JKSCI

한국컴퓨터정보학회논문지
Journal of The Korea Society of Computer and Information
Vol. 25 No. 11, pp. 147-155, November 2020
https://doi.org/10.9708/jksci.2020.25.11.147

A Study on the Standardization of System Support Software in the Combat Management System

Young-Dong Heo*

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

[Abstract]

System support software is one of the software that makes up ship combat management system and has the characteristics of being mounted in the combat management systems of all ships but with little functional change. However, despite these characteristics, software modifications due to equipment, etc. are inevitable in the application of new ships. Modification of software causes software reliability testing which is a key factor in increasing development costs. In this paper, the structure of the existing system support software was analyzed to identify and supplement the code change factors, and the system support standardization architecture was designed. The feature model elicited common and variable elements of system support software, and applied white-box reuse to improve software design. In addition, the results of comparing existing system support software with the new architecture in terms of development elements and time to perform reliability test were presented to verify the effectiveness of the new one.

▸ **Key words**: Combat Management System, System Support Software, Standardized Software, White Box Reuse, Architecture

[요 약]

체계지원 소프트웨어는 함정 전투관리체계를 구성하는 소프트웨어 중의 하나로 모든 함정의 전투관리체계에 탑재되지만 기능 변경이 거의 일어나지 않는 특성을 가진다. 하지만, 이러한 특성에도 실제로는 신규함정 적용 시 장비 변경 등으로 인한 소프트웨어 수정이 불가피하게 이루어진다. 이러한 소프트웨어의 수정은 소프트웨어 신뢰성 시험 등의 작업을 초래하며, 개발 비용 증가의 핵심요인으로 작용한다. 본 논문에서는 기존 체계지원 소프트웨어의 구조를 분석하여 이를 보완할 수 있는 체계지원 표준화 아키텍처를 설계하였다. 휘처 모델(Feature Model)을 통해 체계지원 소프트웨어의 공통요소와 가변요소를 도출하고, 화이트 박스 재사용(White-Box Reuse)을 적용하여 소프트웨어 설계를 개선하였다. 또한, 기존 체계지원 소프트웨어와 개발 요소 및 신뢰성 시험 수행 시간을 비교하여 새로운 아키텍처의 효과성을 검증한 결과를 제시하였다.

▸ **주제어**: 전투체계, 체계지원 소프트웨어, 소프트웨어 표준화, 화이트 박스 재사용, 아키텍처

• First Author: Young-Dong Heo, Corresponding Author: Young-Dong Heo
*Young-Dong Heo (yd1208.heo@hanwha.com), Naval R&D Center, Hanwha Systems
• Received: 2020. 09. 28, Revised: 2020. 11. 10, Accepted: 2020. 11. 10.

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

I. Introduction

함정 전투체계 내에서 전투관리체계(CMS: Combat Management System)는 인간의 두뇌에 해당하는 역할을 수행함으로써, 통신체계, 센서체계, 무장체계 등을 통제하고 관리하는 핵심적인 체계이다. CMS 는 센서 및 무장으로부터 메시지를 수신하여 명령 메시지를 송신하는 역할을 수행하는 연동단, 다기능콘솔(MFC: Multi Function Console), 레이더 및 TV비디오 처리장치 그리고 체계지원(System Support) 과 같은 기능을 수행하는 정보처리장치(IPN: Information Processing Node) 등으로 구성되어 있다[1].

이러한 함정의 전투관리체계에 들어가는 소프트웨어 중 체계지원 소프트웨어는 전투체계 운용 시 작전을 원활하게 수행 할 수 있도록 부가적인 서비스를 제공해주는 소프트웨어의 모음이다. 체계지원 소프트웨어는 체계 공통기능으로 함정마다 탑재되는 센서나 무장의 종류가 달라져 소프트웨어 변경이 심한 연동단 소프트웨어와는 달리 센서나 무장의 영향성을 받지 않기 때문에 각 함정에 들어가는 소프트웨어의 기능 변경이 거의 일어나지 않는다.

하지만 이런 체계지원 소프트웨어도 하드웨어 장비의 수량이나 운영체제의 변경 등으로 소스코드 변경이 불가피하게 발생하게 되며, 이러한 의존성은 개발 비용의 증가를 야기하게 된다. 특히, 소스코드의 변경은 소프트웨어 설계, 구현, 기능시험과 소프트웨어 정적시험(MISRA [2], Code Sniper [3]) 및 동적시험(Quality Scroll Cover [4])이라는 소프트웨어 신뢰성 시험을 동반하므로 많은 노동력과 시간을 요한다. 또한, 소프트웨어 수정 및 신뢰성 시험의 수행은 소스코드 규모 및 난이도, 개발자의 숙련도, 보조도구 사용여부, 신뢰성시험 검증 툴의 라이선스 수 등 업무 수행에 영향을 미치는 요소들을 가지고 있으며 사람에 의한 작업으로 인적 오류의 잠재적인 요소도 존재한다[5].

본 논문에서는 체계지원 소프트웨어를 주변 환경의 변화에도 불필요한 소스코드 수정이 없는 표준화된 소프트웨어로 고도화하기 위한 아키텍처를 제안한다. 체계지원 소프트웨어의 구조를 분석하여 코드 변경 요인을 식별하고, 이를 보완할 수 있는 체계지원 표준화 아키텍처를 설계하였으며 또한, 기존 체계지원 소프트웨어와 개발 요소를 비교하여 소프트웨어의 효과성을 검증한 결과를 제시하였다.

본 논문의 구성은 다음과 같다. 2장에서는 관련연구로 현재 함정 전투관리체계 소프트웨어의 구조와 개선사항에 대해 설명하고 연구의 목적을 언급하며 3장에서는 새로운 체계지원 소프트웨어 아키텍처 설계에 대해 설명한다. 4장에서는 본 논문에서 새롭게 설계한 체계지원 소프트웨어와 기존 소프트웨어의 개발 요소를 비교하여 개선 효과를 제시하며 마지막 5장에서 연구결과에 대한 결론으로 마무리한다.

II. Preliminaries

1. Related works

1.1 Naval Combat System

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

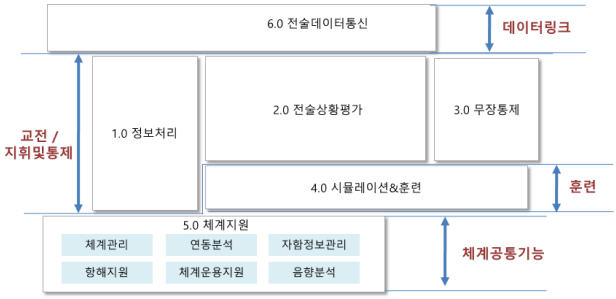

Fig. 1. Function of Combat Management System

Fig. 1 에 체계공통기능으로 분류되어 있는 것으로 알 수 있듯 체계지원 소프트웨어는 모든 함정에 공통적으로 들어가며, 운용자가 교전/지휘 및 통제, 훈련 등을 할 수 있도록 지원하는 서비스이다.

1.2 DSS Communication with Combat System

함정 전투관리체계 소프트웨어는 Fig. 2 와 같은 메시지 기반의 DDS(Data Distribution Service) 통신을 사용한다.

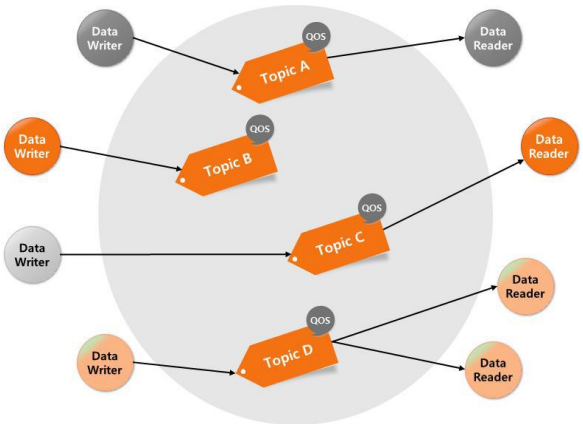

Fig. 2. DDS Middleware

DSS 통신은 DDS 를 기반으로 만든 통신 서비스이다. 체계지원 소프트웨어는 DSS 메시지를 이용하여 기능을 동작하는데 필요한 데이터를 주고 받으면서 (Publication/Subscription) 동작하며 이를 사용하기 위한 관련된 클래스들을 기본적으로 포함한다. DSS 통신 개념상 메시지 변경의 발생 시 DSS 관련 클래스는 반드시 변경되어야한다. 또한 DSS 통신을 사용하기 위해서는 별도의 NodeAgent 라는 응용프로그램이 필요하며 메시지 변경 시 NodeAgent 도 재빌드 및 설치작업이 필요하다[6].

1.3 White-Box Reuse

본론에 앞서, 본 논문에서 소프트웨어 변경을 최소화 하기 위해 사용한 화이트 박스 재사용(White-Box Reuse) 관련 연구를 살펴본다[7]. 소프트웨어의 복잡도가 높으면 이해성이 낮아지고 수정하는데 그만큼 어려움이 따르며, 클래스에 선언된 메소드가 많을수록 재사용을 어렵게 해 소스코드 수정을 요한다. 또한, 재사용율이 낮아지게 되면 소스코드 수정 요소를 증가 시키므로, 소프트웨어의 신뢰성 시험에 악영향을 주게 되어 투입되는 시간, 공수(비용) 역시 증가하게 된다[8]. 화이트 박스 재사용은 소프트웨어의 복잡도를 낮추며 이는 전투관리체계 소프트웨어에 요구되어 지는 신뢰성 시험 효율을 향상시킬 수 있다는 의미가 된다[9].

1.4 Feature Model

휘처(Feature)란 사용자나 개발자가 식별할 수 있는 소프트웨어 시스템의 구별적인 특징을 의미한다. 휘처 모델은 이러한 휘처에서 나온 것으로 사용자나 개발자의 의사소통 매체로 사용되기도 하지만 공통요소과 가변요소를 식별하는데 많이 사용하는 기법이다[10]. 본 논문에서는 휘처 모델을 사용하여 전투관리체계 체계지원 소프트웨어에서 이전 사업들을 비교하였으며 공통요소과 가변요소를 좀 더 효과적으로 도출하였다.

2. Purpose of Study

2.1 System Support Software

체계지원 소프트웨어는 전투체계 운용 시 작전을 원활하게 수행 가능 하도록 부가적인 서비스를 제공해주는 소프트웨어로 구성되어 있으며 그 종류는 Table 1 과 같다.

Table 1. System Support Service

Category	name
Command Control Support	Console Marker Transmission
	Messenger
Farthest On Circle Management	Farthest On Circle Management
Tactical Area Management	Tactical Area Management
Area Warning Management	Area Warning Management
Platform Library Management	Platform Library Management
Record/Replay	Record
	Replay
Video Distribution Management	Video Distribution Management

체계지원 소프트웨어 중 하드웨어 형상 변경으로 인해 코드 변경이 필요한 소프트웨어는 전체 9개 중 3개로 식별된다. 식별된 소프트웨어는 Console Marker Transmission 과 Messenger, Video Distribution Management 로 현재 설계되어 있는 소프트웨어 구조로는 함정에 들어가는 다기능콘솔의 수량이 상이해진다면 소스코드 수정이 불가피하다.

2.2 Analysis with Featrue Model

휘처 모델을 통해 기존 체계지원 소프트웨어의 공통요소와 가변요소를 식별하였다. Fig. 3 은 기존 전투관리체계의 체계지원 Command Control Support 소프트웨어를 휘처 단위로 분석하고 휘처 모델로 표현한 그림이다.

2.3 Process of Modifying Software

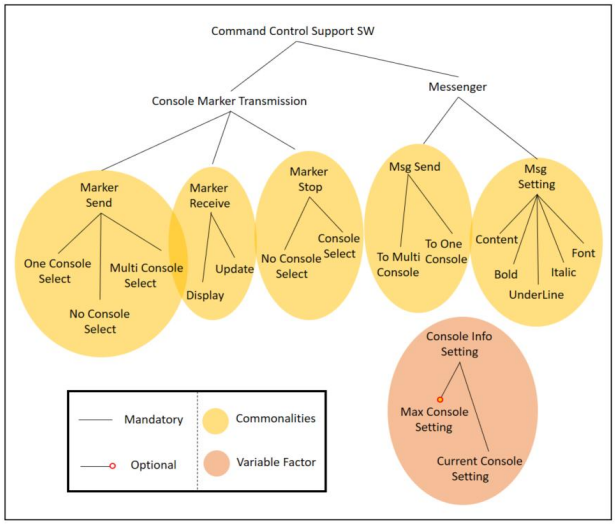

Fig. 3. Feature Model of System Support Software

Fig. 3 의 체계지원 소프트웨어 휘처 모델에서 공통적으로 가지고 있는 부분은 Mandatory 부분이며, Optional 은 가변적인 요소를 가지고 있는 부분이다. 아래 Table 2 는 체계지원 소프트웨어 휘처 모델을 참고하여 공통요소와 가변요소를 정리한 표이다.

Table 2. Commonalities and Variable Factor of System Support Software

Classification	Identification
Commonalities	Marker Send Part
	Marker Receive Part
	Marker Stop Part
	Msg Send Part
	Msg Setting Part
Variable Factor	Console Information Setting Part

Fig. 4. Process of Modifying System Support Software

하드웨어 변경 시 체계지원 소프트웨어를 수정하는 프로세스는 Fig. 4 와 같이 크게 5단계로 나눌 수 있으며 각 단계의 설명은 아래와 같다.

- DSS 메시지 수정(Modify DSS Message)
  - 변경된 하드웨어 식별 및 관련 메시지 필드 수정
- DSS 통신 클래스 생성(Extract DSS Comm Class)
  - MOMAT(Message Oriented Management and Analysis Tool)에서 DSS 통신 관련 클래스(CDataManager, CDSSAdaptor) 생성
- 소스 코드 수정(Modify Code)
  - 메인 클래스에 변경된 하드웨어 관련 코드 수정
- 신뢰성 시험 수행(Reliability Test)
  - 변경된 코드에 대해 신뢰성 시험 수행
- 빌드 및 실행(Rebuild and Run)
  - 변경 코드 빌드 및 실행

2.4 Expectation Effectiveness

본 논문에서는 하드웨어 변경에 대한 체계지원 소프트웨어의 변경을 최소화하기 위한 연구를 수행하였고, 기대효과는 다음과 같다.

- DSS 통신 메시지 변경 불필요
- 체계지원 소프트웨어 소스코드 변경 불필요
- 소프트웨어 신뢰성 시험 불필요

DSS 통신 메시지 변경을 하지 않기 위해 함정에 따른 하드웨어 가변요소를 식별하여 메시지를 재설계 하였고, 불필요한 소스코드 변경이 필요 없도록 설정 파일을 활용하여 가변데이터와 프로그램을 분리하였다. 또한, 화이트박스 재사용(White-box reuse) 기법을 새로운 체계지원 소프트웨어 설계에 적용하였다. 소스코드의 완전 재사용은 이미 검증된 소프트웨어를 재사용하는 것이므로 코드 변경이 일어나지 않기 때문에 소프트웨어 신뢰성 시험 수행을 수행하지 않아도 된다.

III. The Proposed Scheme

1. Message Design

1.1 New DSS Message Design

현재 운용되고 있거나 개발 중인 함정들의 다기능콘솔 형상은 Table 3 과 같다.

Table 3. MFC Configuration

Combat System	MFC Quantity
Case #1	8
Case #2	8
Case #3	9
Case #4	2
Case #5	9
Case #6	9
Case #7	15
Case #8	9
Case #9	10

Table 3 에서 알 수 있듯 각 함형 별로 다기능 콘솔의 형상이 상이하며 그로 인해 신규 사업의 체계지원 소프트웨어 개발 시 관련 DSS 메시지 변경이 불가피하게 이루어져 왔다. 이러한 DSS 메시지의 변경은 DSS 관련 클래스들에도 영향을 주기 때문에 메시지 변경 후 DSS 통신 클래스도 변경이 필요하다.

체계지원 소프트웨어의 DSS 메시지는 Table 4 와 같이 메시지를 수신하는 다기능콘솔의 여부를 넣어주는 Recevie MFC Field 가 bool Type의 다기능콘솔 수량 크기 배열로 설계 되어 있었다. 따라서 다기능콘솔의 형상이 달라지면 항상 배열의 크기 수정이 필요했다.

Table 4. Current DSS Message Configuration of System Support

Field Name	Type	Description
stMessageHead	Message Header	Message Header
byCommand	bool	Message Command Type
usConsoleNo	unsigned short	Send MFC Num
byReceiveMFC [MFC Quantity]	bool	Receive MFC
...	...	...

다기능콘솔의 형상이 달라져도 메시지 변경 없이 같은 메시지를 사용하면 되도록 새로운 체계지원 소프트웨어 DSS 메시지를 설계하였다. bool Type의 배열로 되어 있던 Recevie MFC 필드를 Table 5 와 같이 직접적으로 수신 다기능 콘솔 번호를 넣을 수 있도록 unsigned short Type 으로 변경하여, DSS 메시지 수정과 DSS 관련 클래스 생성이 불필요 하도록 개선하였다.

Table 5. New DSS Message Configuration of System Support

Field Name	Type	Description
stMessageHead	Message Header	Message Header
byCommand	bool	Message Command Type
usConsoleNo	unsigned short	Send MFC Num
usReceiveMFC	unsigned short	Receive MFC Num
...	...	...

2. Architecture Design

2.1 Separation of Data and Program using File

체계지원 소프트웨어 소스코드에서 다기능콘솔의 형상 변경으로 인해 영향을 받는 부분을 식별하였다. 아래는 체계지원 소프트웨어 소스코드에서 다기능콘솔 형상과 관련된 코드 중 수정이 필요한 부분이다.

SSICMMgrDlg.h

```
#include "SSICMSkinDlg.h"
...
#define MAX_MFC_NUM 9
...
class CSSICMMgrDlg : public CSkinBaseDlg
{
private:
...
CSSICMSkinDlg m_icmSkin[MAX_MFC_NUM+1];
int iModeInfo[MAX_MFC_NUM+1];
...
}
```

소스코드를 살펴보면 다기능 콘솔 수량(MAX_CONSOLE_NUM) 을 하드코딩으로 define 하고, 객체와 변수를 해당 define 을 사용하여 배열로 생성한다. 즉, 다기능콘솔 형상이 변경 되면 하드코딩 되어 있는 define 문을 바꿔주어야 하기 때문에 코드 수정이 반드시 필요하다. 소스코드 수정을 하지 않기 위해서 해당 define 을 삭제하고 다기능콘솔 수량(Data)을 파일로 따로 분리하였으며, 수정한 소스코드는 아래와 같다.

SSICMMgrDlg.h

```
#include "SSICMSkinDlg.h"
...
class CSSICMMgrDlg : public CSkinBaseDlg
{
private:
...
int m_iMFCMaxNum;
CSSICMSkinDlg * m_icmSkin;
int * iModeInfo;
...
public:
...
void SetMFCMaxNum(void);
...
}
```

헤더파일을 살펴보면 하드코딩 되어 있는 define 문을 제거 하였고 다기능콘솔 수량 저장을 위한 변수와 함수를 추가 선언해주었다. 또한, 기존 MAX_MFC_NUM+1 크기의 배열로 생성 했던 변수와 객체를 포인터로 선언해주었다.

SSICMMgrDlg.cpp

```
#include "CSSICMMgrDlg.h"
...
BOOL CSSICMMgrDlg::OnInitDialog(void)
{
  SetMFCMaxNum();
  ...
  m_icmSkin = new CSSICMSkinDlg[m_iMFCMaxNum+1];
  iModeInfo = new int[m_iMFCMaxNum+1];
 ...
}
```

소스파일을 살펴보면 메인 다이얼로그가 생성(OnInitDialog 함수) 될 때 먼저 SetMFCMaxNum 함수를 호출 하여 m_iMFCMaxNum 변수(다기능콘솔 수량) 의 값을 설정한다. 그 다음 포인터로 선언했던 객체와 변수를 m_iMFCMaxNum+1 크기의 배열로 동적할당 한다[11]. 이렇게 되면 다기능콘솔의 형상이 달라져도 dat 파일(설정파일) 만 수정하면 되므로 코드 수정을 할 필요가 없다.

2.2 Architecture Design with White-Box Reuse

기존 체계지원 소프트웨어(Console Marker Transmission, Messenger)의 Class Diagram과 Class 설명은 아래 Fig. 5, Fig. 6, Table 6, Table 7 과 같다.

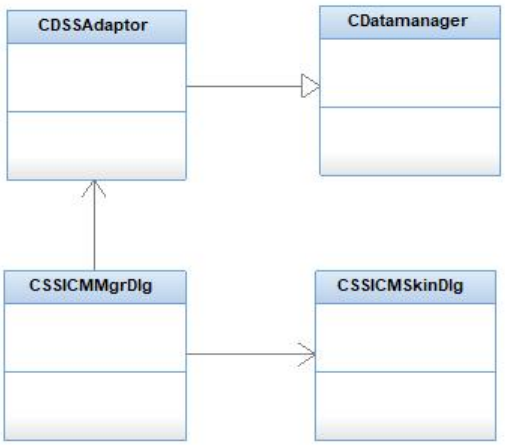

Fig. 5. Console Marker Transmission Software Class Diagram

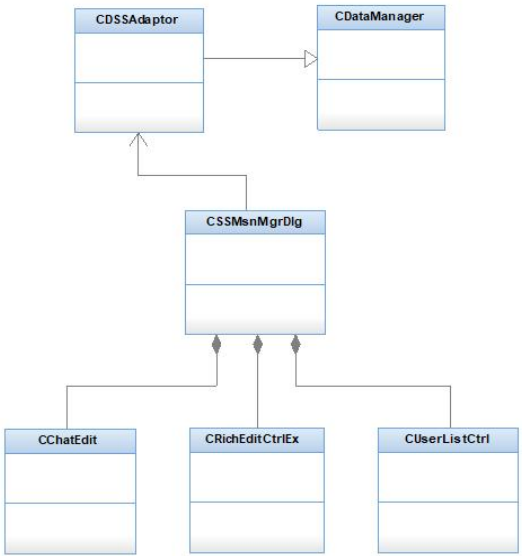

Fig. 6. Messenger Software Class Diagram

Table 6. Console Marker Transmission Software Class Description

Category	name
CDataManager	Class for DSS Communication
CDSSAdaptor	
CSSICMMgrDlg	Main Class
CSSICMSkinDlg	Class for Marker Painting

Table 7. Messenger Software Class Description

Category	name
CDataManager	Class for DSS Communication
CDSSAdaptor	
CSSMsnMgrDlg	Main Class
CChatEdit	Class for Typing
CRichEditCtrlEx	Class for Viewing
CUserListCtrl	Class for Participant Viewing

하드웨어 변경 시 수정되어야 하는 클래스를 식별하였다. 현재 체계지원 소프트웨어 구조상 다기능콘솔의 수량이 변경되면 DSS 메시지가 변경되어야하기 때문에 DSS 통신 관련 클래스들 모두 변경이 필요하다. 또한, 직접적으로 변수를 사용하는 메인 클래스들의 수정도 불가피하다. Table 6 과 Table 7 을 보면 알 수 있듯이 Console Marker Transmission Software 는 DSS 통신 클래스 CDataManager 와 CDSSAdaptor 와 메인 클래스, 마커 전시 클래스 총 4개의 소스파일로 구성되어 있다. 이 중 3개의 소스파일 즉, 전체의 75% 수정이 필요하며, Messenger Software 는 DSS 통신 클래스 CDataManager 와 CDSSAdaptor 와 메인 클래스, 채팅창 클래스, 채팅 전시 클래스, 대화참여자 리스트 클래스 총 6개의 소스파일로 구성되어 있고, 이 중 3개의 소스파일 즉, 50%를 수정해야 하는 것으로 분석되었다.

본 연구에서는 체계지원 소프트웨어의 수정을 최소화할 뿐 아니라 아키텍처에 화이트박스 재사용(White-Box Reuse)을 적용하여 추후 발생할 수 있는 기능 수정에 대한 재사용성을 높이는 전략을 채택하였다. 화이트 박스 재사용을 적용하여 메인 클래스와 파일을 읽어와 데이터를 저장하는 부분을 분리한다[12]. 분리한 클래스는 체계지원 소프트웨어 뿐 아니라 전투관리체계 전체에 다기능콘솔 수량을 얻는 소프트웨어에 사용이 가능하다. 또한, 이 클래스는 변경이 필요 없으므로 추 후 기능변경으로 인해 발생할 수 있는 신뢰성 시험 시 영향을 받지 않는다. 아래 Fig. 7 , Fig. 8 은 화이트박스 재사용을 적용하여 새롭게 설계한 체계지원 소프트웨어의 Class Diagram이다.

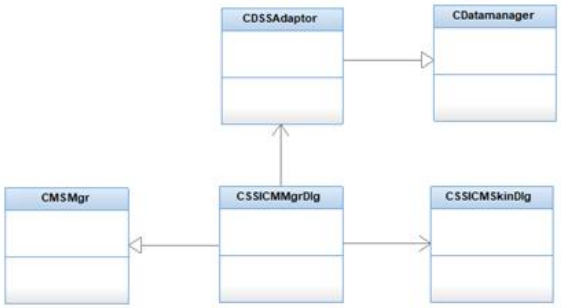

Fig. 7. New Console Marker Transmission Software Class Diagram

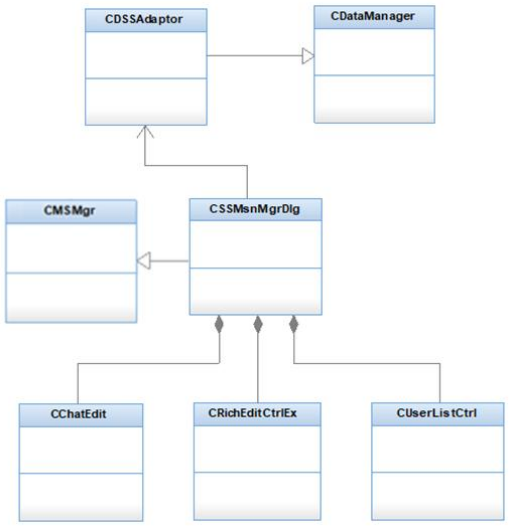

Fig. 8. New Messenger Software Class Diagram

CMSMgr 은 다기능콘솔 수량을 파일로부터 읽어 변수에 저장하는 클래스이다. 이를 상속 받아 메인 클래스를 구현한다. 새로운 체계지원 소프트웨어 구조는 메인 클래스와 해당 기능 클래스를 따로 분리하여 소프트웨어의 복잡도를 낮추고 소스코드 수정을 최소화 할 수 있으며 만약 전투관리체계의 다른 소프트웨어에서 다기능콘솔의 수량이 필요한 경우가 생긴다면 해당 클래스를 상속받아 사용이 가능하다.

IV. Verification and Test

본 논문에서 새롭게 설계한 체계지원 소프트웨어의 효과성을 평가하기 위하여 신규 사업의 다기능콘솔 형상이 달라졌을 시 기존 체계지원 소프트웨어와 새로운 체계지원 소프트웨어의 DSS 메시지 및 소스코드 수정 요소와 신뢰성 시험 수행 시간을 비교하였다. 비교를 위한 실험환경은 아래 Fig. 9, Table 8 과 같다.

Fig. 9. Environment for Verification

Table 8. Environment for Verification

Item	MFC	IPN
CPU	Intel Core i7-6700 @3.40GHz	Intel Core i7-4700 EQ @2.40GHz
Memory	8 GB	8 GB
OS	Window 7	RTST Linux

1. Changeability Analysis

1.1 Changeability of DSS Message and Code

새롭게 설계한 체계지원 소프트웨어와 기존 체계지원 소프트웨어의 DSS 메시지와 코드변경성 확인은 다기능콘솔 형상 변경에 대한 의존성을 실험하는 것으로 다기능콘솔 형상이 변경되었을 시 수정이 필요한 DSS 메시지 개수, 소스파일 수, 소스코드 라인 수를 비교하는 것으로 확인 하였다. 아래 Table 9, Table 10 은 변경이 필요한 DSS 메시지 개수와 소스파일 개수, 소스코드 라인 수를 정리한 표이다.

Table 9. Analysis of Exist Software

-	Total Cnt	Changing Cnt	Reusability
The Number of DSS Message	2 Messages	2 Messages	0%
The Number of Source File	4 files	3 files	25%
The Number of source code lines	3,256 lines	262 lines	91%

Table 10. Analysis of New Software

-	Total Cnt	Changing Cnt	Reusability
The Number of DSS Message	2 Messages	0 Messages	100%
The Number of Source File	5 files	0 files	100%
The Number of source code lines	3,360 lines	0 lines	100%

Table 9, Table 10 을 참고하여 분석을 한 결과는 다음과 같다. 다기능콘솔 형상이 변경되었을 시 기존 체계지원 소프트웨어은 DSS 메시지와 소스파일, 소스코드를 모두 변경해주어야 하며 특히, DSS 메시지는 재사용성이 0%로

측정되었다. 하지만, 새로운 체계지원 소프트웨어는 변경해야 할 DSS 메시지, 소스파일, 소스코드 세 가지 항목이 0개로 100% 재사용이 가능함이 확인되었다. 즉, 새로운 아키텍처를 적용한 결과 재사용률이 모두 100%로, 기존 개선전의 수치와 비교하여 각각 100%, 75%, 9% 가 개선되었음을 확인하였다.

2. Reliability Test Analysis

2.1 Time to Perform Reliability Test

아래 Table 11 은 기존 체계지원 소프트웨어와 새로운 체계지원 소프트웨어의 신뢰성 시험에 수행시간에 대한 실험결과이다. 신뢰성 시험은 정적시험(MISRA, Code Sniper) 및 동적시험(Quality Scroll Cover)을 대상으로 실제 수행시간을 측정하였다.

Table 11. Compare Time to Perform Reliability Test

-	Exist Software	New Software
Static Test	2 days	0 day
Dynamic Test	2 days	0 day

신뢰성 시험 수행시간을 비교 해보면 기존 체계지원 소프트웨어는 정적시험과 동적시험 각각 2일 씩 걸리는 반면 새로운 소프트웨어는 다기능콘솔 수량 정의파일(dat 파일) 변경 외에는 소스코드 수정이 일어나지 않으므로 신뢰성 시험을 전혀 수행 할 필요가 없다. 따라서 다기능콘솔의 형상이 달라져도 신뢰성 시험을 하지 않아도 된다.

2.2 Complexity Analysis

Table 12. Compare Maximum Complexity

-	Exist Software	New Software
Maximum Cyclomatic Complexity	12	9

또한, 신뢰성 시험 측정 결과 Table 12 와 같이 프로그램의 최대 복잡도 수치가 12 에서 9 로 낮아졌음을 알 수 있으며 이는 화이트 박스 재사용과 설정 파일을 활용한 새로운 체계지원 아키텍처의 개선방법이 프로그램 복잡도 향상에 효과가 있음을 확인할 수 있다.

V. Conclusions

함정 전투관리체계에서 체계지원 소프트웨어는 전투체계 운용 시 작전을 원활하게 수행 할 수 있도록 부가적인 서비스를 제공해주는 소프트웨어로 하드웨어 형상이 변경되면 소프트웨어 수정이 불가피하게 이루어져 왔다. 이러한 소프트웨어의 변경은 소프트웨어 설계, 구현, 기능시험 뿐 아니라 소프트웨어 신뢰성 시험과 같은 많은 노동력과 시간을 요구하는 작업을 초래하며, 개발 비용 증가의 핵심 요인으로 작용한다.

본 논문에서는 기존 체계지원 소프트웨어를 환경의 영향을 최소화하여 불필요한 소스코드 변경이 없는 표준화된 체계지원 소프트웨어로 고도화하는 연구를 진행하였다. 새로운 DSS 메시지를 설계하여 DSS 메시지 수정 및 통신 관련 클래스 수정을 최소화 하였으며 또한, 화이트박스 재사용과 설정 파일을 활용하여 가변데이터와 프로그램을 분리함으로써 코드 변경을 최소화한 새로운 아키텍처를 제안하였다. 그리고 비교실험을 통해 소스코드 변경 없이 100% 재사용이 가능함을 확인 할 수 있었다. 새로운 체계지원 소프트웨어는 신규 사업 투입 시 바로 적용이 가능할 것으로 보이며 불필요한 공수 최소화, 개발 비용 감소의 효과를 얻을 수 있을 것으로 판단된다.

REFERENCES

[1] Shin Hun Yong, Kim Joo Yong, "Research of OSD Standardization in Naval Combat System" The Korean Institute of Electrical Engineers, pp. 354-355, Oct 2012.

[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] Ji-Yoon Park, Moon-Seok Yang, Dong-Hyeong Lee, “A Study on IISS Software Architecture of Combat Management System for improving modifiability”, Journal of The Korea Society of Computer and Information, Vol. 25, No. 2, pp. 130-140, May 2020.

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

[7] White-Box Reuse, https://wikidocs.net/894

[8] Wonseok Jang, Donghan Jung, Yunchul Ha, “Analysis of Influential Relationship among Defense Software’s Reusability and Efficiency of Reliability Testing by 6 Sigma”, Journal of Defense Quality Society, Vol. 2, No. 1, pp. 63-73, June 2020.

[9] Hee Whan Yoon, Young Jip Kim, Yeon Seol Koo, “Metrics for Measuring of White - box and Black - box Reusability in Object

- Oriented Programs", Journal of The Korean Institute of Information Scientists and Engineers, Vol. 28, No. 2, pp. 104-112, February 2001.

[10] Seung-Mo Jung, Young-Ju Lee, "A Study on the Model Driven Development of the Efficient Combat System Software Using UML", Journal of the Korea Society of Computer and Information, Vol. 21, No. 10, pp. 115-123, Oct 2016.

[11] K. N. King, "C Programming", W. W. Norton & Company, 2004.

[12] Erich Gamma, "Design patterns : elements of reusable object-oriented software", ADDISON WESLEY, 2003.

Authors

Young-Dong Heo received the B.S degree in Computer Engineering from Kyungpook National University, Korea, in 2014. He is currently working in Hanwha Systems Co. from 2014. He is interested in Naval Combat System, Combat System Support Software, Software Reuse and so on.