 전송되고 각 장치의 NodeAgent에서 소프트웨어로 DSS 메시지를 전달하여 준다. 함기준센서 연동단의 경우 별도의 UDP로 송신하기 때문에 NodeAgent를 거치지 않고 바로 소프트웨어에서 메시지를 수신한다.

2.2 Expectation Effectiveness

본 논문에서는 메시지 변화에 대한 정보저장 소프트웨어의 영향성을 최소화하기 위한 연구를 수행하였고, 기대효과는 다음과 같다.

- DSS 통신 모듈인 NodeAgent 재구성 불필요
- 정보저장 소프트웨어 소스코드 수정 최소화
- 소프트웨어 신뢰성 시험 범위 최소화

DSS 통신 메시지 처리에 대한 개선을 위하여 Packet Sniffing 기법 적용하였고, 소스코드 수정 최소화를 위하여 팩토리 메소드 패턴을 적용한 소프트웨어 아키텍처의 구조를 개선하였으며, 더불어 IDL 설정 파일 적용을 최대화하는 방안을 적용하였다. 소스코드 수정의 최소화는 소프트웨어 신뢰성 시험 수행에 대한 범위를 축소시키고 재사용율을 높이는 이점을 가진다.

III. The Proposed Scheme

1. Software applying Packet Sniffing

정보저장 소프트웨어에서 관리하는 모든 메시지를 통신방식에 상관없이 수신하기 위하여 Packet Sniffing을 적용하였다. 네트워크의 모든 트래픽을 수집할 수 있는 장점을 가지고 있어 전투관리체계 데이터버스를 지나는 모든 트래픽을 전송받아야하는 정보저장 소프트웨어에 필요한 기술이다. 본 논문에서 제안하는 전투관리체계 정보저장 소프트웨어 동작의 전체 구조는 다음 Fig. 6와 같다.

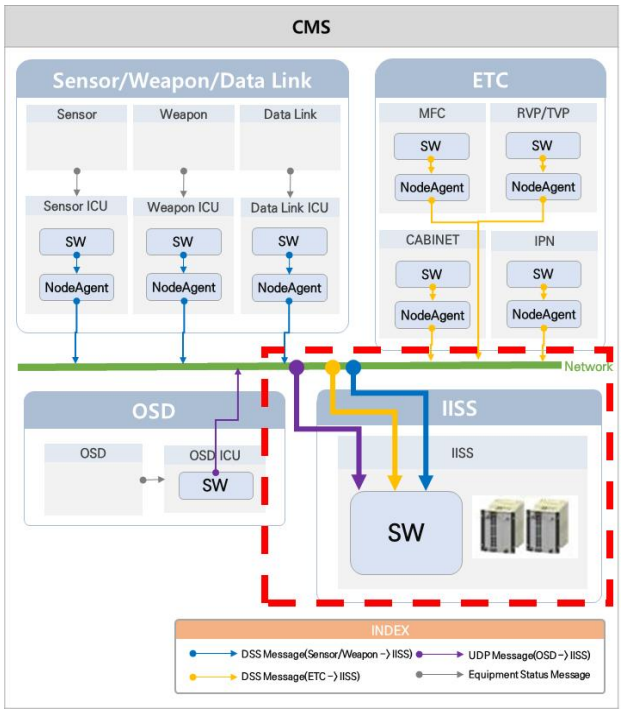

Fig. 6. Behavior of proposed IISS Software

기존 NodeAgent를 통하여 DSS 메시지를 수신하고 별도의 UDP 통신으로 함기준센서와 직접 통신했던 구조에서 Packet Sniffing 방안을 적용하여 NodeAgent나 UDP 통신 소켓 없이도 전투관리체계 데이터버스의 모든 메시지를 수집하도록 하였다. 정보저장 소프트웨어에서는 DSS 메시지를 위한 RTPS 프로토콜의 메시지와 함기준센서 연동단에서 송신하는 UDP 프로토콜의 메시지를 Packet Sniffing을 통해 소프트웨어에서 직접 Packet을 수집한다. 수집한 Packet은 메시지 종류에 따라 각기 다른 수신처리를 수행한다. 센서,무장,데이터링크 장비 상태메시지에 DSS 헤더를 씌워 송신한 장비상태 메시지와 전투관리체계의 전술,지휘 운용을 위한 DSS 메시지, 그리고 함기준센서에서 바로 전송하는 UDP 메시지에 따라 나뉘어 메시지를 처리한다.

DSS 메시지의 경우 Fig 7, Table 1.와 같은 메시지 구조를 가진다. 이더넷 프레임의 데이터 영역에서 DSS 메시지를 획득하고 DSS 메시지 헤더 영역에서 Message ID를 통해 메시지를 식별하고 DSS 메시지 헤더영역을 제외한 Message Data 영역에서 파일에 저장하고자 하는 실제 데이터의 값을 받아온다.

Message Header	Message Data

Fig. 7. DSS Message Configuration

Table 1. DSS Message Header Configuration

Data	Type	Description
Message Header	Message Size	Total Message Size(Include DSS Message Header)
	Message ID	Message ID
	Domain ID	Domain ID(Tactical, Training)
	Mode ID	Mode ID(Tactical, Training)
	...	...

2. Architecture Design with Design Pattern

기존 정보저장 소프트웨어의 Class Diagram과 Class 설명은 아래 Fig. 8, Table 2와 같다.

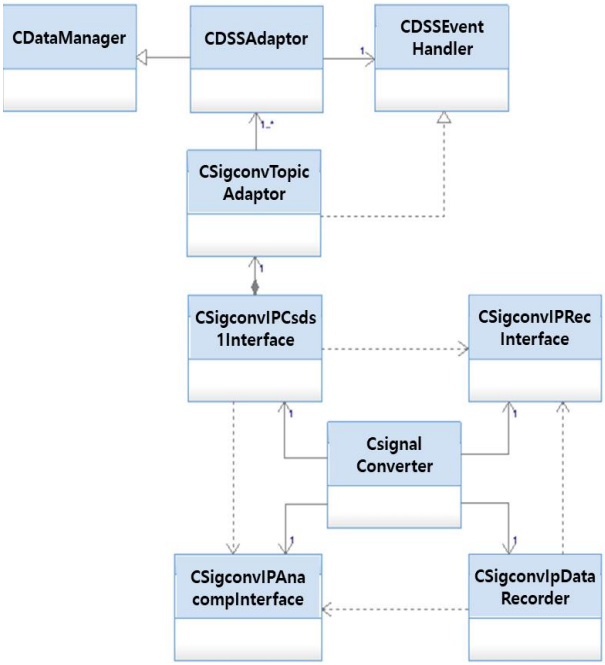

Fig. 8. Existing IISS Software Class Diagram

Table 2. IISS Software Class Description

Class	Description
CDataManager	Class for DSS Communication
CDSSAdaptor	
CDSSEventHandler	
CSignalConverter	Main Class
CSigconvTopicAdaptor	Class for Message Receving
CSigconvIPCsds1Interface	Class for Message Processing
CSigconvIPRecInterface	Class for File Writing
CSigconvIPDataRecorder	
CSigIPAnacompinterface	Class for Analysis PC and Interface

CDataManager, CDSSAdaptor, CEventHandler 클래스는 DSS 메시지별 초기화와 메시지별 수신을 위한 Listener 할당, 메시지별 수신 함수 선언 및 정의가 구현되어 있다. 메시지별로 기능이 구현되어 있기 때문에 메시지의 변경에 따라 위의 3개의 클래스는 소스코드를 반드시 수정해주어야 한다. 또한 현재 구조에서는 통신 방식의 구분없이 CSigconvTopicAdaptor 클래스와 CSigconvIPCsds1Interface 클래스에서 모든 메시지를 수신 및 처리하도록 구현되어 있어 소프트웨어 코드 관리에 비효율적인 구조를 가진다. 앞서 설명한 메시지 3가지에 따라 클래스를 나누고 클래스의 구조를 계층화한다면 소프트웨어의 가변 요소를 줄일 수 있는 이점이 있다. 본 논문에서는 정보저장 소프트웨어의 설계과정에서 적용 가능한 디자인 패턴들 중 팩토리 메소드 패턴(Factory Method Pattern)을 적용하였다. 팩토리 메소드 패턴은 객체 생성 처리를 서브클래스로 분리해 처리하도록 캡슐화하여 특정 기능의 구현은 서브클래스를 통해 제공할 수 있는 패턴이다. 아래 Fig. 9는 팩토리 메소드 패턴을 적용한 Packet Sniffing 방식의 정보저장 소프트웨어의 Class Diagram이다.

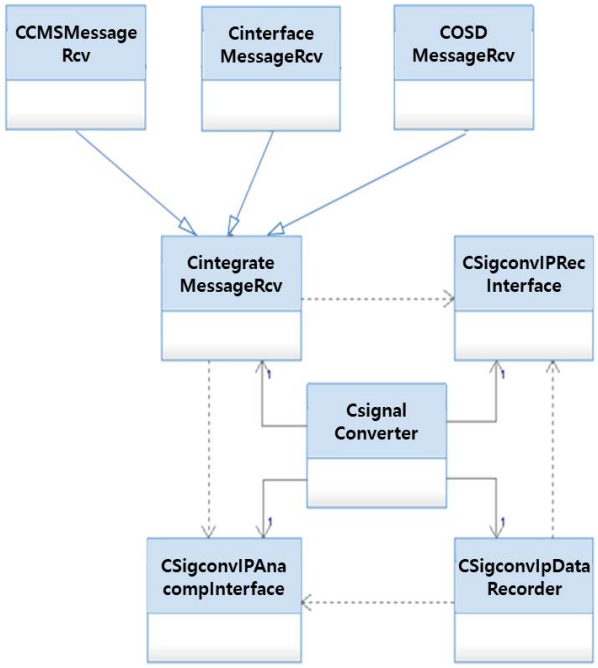

Fig. 9. IISS Software Design with Factory Method Pattern

정보저장 소프트웨어는 연동장비와 전투관리체계로부터 수신한 메시지에 시간 정보를 추가한다. 수신된 메시지는 로그파일 형태로 저장하여 운용자가 이해하기 쉬운 데이터 형식으로 관리한다. 메시지를 Unpacking하는 과정과 새로운 헤더를 추가하고 파일에 저장하는 과정이 필요하지만 다른 포맷의 메시지를 수신하기 때문에 그 처리는 메시지마다 달라질 수 있다. Fig. 9에서 수퍼클래스는

CIntegrateMessageRcv 클래스 이다. Packet Sniffing으로 캡처한 메시지를 A, 분석컴퓨터 전송을 위한 메시지를 B라고 할 때, CIntegrateMessageRcv 클래스는 서브클래스 생성에 대한 인터페이스와 메시지 A의 Unpacking, 그리고 메시지 B의 가공에 대한 추상메소드를 제공한다. 수퍼클래스인 CIntegrateMessageRcv 클래스를 상속받은 서브클래스 CCMSMessageRcv, CInterfaceMessageRcv, COSDMessageRcv는 추상메소드에 생성하고자하는 메시지 B를 일련의 과정을 통해 생성하여 리턴한다. 이런 방식으로 서브클래스에서는 전달된 메시지를 분석컴퓨터를 위한 메시지 B로 가공할 수 있다. 팩토리 메소드 패턴을 적용한 설계로 메시지에 대한 의존성 및 영향성을 줄이고 빈번한 소프트웨어의 수정과 빌드 및 설치와 같은 개발 시간 단축시킬 수 있다.

정보저장 소프트웨어에서는 연동단의 메시지, 전투관리체계의 메시지, 함기준센서의 메시지에 따라 서브클래스를 구성하였고 만약 신규 장비로 인해 새로운 인터페이스가 추가될 경우 CIntegrateMessageRcv클래스를 상속받는 클래스와 인스턴스만 추가하면 되므로 소프트웨어 변경용이성이 높다.

3. Configuration File

정보저장 소프트웨어의 코드 수정 최소화를 위한 소프트웨어 설계를 위해 큰 가변 요소인 메시지 구조 정의 부분은 별도의 IDL 파일로 분리하였다. 정보저장 소프트웨어의 초기화 시 IDL 파일을 읽어 메시지의 구조를 로컬데이터로 저장함으로써 메시지 추가 및 수정에 대하여 소프트웨어 소스코드의 수정이 최소화 할 수 있다. 아래 Fig. 10은 정보저장 소프트웨어에서 읽어들이는 IDL 파일의 구조이다.

```
//-----------------------------------------------------
// Power Control Message
//-----------------------------------------------------

struct HCI_PWR_CMD {
        SMSG_HEADER stMessageHead;
        SCONSOLEPOWER stConsolePower;
        SVOYAGECONSOLE stVoygeConsole;
        SSIGHANDLINGCABINET stSigHandlingCabinet;
        SDIRCONTROLCABINET stDIRControlCabinet;
        SSONARSENSOR stSonarSensor;
        SHORFIRINTERLOCK stHORFirInterlock;
        SUNDFIRINTERLOCK stUNDFirinterlock;
        SONESELFHANDLING stOneselfHandling;
        EPOWERCONTROL_ID enPowerControlID;
        ESUBSYSTEM_ID enConsoleNum;
};
//-----------------------------------------------------
// ENUM TYPE
//-----------------------------------------------------

enum ESSSHUTDOWN {
        EBOOTINGCOMPLETE,
        EQUIPMENTBOOTING,
        EQUIPMENTAUTOSHUTDOWN
};
```

Fig. 10. IDL Configruation File

전투체계에서 사용되는 DSS 메시지는 struct 타입으로 정의하였다. DSS 메시지 외에 체계에서 공통적으로 사용되는 값을 enum 으로 정의하여 IDL 파일로 관리할 수 있다.

IV. Software Evaluation

본 논문에서 제안한 정보저장 소프트웨어의 개선 정도를 평가하기 위하여 메시지 변경에 대한 클래스 및 소스코드 수정 요소와 신뢰성 시험 수행 시간 비교로 확인하였다. 평가 실험에 대한 환경은 아래 Fig 11, Table 3과 같다.

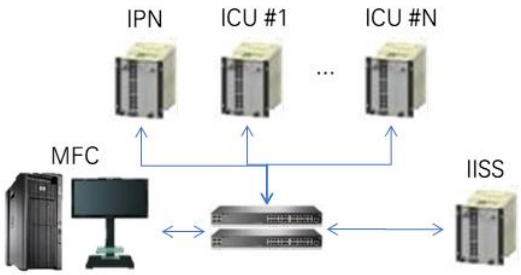

Fig. 11. Test Environment

Table 3. Test Environment

Item	MFC	IPN, IISS	ICU
CPU	Intel Core i7-6700 @3.40GHz	Intel Core i7-4700 EQ @2.40GHz	Intel Core i7 E610 @2.53GHz
Memory	8 GB	8GB	4GB
OS	Window 7	RTST Linux	RTST Linux

1. Code Changeability

본 논문에서 제안한 정보저장 소프트웨어의 코드 가변성 확인은 메시지 변경에 대한 의존성을 실험하는 것으로 신규 장비 추가로 인한 메시지가 추가되는 경우 영향을 받는 클래스의 수, 소스 파일 수, 소스코드 라인 수를 비교하는 것으로 확인 하였다. Table 4, 5는 그 결과를 정리한 표이다. 메시지 추가는 연동장비와 송수신 메시지 최소 2개, 장비 제어를 위한 전시 소프트웨어와 송수신 메시지 최소 2개로 총 4개의 메시지로 가정하였다.

Table 4. Compare number of classes and source file

-	Exist Software	Proposed Software
The Number of classes	4 classes	0 classes
The Number of source file (Include configuration file)	6 files	1 files

Table 5. Compare number of source code lines

-	Exist Software	Proposed Software
The Number of source code lines	about 800 lines added	about 60 lines added

신규 장비 도입으로 인한 메시지 추가 시 수정이 필요한 클래스의 수는 제안한 정보저장 소프트웨어에서 총 0개, 파일의 개수는 1로 나타났다. 이는 새로운 메시지가 추가되어도 IDL 설정파일에만 영향을 주고 메시지를 수신하고 시간정보를 추가하여 파일로 저장하는 로직에 전혀 영향을 주지 않기 때문에 수정이 필요한 클래스는 0으로 나타났다. 소스코드 라인 수는 설정파일 포함 약 90%의 감소율을 보였다. 이를 통해 소프트웨어의 재사용성이 향상된 것을 확인 할 수 있다.

2. Time to Perform Reliability Test

Table 6는 앞선 실험과 똑같은 가정 하에 신뢰성 시험 수행 시간을 비교한 결과이다. 신뢰성 시험은 MISRA, Code Sniper 기반의 정적시험 및 Quality Scroll Cover 기반의 동적시험을 대상으로 하였으며 총 3회 수행한 평균을 계산하였다.

Table 6. Compare time to perform reliability test

Reliability Test	Exist Software	Proposed Software
MISRA Code Sniper	2 days	-
Quality Scroll Cover	3 days	-

신뢰성 시험 수행 시간 비교에서도 차이를 보였다. 기존 정보저장 소프트웨어는 약 10만 라인의 규모가 큰 소프트웨어로 신뢰성 시험 수행을 위한 빌드에서 많은 시간과 동적시험 시 모든 메시지 수신 처리에 대한 테스트 케이스 수행에 많은 시간이 할애되었다. 제안된 정보저장 소프트웨어에서는 소스코드 수정이 없으므로 추가적인 신뢰성 시험이 불필요하며 메시지가 변경되어도 추가적인 개발 시간이 요구되지 않는다.

V. Conclusions

정보저장 소프트웨어는 전투관리체계의 메시지의 변경에 따라 소스코드 수정 및 신뢰성 시험 수행이 필요하여 개발 cost가 높은 소프트웨어이다. 변경되는 메시지의 개수에 비례하여 증가되는 개발 요소 및 개발 시간은 개발자에게 일의 만족도 저하와 개발의 효율성을 떨어뜨릴 수 있다.

본 논문에서는 변경 용이성 향상을 위한 전투관리체계 정보저장 소프트웨어 아키텍처의 구조 개선 연구를 진행하였다. Packet Sniffing 방식을 이용한 정보저장 소프트웨어를 제안함으로써 메시지의 변경의 의존도를 줄였으며 디자인패턴 기법 중 팩토리 메소드 패턴을 적용하여 소프트웨어를 설계하여 메시지 변경 시 영향을 받는 클래스를 최소화 하였다. 제안한 정보저장 소프트웨어의 클래스, 함수 및 코드 라인 수, 신뢰성 수행 시간 비교를 통해 변경 용이성과 재사용성의 향상으로 개발 요소가 크게 저감된 것을 확인 할 수 있었다. 본 논문에서 제안한 방법은 개발 단계에서 발생할 수 있는 신규 장비 도입, 기능 변경 발생 시에도 수정요소에 대한 영향을 줄여 개발 비용 감소 및 인적 오류를 최소화 시킬 수 있어 높은 소프트웨어 품질과 안정적인 체계 운용을 기대할 수 있다.

REFERENCES

[1] Ko, Soon Joo, Park, Do Hyun. "An Examination on Overseas Technology Trend and Domestic Development Pattern of the Naval Combat Management System." Journal of the Korean Association of Defense Industry Studies, Vol. 16, No. 2, pp. 237-258, Dec. 2009.

[2] MISRA, http://www.misra.org.uk/

[3] Code Sniper, http://www.suresofttech.com/html/tool/code_sniper/

[4] Quality Scroll Cover, http://www.suresofttech.com/html/tool/ quality_cover/

[5] 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.

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

[7] Sang-Min Kwon, Seung-Mo Jung, “Virtualization based high efficiency naval combat management system design and performance analysis”, Journal of The Korea Society of Computer and Information, Vol. 23, No. 11, pp. 9-15, November 2018.

[8] DDS, https://www.omg.org/omg-dds-portal/

[9] Data Distribution Service for Real-Time Systems Specification, OMG, March 2004.

[10] S. Ansari, S. G. Rajeev and H. S. Chandrashekar, "Packet sniffing:

a brief introduction," in IEEE Potentials, vol. 21, no. 5, pp. 17-19, Dec. 2002-Jan. 2003.

[11] Sungmo Jung, Jae-gu Song, Seoksoo Kim, Gil-Cheol Park. "A Study on Efficient Monitoring System using Packet Sniffing", Korean Institute of Information Technology, pp. 587-590, Jun. 2009.

[12] Briscoe, Neil. "Understanding the OSI 7-layer model.", PC Network Advisor, 2000.

Authors

Ji-Yoon Park received the B.S. and M.S. degrees in Computer Science and Engineering from Chungnam National University, Korea, in 2013, 2015. She is currently working in Hanwha Systems Co. from 2015. She is interested in Naval Combat System, Combat Management/Support Software, Design Pattern and so on.

Moon-Seok Yang received the B.S degrees in Electronics from In-Ha University, Korea, in 1998. He has been a software development manager in Hanwha Systems since 1998. He is interested in Combat System, Combat Management/Support Software, Human Computer Interface, The Fourth Industrial Revolution and so on.

Dong-Hyeong Lee received B.S degree in Computer Engineering from Kyungpook National University, Korea, in 2010. He is currently working in Hanwha Systems Co. from 2009. He is interested in Naval Combat System, Combat Management/Support Software, Design Pattern and so on.