JKSCI

한국컴퓨터정보학회논문지
Journal of The Korea Society of Computer and Information
Vol. 27 No. 12, pp. 197-207, December 2022
https://doi.org/10.9708/jksci.2022.27.12.197

A Study on the Improved Post-Analysis Development System in the Naval Combat System

Chang-Won Seo*

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

[Abstract]

The Naval Combat System operates in multiple software components for efficient functional processing. Data transmission and reception between components is performed in real time using DDS(Data Distribution Service). Reliable Post-Analysis requires the collection of all DDS messages. However, Software workload and development costs increase because common functions for all messages must be developed directly. In the paper, We propose an improved Post-Analysis based on the Feature Model and a new development system using it. Functions for the modified message were applied as variable domains of the Feature Model. When Build Center updates Post-Analysis, it automatically processes all software tasks associated with Post-Analysis. The proposed development system simplified the overall work procedure, and work time was significantly reduced compared to the existing development system.

▸ **Key words**: Naval Combat System, DDS, Post-Analysis, Feature Model, Design Pattern

[요 약]

함정 전투체계는 안정적인 기능 처리를 위해 다수의 소프트웨어 컴포넌트로 나뉘어 동작하며, 컴포넌트 간의 데이터 송수신은 메시지 지향 미들웨어(DDS, Data Distribution Service)를 이용하여 실시간으로 이루어진다. 신뢰성 높은 사후분석을 위해서는 컴포넌트 간 송수신 되는 메시지들을 모두 수집해야 하는데, 각 메시지에 따른 공통 기능들을 직접 구현해야 하는 만큼 소프트웨어 작업도 증가하게 되고 이는 개발 비용의 증가로 이어진다. 본 논문에서는 휘처 모델(Feature Model)에 기반한 사후분석 자동화 기법과 이를 적용한 사후분석 개발 프로세스를 제안한다. 메시지 관리 시스템에서 변경되는 메시지들을 휘처 모델의 가변 요소로 적용하였고, 빌드 센터에서 사후분석을 최신화하면 메시지와 관련된 모든 소프트웨어 작업을 자동으로 처리된다. 제안하는 개발 체계는 전체적인 작업의 프로세스가 단순화되었고, 소프트웨어 작업 시간이 감소함을 확인하였다.

▸ **주제어**: 함정 전투체계, DDS, 사후분석, 휘처모델, 디자인 패턴

• First Author: Chang-Won Seo, Corresponding Author: Chang-Won Seo
*Chang-Won Seo (cwseo541@hanwha.com), Naval R&D Center, Hanwha Systems
• Received: 2022. 10. 28, Revised: 2022. 11. 29, Accepted: 2022. 12. 15.

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

I. Introduction

함정 전투체계는 다수의 개발자가 개발에 투입되므로 하나의 산출물로써 운용되기에는 현실적으로 어려움이 있다. 이에 무장, 센서 등 장비의 특성에 따라 수십 개에서 수천 개의 소프트웨어 컴포넌트들로 나뉘어 구성된다[1]. 컴포넌트들은 서로 독립적으로 개체로써 운용되므로 각 컴포넌트 간 인터페이스를 정의하고 이를 관리하는 것은 단순 체계의 운용뿐만 아니라 개발 효율에도 직접적인 영향을 미친다[2]. 함정 전투체계에서는 메시지 지향 미들웨어 DDS(Data Distribution Service)를 통한 컴포넌트 간 실시간 통신을 사용한다[3]. 컴포넌트 간 필요한 동작을 메시지로 정의하여 송수신하고 이에 해당하는 동작을 처리함으로써 다수의 컴포넌트는 하나의 체계로 운용할 수 있다. 다만, 하나의 동작에 한 개 이상의 메시지가 필요하므로 컴포넌트의 수가 증가함에 따라 이에 비례하여 동작 처리를 위한 메시지의 수가 증가하게 된다. 함정 전투체계의 규모로 인해 메시지 정의 및 관리 도구가 취급하는 프로젝트와 메시지, 개발자의 규모가 동반 상승하였고, 결과적으로 작업량 및 개발 체계의 부하 증가에 따른 개발 생산성의 저하[4]와 개발 비용의 증가로 이어졌다.

운용되는 플랫폼에 대한 신뢰성 향상을 위하여 실제 장비의 알고리즘을 모델화하는 노력이 진행되고 있으며, 특히 운용 결과에 대한 사후분석이 널리 활용되고 있다[5]. 신뢰성 높은 사후분석을 위해서는 운용 당시 컴포넌트 간 송수신 된 모든 메시지 정보를 수집할 필요가 있다. 앞서 설명된 바와 같이 컴포넌트의 수만큼 처리되는 메시지의 수도 비례하여 증가하고 소스코드 레벨에서 메시지 단위로 수정되어야 한다. 현재 메시지들의 세부 기능을 제외한 송수신, 검색 및 초기화와 같은 공통 기능들은 기능명과 메시지에 따른 헤더 설정에서만 차이가 있을 뿐 사후분석에서 취급하는 함수 내부 기능들은 모두 동일한 구조를 가진다. 메시지 정의 및 관리 시스템에서는 특정 컴포넌트에 대하여 송수신되는 메시지에 대한 공통 기능을 구현한 함수들을 자동 생성하는 기능을 제공하고 있다. 현재의 구조로도 동작에는 문제가 없으나 동일한 동작을 하는 함수들을 메시지의 수만큼 별도로 구현하는 것은 일부 내용이 변경될 경우 모든 메시지 함수에 대하여 동일한 수정을 반복해야 하고, 늘어난 함수의 수와 비례하여 소스코드의 길이도 늘어나 소프트웨어 빌드 및 신뢰성시험 시간이 증가하므로 관리가 어렵고 비효율적이다. SW 신뢰성시험 시간을 단축하고 인적오류를 감소시키는 방안은 SW 품질확보 및 SW 생산성을 증대로 이어진다[6].

본 논문에서는 휘처 모델(Feature Model)에 기반한 사후분석 자동화 기법을 제안한다. 송수신 및 초기화와 같이 동일한 동작을 하는 함수들을 공통 요소로 정의하고 추가, 변경 및 삭제되는 메시지들을 휘처 모델의 가변 요소로 설정하였다. 공통 요소에 가변 요소들이 접근하여 사용하도록 구조를 단순화하여 관리 효율을 높였다. 제안하는 기법은 소프트웨어 처리에서 작업 시간이 단축되고 소스코드 구조도 단순화된 만큼 관리에서의 효율성 증가와 더불어 작업 시간이 단축됨을 확인하였다.

본 논문의 구성은 다음과 같다. 2장에서는 현재 전투체계의 구조와 제안하는 기법에 적용되는 개념을 설명한다. 3장에서는 현재의 사후분석과 전투체계의 동작을 설명하고, 4장에서는 기존 개발 체계를 개선한 사후분석 개발 체계 자동화 기법을 제안한다. 5장에서는 현재의 개발 체계와 제안하는 개발 체계의 성능을 비교 및 분석한다. 마지막으로 6장에서는 결론 및 기대하는 향후 연구 방향성을 기술한다.

II. Preliminaries

1. Related works

1.1 Naval Combat System

함정 전투체계는 크게 교전/지휘 및 통제, 데이터링크, 훈련 및 체계공통기능으로 이루어져 있으며, 각 기능들은 세부적으로 수십 개에서 수백 개의 컴포넌트로 나뉜다. 함정 전투체계의 기본적인 구조는 Fig.1과 같다[7].

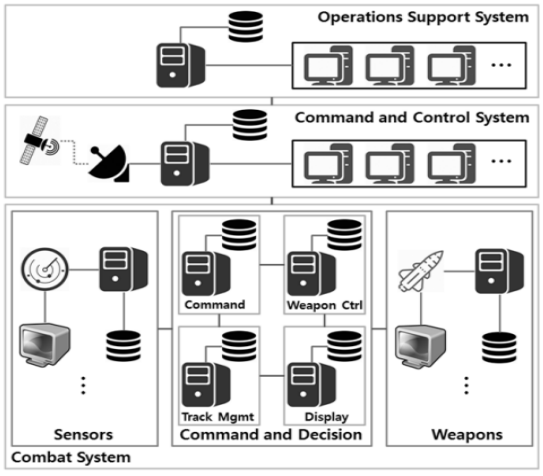

Fig. 1. Structure of legacy Naval Combat Systems

컴포넌트들은 특정한 동작을 위해 정의된 메시지를 송수신하고 이를 처리함으로써 하나의 체계로서 기능 수행이 가능해진다.

함정 전투체계의 사후분석은 작전 상황들을 기록 및 재생하면서 이에 대하여 분석하는 것이며, 사후분석을 통해 더 나은 작전을 수행하는 것이 가능하다. 이러한 이유에서 전투체계에서는 사후분석을 중요한 기능으로 취급하고 있으며, 효율적인 기록 및 재생 기법에 대한 연구도 진행되고 있다[8].

1.2 Real-time Communication with DSS

함정 전투체계는 DSS(Data Sharing Service)를 이용한 실시간 통신으로 정보를 교류하며, DSS는 국제 표준인 실시간 데이터 분배 미들웨어 DDS를 활용한다.

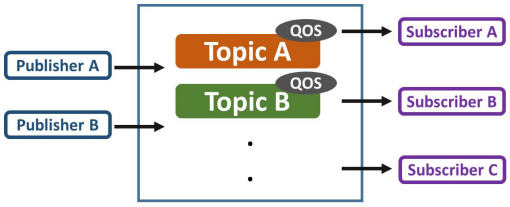

Fig. 2. Publisher/Subscriber of DDS Middleware

DDS는 고효율의 정보 분배를 위해 Topic의 발행자와 가입자 사이에 Publisher/Subscriber 패러다임을 적용한다[9-10]. DSS를 사용하기 위하여 메시지의 기본적인 정의 및 동작을 위한 소스코드가 구현되어 있으며, 메시지 변경 시 해당 소스코드들의 수정도 무조건 병행되어야 한다.

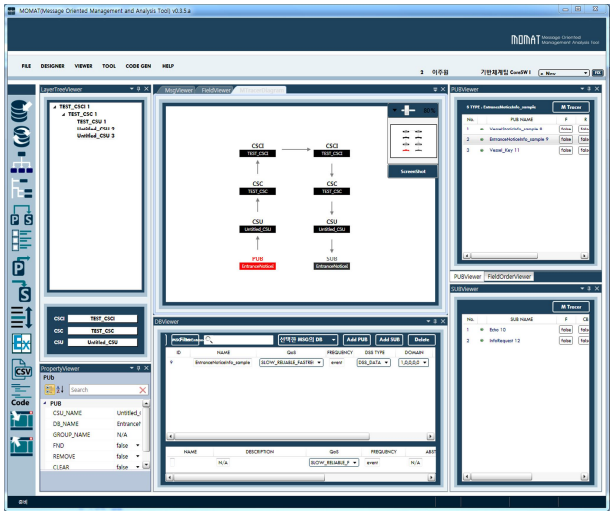

Fig. 3. GUI of MOMAT

함정 전투체계는 Fig.3과 같은 메시지 추가, 수정 및 삭제는 메시지 정의 및 관리 체계인 MOMAT(Message Oriented Management and Analysis Tool)을 사용하여 메시지를 관리한다[11]. MOMAT은 기존에 사용된 MDMS(Message Define & Management System)에서 다수 사용자의 접근 시 성능 저하가 발생하는 문제를 해소한 메시지 형상관리 도구로써, 컴포넌트 간 동작을 위한 메시지 정의 및 관리가 가능하다. 특히, 컴포넌트에 요구되는 Topic을 생성할 수 있을 뿐만 아니라 Publisher/Subscribe 적용, 메시지 검색 기능 활성화 등과 같은 설정을 MOMAT이라는 하나의 도구에서 모두 처리할 수 있으며, 설정된 기능에 대한 소스코드를 사용자의 요구에 따라 전체 또는 컴포넌트 단위로 자동으로 구현하여 출력해주는 기능도 지원한다.

1.3 Feature Model

소프트웨어 개발에 있어 기능의 단순 반복을 최소화하고 재사용(Reusability) 가능한 부분을 식별하여 이를 적용하는 것은 개발뿐만 아니라 관리 측면에서도 효율성을 높이는데 중요한 요소이다[12]. 이를 실제로 구현하기 위해서는 공통 요소와 가변 요소를 정확히 식별할 수 있어야 한다. 현재 가장 많이 사용되는 분석 방법으로 휘처 모델이 있다. 휘처란 사용자나 개발자가 식별할 수 있는 소프트웨어 시스템의 구별적인 특징을 의미하며, 휘처 모델은 이러한 휘처에서 나온 것으로 공통 요소와 가변 요소를 휘처의 관점에서 식별하는 기법이다[13]. FORM 방법론은 FODA[14]에서 제안한 휘처 모델로서 특정 도메인의 가변성을 식별 및 공통성을 가지는 컴포넌트를 재사용이 가능하도록 만들고 이를 조합하여 개별 기능을 빠르게 생성해내는 방법이다[15]. 가변적인 부분에서 공통 요소에 접근하는 기능을 별도로 구현해야 하는 번거로움이 있지만 소프트웨어 재사용 가능성 측면에서는 우수한 방법이다.

본 논문에서는 함정 전투체계의 DSS 메시지 인터페이스에 휘처 모델을 적용하였다. 각 메시지들이 동일하게 처리하는 기능들을 공통 요소로, 메시지에 따라 다르게 생성 및 요구되는 공통 기능들을 가변 요소로 식별하였다. 식별한 내용을 기존의 사후분석 구조에 적용함에 따라 하나의 DSS 메시지 인터페이스 클래스에서 메시지와 관련된 모든 기능을 수행하던 것에서 공통 요소와 가변 요소를 정의하는 클래스가 별도로 구현되도록 변경되었다.

1.4 Template Method Design Pattern

템플릿 메소드 디자인 패턴(Template Method Design Pattern)은 소프트웨어 클래스 구조에서 여러 작업들이 완전히 동일한 동작을 하지만, 일부 동작은 각각 다르게 구현해야 할 때 사용되는 디자인 패턴으로 중복되는 동작을 최소화할 수 있다[16]. 공통 기능을 상위 클래스에서 정의하고 서로 다르게 구현해야 하는 가변 기능을 하위 클래스에서 구현한다. 이러한 구조는 코드의 중복을 최소화할 수

있고, 동일한 기능을 수정이 필요하면 상위의 공통 기능만 변경해도 되므로 관리 측면에서 유용하며, 코드의 재사용 측면에서도 우수하다.

본 논문에서는 메시지의 송수신, 검색 및 초기화의 공통 기능을 상위 클래스에서 구현하고 메시지의 형식이나 인자 값에 따라 변경되는 내용을 하위 클래스로 구분하여 템플릿 메소드 디자인 패턴을 적용 및 구현하였다. 디자인 패턴의 적용으로 기존의 절차와 비교하여 중복 코드를 줄이고 코드의 재사용율을 크게 높였다.

III. The Exist Scheme

1. Post-Analysis

함정 전투체계의 사후분석은 DSS 메시지 인터페이스, 메시지 기록 및 재생, 기록 파일 관리 그리고 모듈 상태 보고의 4가지의 기능 그룹으로 구성된다. 메시지 인터페이스는 DSS 메시지의 정의 및 동작과 관련된 기능들이 구현되어 있고, 기록 및 재생은 DSS 메시지의 송수신과 처리에 대한 정보를 시간의 경과에 따라 기록하고 이를 재생한다. 본 논문에서는 실제 DSS 메시지의 처리와 관련된 DSS 메시지 인터페이스와 기록 및 재생의 2가지 기능을 다룬다. 사후분석 소프트웨어에 대한 클래스 다이어그램과 클래스 설명은 아래 Fig. 4 및 Table 1과 같다.

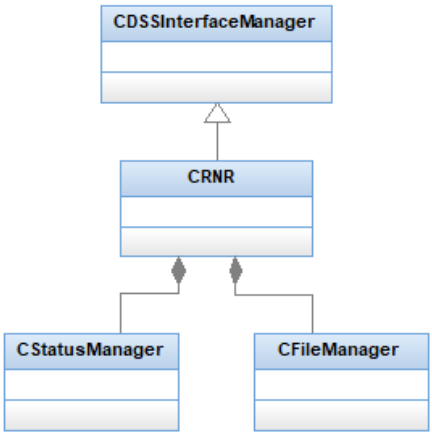

Fig. 4. Existing Post-Analysis software class diagram

Table 1. Existing Post-Analysis software class description

Class	Description
CRNR	Class for Recording and Replay
CDSSInterfaceManager	Class for DSS Communication
CStatusManager	Class for Module's Status
CFileManager	Class for Recorded File Management

DSS 메시지 인터페이스는 CDSSInterfaceManager 클래스 그리고 기록 및 재생은 CRNR 클래스와 매핑된다. DSS 메시지는 기본적으로 송신, 수신, 검색 그리고 초기화 기능을 MOMAT을 통해 자동으로 구현할 수 있으며, 이러한 기본 기능은 메시지의 Publisher/Subscribe 설정 및 메시지 형식에 따라 생성 여부가 결정된다. 컴포넌트에서 특정 메시지가 Publisher로 설정된다면 해당 메시지에 대한 송신 기능이 생성되고, Subscribe로 설정된다면 수신 기능이 추가된다. 물론, Publisher와 Subscribe를 모두 설정하는 것도 가능하며 이에 해당하는 사후분석은 모든 메시지에 대하여 송신 및 수신 기능이 구현된다. 메시지 형식은 단일 메시지인지 메시지의 집합인지 여부에 따라 Data형과 Table형으로 구분된다. 송신 및 수신 기능은 DSS 메시지 형식에 영향을 받지 않는 공통된 구조를 가지지만, 검색과 초기화는 메시지 형식에 따라 차이가 있다. 다만, 메시지 형식에 의한 차이가 있을 뿐 동일한 메시지 형식의 검색 및 초기화 기능은 완전히 같은 동작을 한다. 결과적으로 사후분석에서는 메시지의 수만큼 동일한 동작을 하는 함수가 기능별로 하나씩 자동으로 생성된다.

사후분석은 높은 신뢰성을 얻기 위해 최소 수백 개에 달하는 메시지들을 수집할 필요가 있다. 이때 동일한 동작을 함에도 그 기능을 호출하는 메시지가 다르다는 이유로 별도 함수로 구분하여 생성하는 구조로 구현이 되어 있다. 아래의 표는 DSS 메시지 1개 당 자동으로 생성되는 기능들의 소스코드 길이를 나타낸다.

Table 2. Number of source code lines for function

Function	Source Code Line
Send	50
Callback	4
Remove/Clear	4~8
Find	13

결과적으로 하나의 메시지 당 약 70라인 이상의 소스코드가 자동 생성됨을 확인할 수 있다. 소스코드가 길어지는 만큼 소프트웨어 빌드나 신뢰성시험의 시간이 증가하게 되고, 동일한 기능의 수정을 하게 되면 메시지의 수만큼 반복하여 작업을 해야 하므로 작업 효율성이 크게 감소한다.

2. Development Process

함정 전투체계 컴포넌트의 메시지 변경 및 반영 절차는 Fig. 5와 같이 이루어진다.

Fig. 5. Exist development process

소스코드 형상 관리 툴인 SVN(Subversion), 최신 소스코드를 기준으로 빌드하고 결과물을 배포하는 빌드 센터 그리고 메시지 관리 체계인 MOMAT은 별도로 운용된다. MOMAT에서는 특정 컴포넌트의 메시지를 변경한 경우, 변경된 메시지에 대한 공통 기능이 구현된 소스코드 출력 기능을 제공한다. 변경된 메시지에 대하여 공통 기능이 구현된 소스코드를 기존 소스코드에 반영하는 작업을 거치게 된다. 일반적으로 컴포넌트들은 MOMAT에서 출력된 소스코드를 그대로 적용해도 되는 구조를 가지고 있다. 다만, 필요에 의해 메시지 및 공통 기능을 구현한 소스코드 파일을 수정할 수 있고, 이러한 경우 덮어쓰는 과정에서 기존 코드와 충돌이 발생할 가능성이 있다. 따라서 기존 소스코드에 변경된 메시지에 대한 기능만 반영하는 것이 개발 안정성 측면에서 유리하다. 수정이 완료된 소스코드는 SVN에 입고되고, 선택한 컴포넌트를 최신 버전 기준으로 빌드함으로써 결과적으로 모듈이 생성된다.

앞선 절차에서 Step 3과 4에 해당하는 변경된 소스코드 반영 과정은 개발자의 업무 속도나 수정이 필요한 소스코드의 양에 따라 작업 처리에서 많은 시간이 소요될 수 있다. 그러나 사후분석의 경우 자동 생성되는 공통 기능들이 모두 같은 구조를 가진다. 이는 재사용이 가능한 공통 요소와 형식이나 상황에 따라 변경되어야 하는 가변 요소만 정확히 식별할 수 있다면 변경 메시지에 대한 소스코드 수정사항 반영을 자동화할 수 있다는 것을 의미한다. 결론적으로 사후분석은 소스코드 반영과 관련된 절차만 자동으로 처리하도록 동작을 변경하면 이 외의 절차까지도 모두 단순화할 수 있는 구조를 가진다.

본 논문에서는 MOMAT, SVN 그리고 빌드 센터를 연계하여 MOMAT에 메시지 변경 내역을 반영하고 빌드 센터에서 사후분석의 빌드 요청만 하면 변경 메시지 관련 소스코드 반영 및 신뢰성 시험을 포함한 최신 산출물 생성까지의 모든 동작을 자동으로 처리하는 체계를 제안한다.

IV. The Proposed Scheme

이번 장에서는 사후분석의 동작 구조를 개선하고 소스코드의 재사용율을 높이는 방법을 제안한다. 사후분석은 메시지에 따른 일부 식별 정보만 제외하면 기본적으로 공통 기능들은 내부적으로 동일한 동작을 하도록 설계되어 있으며 적합한 디자인 모델을 적용한다면 보다 효율적인 구조로 개선할 수 있다. 이에 사후분석에 휘처 모델을 적용하여 공통요소와 가변요소를 식별하여 동작 구조를 최적화하였고 템플릿 메소드 디자인 패턴을 활용하여 실제 소스코드의 구조를 개선 및 구현하였다. 그리고 개선된 사후분석 모델을 함정 전투체계의 개발 프로세스에 활용하는 방안을 제시한다. 함정 전투체계의 개발 프로세스는 메시지에 따른 소스코드의 변경 사항을 기존의 형상에 반영한다면 MOMAT을 비롯하여 SVN 및 빌드 센터와 같은 개발 도구들과 단순 명령어의 전달만으로 연계하기 용이한 구조를 가지고 있다. 이러한 구조적 이점을 활용할 수 있는 함정 전투체계 개발 프로세스 자동화에 대한 설명과 구현에 방법을 본 장에서 제안한다.

1. Proposed Post-Analysis Model

본 절에서는 사후분석의 구조를 개선하는 방안을 제시한다. 현재 함정 전투체계에 속한 컴포넌트의 DSS 메시지 공통 기능들은 동일한 동작을 하는데 Table.3과 같이 다른 메시지라는 이유로 각 메시지마다 별도로 기능을 구현하고 있다. 이러한 구조는 소프트웨어 관리 및 소스코드의 재사용 측면에서도 유용하지 않다.

Table 3. Exist CDSSInterfaceManager class sample source code

Exist CDSSInterfaceManager.cpp

```
...
void CDSSInterfaceManager::On_CSCI_FUNC_1(int i_iDomain, ...
, CSCI_FUNC_1* i_pData, ...)
{
   ...
   m_cRNR.SaveToFile(reinterpret_cast<char*>(i_pData, ...);
   ...
}

void CDSSInterfaceManager::On_CSCI_FUNC_2(int i_iDomain, ...
, CSCI_FUNC_2* i_pData, ...)
{
   ...
   m_cRNR.SaveToFile(reinterpret_cast<char*>(i_pData, ...);
   ...
}
...
```

기존에는 DSS 메시지와 관련된 정의와 기능들을 Fig.4의 CDSSInterfaceManager 클래스에서 모두 관리했다.

즉, 하나의 클래스에서 수백 개의 메시지가 정의되고 각 메시지에 따른 송신, 수신, 검색 그리고 초기화 함수가 정의된 것이다. 결과적으로 필요시 호출되는 기능들만 구현된 클래스임에도 소스코드가 수만 개의 라인을 초과했다.

샘플 소스코드와 같이 자동 생성된 수신 기능은 메시지에 따른 함수 이름과 메시지 형식에 의한 함수 인자에서 차이가 있을 뿐 나머지는 완전히 구조가 일치하며 송신 기능을 포함한 나머지 기능들도 이와 동일한 형식을 가진다. 결론적으로 유사한 부분을 하나의 기능으로 구현하고 차이가 있는 부분을 공통 기능에서 접근하는 방식으로 동작한다면 소스코드 재사용 가능성을 높일 수 있으며, 이는 전체 소스코드의 길이의 감소로 이어진다.

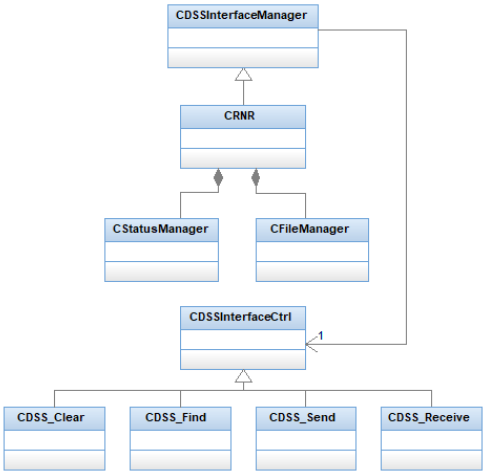

Fig. 6. Proposed Post-Analysis software class diagram

Table 4. Proposed Post-Analysis software class description

Class	Description
CDSSInterfaceCtrl	Class for DSS Message Information Management and Control
CDSS_Clear	Class for DSS Clear Config Management
CDSS_Find	Class for DSS Search Config Management
CDSS_Send	Class for DSS Transmit Config Management
CDSS_Receive	Class for DSS Reception Config Management

제안하는 기법을 적용한 사후분석의 클래스 다이어그램과 클래스 설명은 그림 Fig. 6 및 Table 4와 같다. Table 4는 새로 추가되었거나 변경된 클래스에 대하여 설명한다. 제안하는 사후분석 모델은 기존 소프트웨어 구조에 휘처 모델을 적용하여 공통 요소와 가변 요소를 구분한다. DSS 메시지의 송신, 수신, 검색 그리고 초기화 기능은 앞서 설명했듯이 동일한 구조를 가지므로 공통 요소로 식별할 수 있고, 메시지의 설정이나 형식에 따라 일부 구조체형이나 함수 인자 형식과 같은 일부분에서만 차이가 발생하므로 이를 가변 요소로 식별하면 사후분석의 동작 자체를 아래의 그림과 같이 단순화할 수 있다.

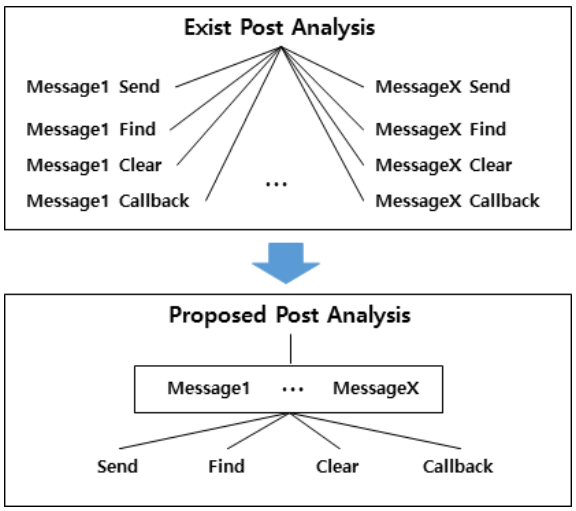

Fig. 7. Changed Post-Analysis structure

휘처모델을 적용한 모델을 실제 소프트웨어로써 구현하기 위하여 템플릿 메소드 디자인 패턴을 사용한다. 템플릿 메소드 디자인 패턴은 여러 클래스에서 공통으로 사용하는 기능을 상위 클래스에서 정의하고 가변 요소를 하위 클래스에서 구현하는 디자인 패턴이다. 공통 기능으로 식별된 송신, 수신, 검색 그리고 초기화 기능을 하나씩만 구현하고 가변 요소에 해당되는 해당하는 정보를 별도로 정의하였다. 이후 가변 요소가 호출이 되면 해당되는 정보를 반환받아 공통 기능에 접근 및 입력하여 동작하도록 하였다. 해당 기능을 구현하기 위하여 추가한 클래스가 CDSSInterfaceCtrl 클래스, CDSS_Clear 클래스, CDSS_Find 클래스, CDSS_Send 클래스 그리고 CDSS_Receive 클래스이다. 기존 DSS 메시지에 대한 모든 정의 및 기능이 구현되어 있던 CDSSInterfaceManager 클래스는 송수신되는 메시지에 대한 기능 호출만 하도록 변경된다. 실제 공통 기능과 기능을 수행하기 위한 세부 정보는 하위 클래스에서 정의한다. 공통 기능은 하나씩만 구현되며 이는 새로이 추가된 CDSSInterfaceCtrl 클래스에서는 DSS 메시지에서 사용되는 공통 기능만을 다룬다. 공통 기능의 사용이 요구되는 메시지 정보를 전달받은 CDSSInteface 클래스는 송신, 수신, 검색 그리고 초기화 기능에 맞게 하위 클래스에 접근하고, 전달받은 메시지가 해당 기능을 사용 가능 여부와 사용이

가능할 경우 기능을 사용하는데 필요한 세부 정보를 반환한다. 결과적으로 반환받은 정보로 공통 기능은 메시지 정보와 형식에 해당하는 동작을 하게 된다.

제안하는 사후분석 모델은 기존의 사후분석에 요구되는 동작들은 모두 동일하게 지원하면서도 공통 요소에서 가변 요소를 사용하는 동작 방식을 적용함으로써 소스코드의 소프트웨어 재사용율을 높였고, 공통 기능의 수정 및 관리에 있어서도 이점이 있다.

2. Proposed Development Process

앞서 개선된 사후분석 모델을 적용하여 함정 전투체계의 개발 프로세스를 개선하는 방안을 제시한다. 함정 전투체계의 개발 프로세스의 도구들은 명령어만으로 서로를 제어할 수 있어 연계가 용이한 구조를 가지고 있다. 이에 개선된 사후분석 모델을 사용한다면 함정 전투체계의 개발 프로세스의 특성과 연계하여 전체적인 개발 체계를 자동화할 수 있다. 본 절에서는 사후분석 개발 프로세스의 자동화 기법에 대하여 설명한다. 제안하는 함정 전투체계의 컴포넌트 개발 절차는 Fig. 8과 같다.

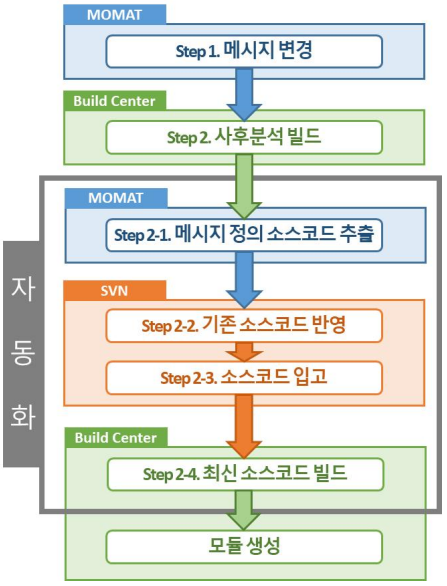

Fig. 8. Proposed development process

MOMAT에서 변경된 메시지 내용을 적용하는 Step 1은 기존의 체계와 완전히 동일하다. 그러나 Step 2의 경우 기존 체계는 사용자가 MOMAT에서 적용한 메시지가 반영된 소스코드를 출력하는 반면, 제안하는 체계는 MOMAT이 아닌 빌드 센터에서 사후분석 컴포넌트의 빌드를 요청하게 된다. 사후분석 빌드 요청 후 빌드 센터는 자체적으로 MOMAT에 변경 메시지를 반영한 소스코드의 출력 요청을 보낸다. 이후 소스코드 출력이 확인되면 앞서 제안한 사후분석 구조 단순화 기법을 실행하여 공통/가변 요소를 구분하고 이를 반영한 새로운 소스코드 산출물을 생성한다. 여기까지의 절차가 완료되었다면 수정된 소스코드를 SVN에 입고하게 되는데 해당 절차도 물론 자동으로 이루어지게 되고, SVN의 최신 사후분석 소스코드로 빌드한 모듈을 얻을 수 있다.

Fig. 8에서 자동화 그룹으로 묶여 있는 절차들을 기존 체계에서는 사용자가 모두 직접 접근하여 작업할 필요가 있었으나 제안하는 체계에서는 재사용 및 단순화가 가능한 요소들을 모두 식별함으로써 소스코드 수정 및 반영과 관련된 절차들을 자동화가 가능한 요소로 그룹화하여 구현할 수 있었다. 결과적으로 SVN과 관련된 작업이 모두 자동화됨으로써 사용자가 접근하는 체계가 MOMAT과 빌드 센터 2개로 감소하였다. 이러한 동작은 메시지 변경에 따른 소스코드 형상 수정을 사후분석의 구조 개선을 통해 모든 동작을 개발자의 개입 없이 자동으로 처리되도록 변경됨으로써 가능한 것이다.

제안하는 개발 프로세스 개선 방안은 사용자에게 요구되는 작업은 변경 메시지를 MOMAT에 반영하고 빌드센터에서 빌드 요청을 하는 것이 전부이므로 기존 체계를 사용할 때와 비교하여 사용자의 작업량이 감소하였고 이에 비례하여 사후분석 모듈을 생성하기까지의 작업 시간도 크게 감소하였다.

V. Software Evaluation

본 논문에서는 제안하는 기법의 성능을 평가하기 위해 메시지 수에 따른 소스코드 자동화 처리, 산출물 출력까지의 전체 프로세스 동작 시간 및 신뢰성시험 결과를 비교하였다. 제안하는 기법은 메시지의 수가 증가할수록 이에 비례하여 작업 시간이 감소하고 소스코드의 재사용율이 증가하므로 성능 검증에서 비교군은 메시지의 수로 설정하였다. 시험 환경과 시험에 사용한 PC의 사양은 Fig. 9, Table 5와 같다.

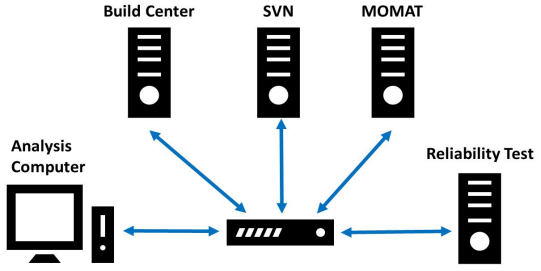

Fig. 9. Test environment

Table 5. Analysis computer specification

Item	Specification
CPU	Intel(R) Core(TM) i5-9400 CPU@2.90GHz
Memory (RAM)	16 GB
OS	Windows 10 x64

기존 개발 모델과 제안하는 개발 모델은 동일한 환경에서 적용 및 시험하였다. 실제 제안하는 개발 모델은 기존 개발 모델에 사후분석 자동화를 위한 소스코드 구조 변경 및 메시지 구분을 위한 스크립트 추가 그리고 개발 도구 연계를 위한 명령과 동작 순서 변경을 적용하였을 뿐이므로 기존 개발 모델 대비 개발 환경 성능의 변경은 요구되지 않는다.

1. Proposed Post-Analysis Model

1.1 Compare Created Source Code Lines

메시지들이 모든 공통 기능을 사용한다는 전제하에 시험을 수행하였다. 실제 메시지들이 사용하는 공통 기능을 식별해서 시험을 할 경우 어떤 메시지를 시험 비교군으로 선택하는지에 따라 소스코드 라인 차이에서 다양한 변수가 발생할 수 있으므로 이러한 변수 발생을 최소화 하고자 기존 사후분석 모델과 제안하는 사후분석 모델은 모든 공통 기능을 사용하는 것으로 설정하였다. Fig. 10과 Table 5는 제안하는 사후분석 모델과 기존 사후분석의 메시지 수에 따라 자동 생성되는 소스코드 라인 수를 비교 및 정리한 것이다.

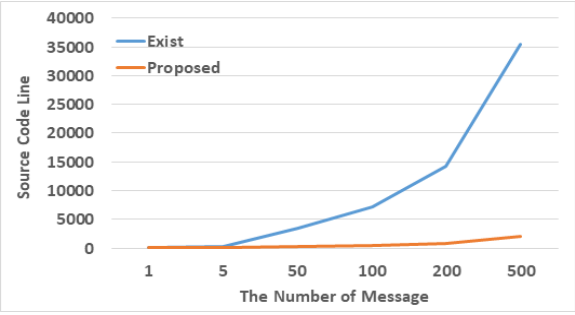

Fig. 10. Comparison of source code lines with number of messages

Table 6. Comparison of source code lines with number of messages

No. of Msg	Exist Model	Proposed Model	Diff.
1	71	75	-4
5	355	91	264
50	3550	271	3279
100	7100	471	6629
200	14200	871	13329
500	35500	2071	33429

메시지의 수가 한 개인 경우에는 기존의 사후분석 모델과 제안하는 사후분석 모델 모두 공통 기능만 생성된 것과 동일하므로 소스코드 라인 수의 차이가 크지 않다. 그러나 메시지가 2개 이상인 경우부터 차이가 크게 발생한다. 기존의 사후분석 모델은 메시지에 대한 공통 기능을 각각 구현하므로 메시지의 수 만큼 비례하여 소스코드의 길이가 증가한다. 반면 제안하는 사후분석 모델의 경우 공통 기능은 기능 별로 하나씩만 생성되고 각 기능을 사용하기 위한 메시지 형식이나 인자 값만 별도로 생성되므로 메시지 하나 당 최대 4줄의 소스코드 라인만 증가한다. 결과적으로 메시지 수가 늘어남에 따라 자동 생성되는 소스코드 라인의 차이가 비례하여 크게 증가함을 확인할 수 있다.

1.2 Compare Reliability Test Result

DSS 메시지가 변경되면 관련 동작을 구현하기 위해서는 소스코드의 수정도 함께 병행되어야 한다. 이때 수정된 소스코드가 반영된 산출물의 정상 동작 여부만큼이나 소스코드가 정해진 규칙에 따라 작성되었는지의 검사도 중요하다. 형상 관리되는 소스코드의 신뢰성 확보를 위하여 함정 전투체계에서는 소스코드의 수정과 함께 수정되는 컴포넌트의 SW 신뢰성 시험을 수행하며, 신뢰성 시험에는 LDRA 도구를 이용한다. LDRA 도구는 소프트웨어의 품질 확보를 위한 요구사항 추적, 코드분석, 시험 등의 기능을 제공하며, 국제 표준 코딩룰(MISRA, CWE, CERT 등)을 적용 잠재적 결함 검출뿐만 아니라 소프트웨어의 구조적 검증 및 기능 시험을 지원한다[17]. 함정 전투체계의 사후분석은 LDRA 도구의 정적 시험을 수행하여 검출된 결함을 조치하고 코딩룰에 맞게 형상을 관리한다. 아래의 표는 기존의 사후분석 모델과 제안하는 사후분석 모델의 메시지 수에 따른 LDRA 정적 시험 결과를 비교한 것이다.

Table 7. Comparision of LDRA test results with number of messages

No. of Msg	Exist Model	Proposed Model	Diff.
5	27	17	10
50	117	17	100
500	1017	17	1000

제안하는 사후분석 모델은 메시지 수에 상관없이 검출된 항목의 수가 17개로 모두 동일하다. 검출된 항목들도 실제 오류가 아닌 예외처리 항목이므로 코딩룰 상 오류가 없다. 그러나 기존의 사후분석 모델의 경우 메시지마다 고정적으로 2개의 예외처리 항목이 검출된다. 제안하는 사후분석 모델 대비 추가적으로 검출되는 항목들은 예외처리

항목이므로 사실상 기존의 사후분석 모델도 코딩룰을 제대로 지켜서 작성되었고 그대로 사용하더라도 아무런 문제가 되지 않는다. 다만, 검출 항목이 늘어나는 만큼 오류 및 예외처리 여부를 확인하는 시간이 증가하게 되고 이는 곧 작업량의 증가로 이어진다. 반면 제안하는 사후분석 모델의 경우 MOMAT에서 메시지를 정상적으로 변경하였다면 자동으로 생성된 소스코드의 LDRA 정적 시험 결과는 변경 전 시험한 결과와 무조건 동일하다. 결과적으로 별도의 검출 항목 식별을 할 필요가 없어지므로 전체적인 작업량이 크게 줄어드는 이점이 있다.

다음은 사후분석 모델 간 소요되는 신뢰성 시험 시간을 비교한다. DSS 메시지의 공통 기능이 모두 구현된 상태에서 특정 메시지에 대한 수신 기능 옵션이 변경되는 경우를 전제하였다. 기존 사후분석 모델의 경우 옵션값만 변경이 되더라도 CDSSInterfaceManager 클래스에 DSS 메시지에 대한 모든 기능들이 구현되어 있는 만큼 변경 사항이 없는 기능들도 포함하여 신뢰성 시험을 수행할 수 밖에 없다. 즉, 항상 DSS 메시지 수에 따른 최대 시험 시간이 소요되는 것이다. 반면에 제안하는 사후분석 모델은 메시지 수신 정보를 관리하는 CDSS_Receive 클래스만 신뢰성 시험을 수행한다. 하나의 클래스에서 모든 기능에 대한 정보를 관리하는 것에 비하여 소스코드 라인 수도 적고 구조도 단순한 만큼 기존 사후분석 모델과 비교하여 적은 시험 시간이 소요된다. DSS 메시지 수에 따른 신뢰성 시험 시간의 차이는 Table 8과 같다.

Table 8. Comparison of LDRA test times with number of messages

No. of Msg	Exist Model	Proposed Model	Diff.
5	2 Min	1 Min	1 Min
50	3 Min	1 Min	2 Min
500	16 Min	1.5 Min	14.5 Min

결과적으로 제안하는 사후분석 모델은 기존 사후분석 모델 대비 짧은 SW 신뢰성 시험 시간을 보장하며, 메시지 수에 따라 고정적으로 검출되던 예외처리 항목이 제거됨으로써 전체적인 작업 시간을 크게 줄였다.

2. Proposed Development Process

OOO함정 전투체계의 개발 체계 내에서 기존 모델과 제안하는 체계 모델의 작업 시간을 비교하였다. 기존 모델의 절차에서 사용자나 실제 작업량 및 작업 환경에 따라 작업 시간 차이가 발생할 수 있다. 특히, 변경된 소스코드를 기존 코드에 반영하는 작업은 평균적인 작업 시간을 산출하는 것이 어렵다. 따라서 본 시험에서는 Fig.4 기존 체계 절차의 변경 메시지에 대한 소스코드 반영은 메시지 당 6초의 작업시간이 소요된다고 전제하였다. 실제로는 6초 이상의 시간이 소요되지만 비교 데이터 산출을 위하여 최소 작업 시간을 설정하였다. 본 시험의 시험 비교군은 아래의 3가지로 설정하였다.

· Model-1 : 기존 개발 프로세스
· Model-2 : 기존 개발 프로세스에 사후분석 소스코드 자동화 적용
· Model-3 : 제안하는 사후분석 개발 자동화 프로세스

메시지 수를 각 5/50/500개인 경우로 나누어 성능을 비교하였다. 개발 모델에 따른 작업 시간을 비교한 결과는 Table 9와 같다.

Table 9. Comparison of working times with number of messages

No. of Msg	Model-1	Model-2	Model-3
5	5 Min	4 Min	3.5 Min
50	10.5 Min	5.5 Min	5 Min
500	59.5 Min	10 Min	9 Min

작업량이 많은 소스코드 반영 절차를 최소 시간으로 전제하여 시험하였음에도 사후분석 소스코드 자동화를 적용한 모델들과 Model-1의 작업 시간이 메시지 수에 따라 적게는 1분에서 많게는 약 50분까지 시간 차이가 발생하였다. 변경 메시지의 수가 적은 경우에는 자동화 기법의 적용 여부에 따른 시간 차이가 크지 않지만 메시지 수가 늘어남에 따라 제안하는 사후분석을 적용한 개발 모델의 작업 시간 효율이 급격히 증가함을 확인할 수 있다. 또한, 해당 작업과 병행해야 하는 신뢰성 시험 시간까지 작업 시간에 포함하면 제안하는 개발 모델의 적용 여부에 따른 작업 시간 차이는 더욱 증가한다. Model-1에서 사후분석 자동화만 적용한 Model-2는 사용자의 직접적인 개입을 상대적으로 줄이긴 했어도 개발 도구 간 이동 및 사용하는 절차가 Model-3에 비하여 많고 이는 작업 시간의 증가로 이어졌다. 실제로 Table 9에서 Model-3가 Model-2에 비해 작업 시간이 상대적으로 감소하였음을 확인할 수 있다.

개발 과정에서 변경 메시지가 많거나 개발자의 직접적인 개입이 증가할수록 기존 개발 모델과 제안하는 개발 모델 간 작업 시간 차이가 증가한다. 결과적으로 제안하는 개발 모델의 적용은 개발 인력의 작업 공수를 기존 모델 대비 현저히 낮게 책정할 수 있을 뿐만 아니라 개발 비용의 절감 효과를 기대할 수 있다.

VI. Conclusions

본 논문에서는 개선된 사후분석 모델과 이를 적용한 개발 체계 자동화 기법을 제안하였다. MOMAT에서 출력한 메시지의 Publisher/Subscriber 정보를 제안하는 시스템에서 사용 가능한 형태로 변환하였고, 메시지 형식 기준으로 정리된 목록에 휘쳐 모델을 적용하여 공통 요소와 가변 요소를 식별하였다. 템플릿 메소드 디자인 패턴을 사용하여 기존 사후분석 모델에 적용하고 별도로 운용되는 체계들을 연계하였다. 제안하는 사후분석 모델을 적용한 연동 개발 체계는 변경된 메시지 정보를 반영한 소스코드를 자동으로 출력하였고 결과적으로 새로운 사후분석 산출물을 생성함을 확인하였다. 기존의 개발 체계 대비 전체적인 소스코드 라인 수를 크게 감소 시켰으며, 이에 비례하여 소프트웨어 빌드 및 신뢰성 시험 시간도 감소하였다. 개발 및 유지보수 시간은 개발 비용 책정에 있어 큰 부분을 차지하는 요소이므로 제안하는 기법을 적용함으로써 개발 및 유지보수의 효율성 증대와 더불어 개발 비용 감소 효과를 기대할 수 있다. 또한, 사후분석과 같이 DSS 메시지와 관련된 공통 기능을 기준으로 공통 요소와 가변 요소를 구분할 수 있는 컴포넌트라면 제안하는 기법을 그대로 적용할 수 있으므로 뛰어난 확장성을 가진다. 특히, 함정 전투체계의 경우 사후분석과 같이 가변 요소들의 처리 방식이 동일한 구조를 가지는 메시지 기록 기능들에 제안하는 템플릿 메소드 디자인 패턴을 적용한 개발 모델을 적용할 수 있으므로 전체적인 개발 체계 관점에서의 작업 효율 증가를 기대할 수 있다.

REFERENCES

[1] Sun-Ju Go, "Network Centric Warfare to Prepare for Combat Systems Development and Future Direction," The Magazine of the IEIE, Vol. 37, No. 11, pp. 27-38, Nov. 2010.

[2] Je-young Yu, and Jin-hee Park, "A Message Management System for Cooperative Message-based Interface Development," Journal of KIISE : Computing Practices and Letters, Vol. 14, No. 6, pp. 609-613, Aug. 2008.

[3] Schlesselman. J. M, Pardo-Castellote. G and Farabaugh. B., "OMG data-distribution service (DDS): architectural update," Military Communications Conference, Vol. 2, pp. 961-967, Aug. 2004.

[4] Je-young Yu, and Jin-hee Park, "Message Definition and Management System for Cooperative Message-based Interface Development," Proc. of the KIISE Korea Computer Congress 2007, pp. 40-41, Oct. 2007.

[5] Jun-hyeong Kim, and Keun-sung Bae, "Association between Object and Sonar Target for Post Analysis ofSubmarine Engaged Warfare Simulation," Journal of the Korea Society for Simulation, Vol. 26, No. 3, pp. 65-72, 2017.

[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, Oct. 2015.

[7] Jin-yong Im, and Dong-seong Kim, "Performance Evaluation of Virtualization Solution for Next Generation Naval Combat Systems," Journal of IEIE, Vol. 56, No. 2, Feb. 2019.

[8] Seung-mo Jung and Moon-seok Yang, “A Study on the Development of Efficient Naval Combat System Tactical Record and Replay Software,” Proceedings of the Korean Society of Computer Information Conference, pp. 35-36, Jul. 2019.

[9] Data Distribution Service for Real-Time Systems, OMG Available Specification, Jan. 2007.

[10] Jung-hyeok Cha, Jong-woo Lee, Jae-min Lee, and Dong-sung Kim, “Real-time middleware application and trend research for civilian and military ICT convergence technology,” Journal of KICS, Vol. 37, No. 10, pp. 47-54, Oct. 2020.

[11] Ju-won Lee, "Development of Message Define & Management System based on Distributed Processing Environment for Naval Combat Systems," KIISE Transactions on Computing Practices, Vol. 23, No. 12, pp. 670-676, Dec. 2017.

[12] R.E. Johnson snd B. Foote "Designing Reusable Classes," Journal of Object-Oriented Programming, Vol. 1, No. 2, pp. 22-35, Jul. 1988.

[13] Min-seong Kim, and Soo-yong Park, "A Domain Analysis Method for Software Product Lines Based on Goals, Scenarios, and Features," Journal of KISS : Software and Applications, Vol. 33, No. 7, pp. 589-604, Jul. 2006.

[14] Kyo-chul Kang, Sa-joong Kim, Jae-joon Lee, Ki-joo Kim, Eui-seob Shin, and Moon-hang Huh, "FORM: A Feature-Oriented Reuse Method with Domain-Specific Reference Architectures," Annals of Software Engineering, pp. 143-168, May. 1998.

[15] Kyung-mo Yang, Yoon-Ho Jo, Kyo-chul Kang, "Modeling FORM Architectures Based on UML 2.0 Profiling," Journal of KISS :Software and Applications, Vol. 36, No. 6, pp. 431-442, Jun. 2009.

[16] Source Making, https://sourcemaking.com/

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

Authors

Chang-Won Seo received the B.S. and M.S degree in Information and Communication Engineering from Chungbuk University, Korea, in 2016. He is currently working in Hanwha Systems Co. from 2020.

He is interested in Naval Combat System, Software Reuse, Design Pattern and so on.