JKSCI

한국컴퓨터정보학회논문지
Journal of The Korea Society of Computer and Information
Vol. 31 No. 3, pp. 179-189, March 2026
https://doi.org/10.9708/jksci.2026.31.03.179

A Study on Object-Oriented Standard Architecture for Export-Oriented IFF Display Software in Naval Combat Systems

Sung-Uk Jo*, Su-Seok Park*, Young-San Kim*

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

[Abstract]

This study proposes an object-oriented standard architecture to improve the maintainability of Identification Friend or Foe (IFF) display software in naval combat systems. In export projects, frequent equipment replacement and UI modifications increase the burden of maintenance. To address this issue, the proposed approach applies a feature model to separate common and variable elements, and adopts `enhance extensibility. Furthermore, the Naval Shield Component Platform (NSCP) concept is applied to IFF display software modules to ensure compatibility with interfacing software, enabling flexible adaptation to equipment and technology changes through UI module replacement and strategic algorithm modification. Evaluation results demonstrate quantitative improvements, including a 35.8% reduction in average Cyclomatic Complexity (CC), a 12.3% increase in Maintainability Index (MI), and a 42.4% reduction in maximum Lines of Code (LOC). These findings confirm that the proposed architecture effectively addresses the requirements of export projects while also alleviating prolonged reliability testing burdens in domestic projects, thereby contributing to enhanced competitiveness in the defense industry.

▸ **Key words**: Naval combat system, Identification Friend or Foe (IFF), Display software, Object-oriented design (SOLID), Feature model, Standard architecture, Reusability

[요 약]

본 연구는 수출 사업에서의 함정 전투체계 피아식별기(IFF) 전시 소프트웨어 유지보수성 향상을 위해 객체지향 표준 아키텍처를 제안한다. 수출 사업에서는 잦은 장비 교체와 UI 수정으로 유지보수 부담이 증가한다. 이를 해결하기 위해 피처 모델을 적용해 공통성과 가변성을 분리하고, SOLID 원칙 기반의 계층·인터페이스 설계를 도입하여 결합도를 낮추고 확장성을 확보하였다. 또한 기존 연구의 NSCP(Naval Shield Component Platform) 개념을 IFF 전시 소프트웨어 모듈에 적용해 연동 SW와의 호환성을 확보하고, 장비·기술 변경 시 UI 모듈 교체와 알고리즘 변경을 용이하게 하였다. 평가 결과 평균 CC 35.8% 감소, MI 12.3% 향상, 최대 LOC 42.4% 감소가 확인되어, 제안된 아키텍처가 수출 사업 요구에 효과적으로 대응하며 국내 사업에서도 장기화된 신뢰성 시험으로 인한 개발 지연 문제 해소에 기여함을 보여준다.

▸ **주제어**: 함정 전투체계, 피아식별기, 전시 소프트웨어, 객체지향 설계(SOLID), 피처모델, 표준 아키텍처, 재사용

• First Author: Sung-Uk Jo, Corresponding Author: Sung-Uk Jo
*Sung-Uk Jo (josunguk1104@hanwha.com), Naval R&D Center, Hanwha Systems
*Su-Seok Park (suseok.park@hanwha.com), Naval R&D Center, Hanwha Systems
*Young-San Kim (y.san.kim@hanwha.com), Naval R&D Center, Hanwha Systems
• Received: 2026. 02. 05, Revised: 2026. 03. 09, Accepted: 2026. 03. 12.

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

I. Introduction

함정 전투체계(Combat Management System, CMS)는 센서·무장·지원 기능을 통합 운용하여 표적 탐지, 추적, 무장 통제, 교전 수행 등 핵심 임무를 수행한다. 이 중 전시(Human-Computer Interface, HCI) 소프트웨어는 운용자가 전술 상황을 직관적으로 인지하고 신속하게 의사결정을 내릴 수 있도록 지원하는 핵심 인터페이스이다.

그러나 장비 교체, 기능 추가, UI 수정 등 운용 환경 변화가 발생하면 소프트웨어 수정이 불가피하며, 이는 개발 기간과 비용 증가로 직결된다[1-2]. 특히 수출 사업의 경우, 동일한 CMS 버전을 사용하더라도 고객 요구사항 및 함정 규모에 따라 IFF 하드웨어(HW)가 달라지는 경우가 많습니다. 예를 들어 A 사업에서는 X사, B 사업에서는 Y사의 IFF가 사용될 수 있으며, 이로 인해 연동 방식과 화면 표시 요소 전반에 걸쳐 대규모 수정 및 검증 작업이 요구된다. 국내 사업 역시 소스코드 변경 시 필수적인 정적·동적 시험으로 인해 개발 부담이 가중된다[3-4]. 잦은 변경과 회귀시험, 제한된 일정, 개발자 피로도 증가는 오류 가능성을 높이고 품질과 효율성에 부정적 영향을 미친다.

본 연구는 전시 소프트웨어 수정 시 최소한의 소스코드 변경으로 유지보수 효율을 높이고, 고품질·저비용의 설계·개발·유지보수를 실현하는 표준 아키텍처를 제안한다. 이를 위해 장비 변경에 효율적으로 대응 가능한 표준 아키텍처를 설계하고, 기존 소프트웨어를 단일 표준 형식으로 재구성하였다. 또한 공통·가변 요소 분석을 통해 피처 모델을 적용하고, SOLID 원칙 기반 객체지향 설계(OOP)와 전략 패턴(Strategy Pattern)을 활용하여 모듈 대체 기법을 구현하였다. 마지막으로, 제안된 아키텍처 기반 소프트웨어와 기존 시스템을 성능·개발 요소·비용 측면에서 비교·평가하여 그 효과를 검증하였다.

본 연구의 주요 기여는 함정 전투체계 전시 소프트웨어의 유지보수성을 획기적으로 향상시키는 범용 표준 아키텍처를 제시한 것이다. 이를 통해 장비 교체·UI 변경 시 소스코드 수정 범위를 최소화하고 재사용성을 극대화함으로써, 기존 아키텍처 대비 개발 기간과 비용을 유의미하게 절감할 수 있음을 실증적으로 입증하고, 나아가 다양한 장비 환경에 적용 가능한 전투체계 소프트웨어 설계 표준을 제안한다.

본 연구의 구성은 다음과 같다. 제2장에서는 시스템 개념과 구조 및 개선 방법론을 제시하고, 제3장에서는 높은 의존성으로 인한 비용·시간 문제를 분석하며 개선 방안을 논의한다. 제4장에서는 제안된 아키텍처를 적용해 효과를 검증하고, 제5장에서는 연구 결과와 기대효과를 종합하여 서술한다.

II. Preliminaries

1. Related works

1.1 Naval Combat Management System

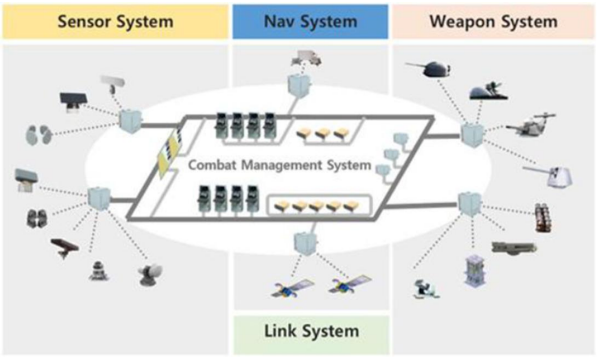

Fig. 1. System Architecture of Naval Combat Management System

Fig. 1은 함정 전투체계(Combat Management System, CMS)의 시스템 아키텍처를 간략히 도식화한 그림으로, CMS 소프트웨어가 각 하드웨어 노드에 탑재되어 운용되는 방식을 보여준다. 센서, 무장, 통신 등 외부 장비를 제어하는 연동 CSCI, 항해 지원 등 보조 기능을 담당하는 체계지원 CSCI, 체계 제어 및 상태 감시를 수행하는 체계관리 CSCI의 구성 관계를 확인할 수 있다[5-6].

CMS는 데이터 링크, 교전·지휘통제, 훈련, 공통 기능으로 구성되며, 단일 무기체계의 사격통제뿐 아니라 다중 센서와 무장을 통합·제어하는 핵심 기능을 수행한다(Fig. 2).

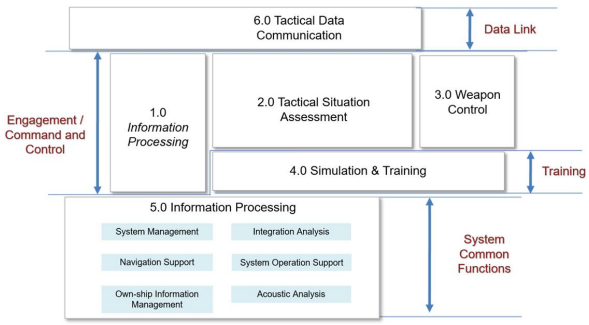

Fig. 2. Function of Combat Management System

Fig. 2는 함정 전투체계(CMS)의 주요 기능을 보여주는 그림으로, 각 핵심 요소가 어떻게 상호 작용하는지 설명한다.

함정전투체계 소프트웨어 개발은 체계 요구분석 → 기본/상세 설계 → 구성품 설계·제작 → 공장수락시험(FAT) → 육상체계통합시험(LBTS) → 개발시험평가(DT) → 운용시험평가(OT) 순으로 진행된다.

소프트웨어 형상은 관리 단위에 따라 CSCI(Computer Software Configuration Item), CSC(Computer Software Component), CSU(Computer Software Unit)로 구분된다. CSCI는 형상관리의 기본 단위이며, CSC는 독립 배포가 가능한 구성요소, CSU는 독립 시험이 가능한 최소 단위이다. 개발 프로세스에서는 하위 단위에서 상위 단위로 순차 검증이 이루어진다.

1.2 Human-Computer Interface(HCI)

전시 소프트웨어는 CMS 운용자가 전술 상황과 장비 상태를 직관적으로 인지하고 신속하게 의사결정을 내릴 수 있도록 시각 정보를 제공하는 핵심 모듈이다. 전투체계는 장비의 종류와 역할에 따라 개별 HCI를 보유하며, 운용자는 이를 통해 함정, 센서, 레이더, 무장을 제어·운용한다.

본 연구에서는 전투관리체계의 IFF 전시 소프트웨어를 중점 대상으로 이전 사업 사례를 비교·분석하였다. 이를 통해 공통 요소와 가변 요소를 체계적으로 분류하고, 기능을 전시 기능과 동작 기능으로 구분하여 분석 효율성을 높였다. 이러한 구분은 향후 표준 아키텍처 설계 시 모듈화와 재사용성을 극대화하는 기반이 된다[7].

1.3 Feature Model

피처(Feature)란 사용자 또는 개발자가 식별할 수 있는 소프트웨어 시스템의 구별되는 특성을 의미한다. 피처 모델(Feature Model)은 이러한 개념에서 발전한 기법으로, 사용자와 개발자 간의 의사소통 수단으로 활용될 뿐 아니라 시스템의 공통 요소와 가변 요소를 식별하는 데 널리 사용된다.

본 연구에서는 피처 모델을 적용하여 전투관리체계의 체계지원 소프트웨어를 대상으로 이전 사업 사례를 비교·분석하였으며, 이를 통해 공통 요소와 가변 요소를 효과적으로 도출하였다[8].

1.4 NSCP(Naval Shield Component Platform)

본 논문에서 언급하는 표준 연동 아키텍처는 NSCP(Naval Shield Component Platform)로, 함정 전투체계 소프트웨어의 효율적인 개발을 위해 제시된 공용 아키텍처이다(Fig. 3). Fig. 3은 NSCP(Naval Shield Component Platform)의 간략화된 클래스 다이어그램으로, 객체지향 프로그래밍(OOP) 기반의 설계 구조를 보여준다. 네 종류의 요소 클래스가 인터페이스 클래스를 통해 간접적으로 연결되어 모듈 간 결합도를 낮추고 유연성을 확보하는 방식을 확인할 수 있다. 이 아키텍처는 객체지향 프로그래밍(OOP)과 다섯 가지 기본 설계 원칙인 SOLID를 준수하며, 연동 CSCI 소프트웨어를 위한 표준 연동 아키텍처로 시작하였다.

NSCP는 기적용되어 유용성이 검증되었으며, 설계 구조상 연동 CSCI뿐 아니라 다른 CSCI 소프트웨어에도 적용 가능한 높은 범용성, 확장성, 유지보수성을 갖춘다[9].

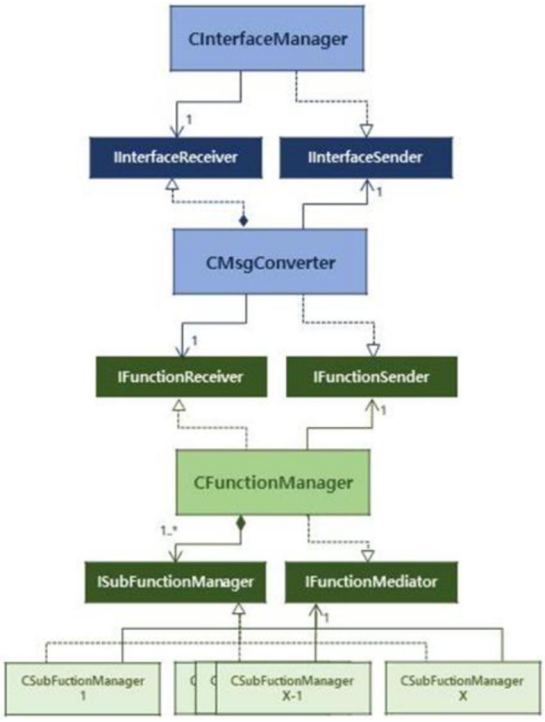

Fig. 3. Simplified Class Diagram of Naval Shield Component Platform

본 연구에서는 기 검증된 NSCP를 활용하여 객체지향적 프로그래밍 설계를 진행함으로써, 모듈의 높은 응집도와 낮은 결합도라는 구조적 토대를 마련하고자 한다. 다만, 이러한 구조적 이점이 실제 분산 환경에서도 그대로 유지되기 위해서는 모듈 간의 데이터 통신 방식이 함께 고려되어야 하며, 특히 기존 방식에서 발생하는 통신 의존성 문제가 해결되어야 한다.

1.5 DDS

함정 전투체계 내부 노드 간 통신은 DDS(Data Distribution Service)를 통해 수행된다. DDS는 비영리 기구인 OMG(Object Management Group)에서 제정한

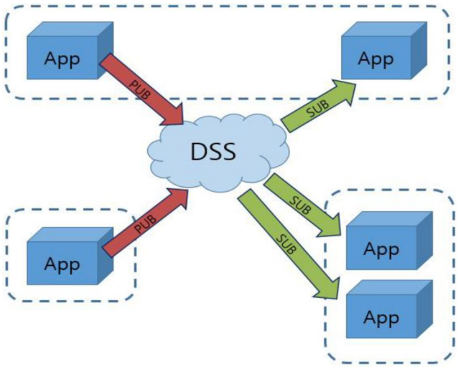

Fig. 4. Publish-Subscribe Model of DSS

표준으로, 분산 시스템 환경에서 데이터 교환 및 공유를 지원하며, 미들웨어 기반의 Publish/Subscribe 통신 방식을 채택한다(Fig. 4). Fig. 4는 DDS(Data Distribution Service)의 Publish-Subscribe 모델을 보여주는 그림으로, 함정 전투체계 내부 노드 간 통신 방식을 설명한다. 메시지 구조가 변경될 경우, 해당 메시지를 처리하는 DDS 관련 클래스는 반드시 수정되어야 하며, 이는 시스템 중단 및 유지보수 비용 증가를 야기할 수 있다. 또한 DDS 통신을 사용하기 위해서는 NodeAgent라는 별도의 응용 프로그램이 필요하며, 메시지 변경 시 NodeAgent 역시 재빌드 및 재설치 작업이 요구된다[10].

따라서 본 연구에서는 NSCP의 구조적 이점을 훼손하지 않으면서 DDS 통신으로 인해 발생하는 이러한 의존성 문제를 해결하기 위해, DDS 통신을 사용하는 모듈을 분리하고 유연한 설계를 적용하고자 한다.

1.6 S.O.L.I.D

S.O.L.I.D는 Robert C. Martin이 2000년대 초반에 제안한 객체지향 프로그래밍 및 설계의 다섯 가지 기본 원칙을 의미한다. 이 원칙들은 시간이 지나도 유지보수와 확장이 용이한 시스템을 구축하기 위한 지침으로, 이를 적용하면 소프트웨어 개발 과정에서 코드 스멜(Code Smell)을 만들어내는 안티패턴과 설계결함, 구조 결함 등을 제거할 수 있다. 또한 가독성과 확장성을 향상시키기 위해 소스코드를 리팩토링시에도 적용된다[11-12].

S.O.L.I.D 원칙은 애자일(Agile) 및 적응형 소프트웨어 개발 방법론의 핵심 전략 중 하나로, 설계 품질을 높이고 변경에 유연하게 대응할 수 있도록 한다. 본 논문에서는 전시 소프트웨어 표준 아키텍처를 설계하는 과정에서, 기존 전시 소프트웨어 클래스의 일부 기능을 새로운 클래스로 분리·추가하는 작업에 S.O.L.I.D 원칙을 적용하였다. 이를 통해 모듈 간 결합도를 낮추고, 유지보수성과 확장성을 동시에 확보하고자 하였다.

피아식별기(IFF)는 질문기와 응답기로 구성되어 전파 또는 음파를 이용해 적과 아군을 구분하며, 공중 및 수상 이동수단을 포함한 다양한 무기체계에서 활용되는 암호장비이다. 본 연구의 주대상인 함정 전투체계의 IFF(Identification Friend or Foe) 전시 소프트웨어는 피아식별기의 연동단 인터페이스 모듈과 통신하여 아군 식별 기능을 수행하며, 운용자가 다기능콘솔을 통해 직접 제어·관리할 수 있도록 지원하는 MFC(Microsoft Foundation Class) 기반 소프트웨어이다.

III. The Proposed Scheme

기존 IFF 전시 소프트웨어의 구조는 연동 소프트웨어뿐 아니라 장비 자체에 대한 의존성이 높아, 성능 향상이나 기능 변화에 따른 수정 시 상당한 비용과 시간이 소요되는 문제가 있다. 이에 따라 장비 및 연동 소프트웨어 변경에 유연하게 대응할 수 있는 구조 개선이 필요하다. 본 연구에서는 IFF 전시 소프트웨어를 표준 아키텍처 기반으로 재설계하였으며, 그 과정은 Table 1과 같이 세 단계로 진행하였다.

Table 1. Primitive Feature of Variable domain in IFF HCI 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. Identification

기존 IFF(Identification Friend or Foe) 전시 소프트웨어의 동작 범위를 체계적으로 분석하여, 수행 기능을 식별하고 목록화하였다. 본 연구에서는 실제 함정 전투체계 프로젝트의 소스 코드 구조를 심층 분석을 통해 다음과 같은 핵심 기능 모듈을 식별하였다.

1.1 Core Functional Module Identification

- 장비 제어 관리자: IFF 질문기 설정과 장비의 지정 제어(Designation Control), 침묵 제어(Silence Control) 등을 담당하는 핵심 모듈로, 운용 모드를 설정하고 상태를 모니터링한다. 운용자의 제어 명령을 장비가 이해할 수 있

는 프로토콜로 변환하며, 장비로부터 수신된 상태 정보를 전투관리체계에 전달하는 중개자 역할을 수행한다.

- 데이터 통신 계층: DDS(Data Distribution Service) 기반의 데이터 분산 처리를 담당하는 통신 모듈로, 분산된 시스템 환경에서 실시간 데이터 교환을 지원한다. 발행/구독(Publish/Subscribe) 패턴을 기반으로 하여 노드 간 효율적인 데이터 전송과 동기화를 보장하며, 실시간성이 요구되는 전투 상황에서 안정적인 통신을 제공한다.

- 메시지 변환 계층: 전투관리체계(CMS)와 연동 소프트웨어(ICU)간 메시지 포맷 변환을 담당하는 모듈로, 이기종 시스템 간 데이터 호환성을 보장한다. CMS의 표준 메시지를 장비별에 따라 달라지는 연동 소프트웨어 특화 형식으로 변환하거나 그 반대 변환을 수행하여 시스템 통합의 유연성을 확보한다.

- 이벤트 처리 계층: 시스템의 다양한 이벤트와 상태 변화를 실시간으로 처리하는 이벤트 핸들러 모듈로, 장비 상태 변화, 운용자 입력, 외부 요청 등을 수신하여 적절한 처리 모듈로 전달하는 중앙 허브 역할을 한다.

- UI 제어 계층: 운운용자 인터페이스를 제어하는 다이얼로그 및 모드별 전시 모듈로, IFF의 다양한 운용 모드(Mode 1, 2, 3A, 5 등)에 따라 상이한 전시 정보를 제공한다. 각 모드별 특성에 맞는 데이터 시각화와 사용자 상호작용을 지원한다.

Step 2. Classification

식별된 기능들을 실제 구현 구조를 기반으로 체계적으로 공통 요소와 가변 요소로 분류하였다.

2.1 Common Elements

- 표준 인터페이스 구조: 모든 서브 기능 모듈이 구현해야 하는 기반 인터페이스로, 상태 업데이트, 정보 조회, 데이터 설정의 표준화된 메서드를 제공한다. 이를 통해 모듈 간 일관된 상호작용 방식을 보장하고 시스템 통합성을 높인다.

- DDS 통신 프레임워크: 장비 종속성과 무관하게 동작하는 데이터 분산 처리 프레임워크로, 실시간 데이터 전송의 신뢰성과 효율성을 보장한다. 모듈 간 통신 프로토콜을 표준화하여 새로운 장비 추가 시 통신 계층의 수정 부담을 최소화한다.

- 메시지 처리 파이프라인: 전투관리체계, 전시 소프트웨어, 연동 소프트웨어 간의 정형화된 메시지 흐름을 정의하는 파이프라인으로, 데이터 변환·전달·검증 과정을 표준화하여 시스템 간 데이터 호환성을 보장한다.

- 이벤트 기반 아키텍처: 상태 변화 알림을 위한 이벤트 처리 메커니즘으로, 관찰자 패턴을 기반으로 구현된다. 이를 통해 시스템 구성요소는 다른 요소의 상태 변화를 실시간으로 인지하고 필요한 조치를 취할 수 있다.

2.2 Variable Elements

- 장비 제어 로직: 각 IFF 장비 제조사별 상이한 제어 알고리즘과 상태 처리 방식을 포함한다. 예를 들어, 특정 장비는 질문기 설정을 위해 별도의 초기화 절차가 필요하거나, 다른 장비는 침묵 모드를 지원하지 않을 수 있다. 이러한 장비별 특성은 소프트웨어의 유연성을 저해하는 주요 요인이다.

- IFF 기능별 전시: IFF의 다양한 기능과 운용 모드에 따라 상이한 전시 처리 방식을 요구한다. Mode 1은 기본 식별 정보를, Mode 2는 고도 정보를, Mode 3A는 특정 식별 코드를, Mode 5는 복잡한 데이터 구조를 전시한다. 이러한 차이는 UI 모듈의 복잡성을 증가시킨다.

2.3 Application of the Feature Model

본 연구에서는 식별된 기능들을 피처 모델을 통해 체계적으로 분류하였다(Fig. 5). 공통 요소는 안정적이고 재사용 가능한 구조로, 가변 요소는 장비 및 기능 종속적인 특성을 가진다.

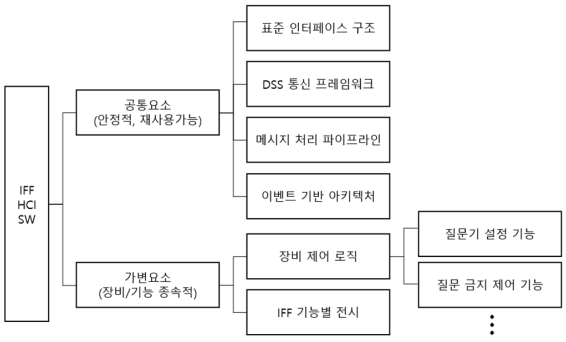

Fig. 5. Feature Model-based Classification

Step 3. Design

피아식별기 전시 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. 기존 구조의 안정적인 부분을 유지하면서 가변 요소를 효과적으로 분리하고 전략 패턴을 적용하여 개선하였다.

3.1 In-depth Analysis of Problems in the Existing Structure

기존 구조는 장비 종속적인 데이터 구조와 처리 로직을 직접 포함하고 있어 복잡도 및 결합도 문제를 가지고 있다 (Fig. 6). 예를 들어 특정 장비의 상태 데이터 구조가 변경되면 해당 관리자 클래스 전체를 수정해야 하며, 새로운 장비가 추가될 때마다 기존 코드를 복제·수정하는 방식으로 대응해야 했다. 이는 다음과 같은 문제를 유발한다.

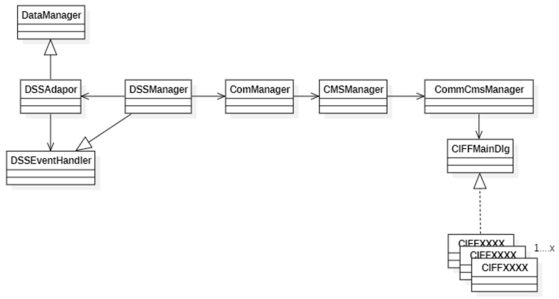

Fig. 6. Pre-Improvement Class Diagram of IFF HCI Interface SW

첫째, 결합도 문제로 인해 한 모듈의 변경이 다른 모듈에 연쇄적인 영향을 미친다.

둘째, 낮은 응집도로 인해 단일 모듈이 여러 책임을 동시에 수행하여 유지보수가 복잡해진다.

셋째, 확장성 부족으로 인해 새로운 요구사항에 대응하기 위한 개발 비용과 시간이 증가한다.

3.2 Improved Design

Fig. 7은 기존 구조의 한계를 보완한 개선된 아키텍처를 보여주며, 전략 인터페이스와 관리자·Maker 클래스 계층을 통해 장비 종속 기능을 분리하고 전략패턴과 SOLID 원칙을 적용하여 유연성과 확장성을 확보한 구조를 나타낸다.

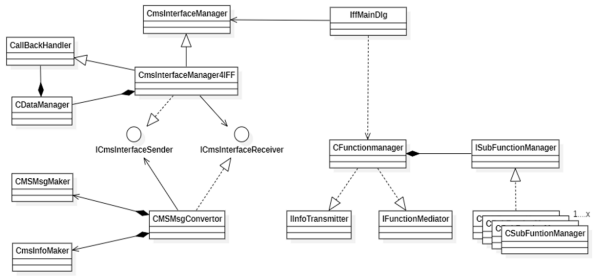

Fig. 7. Post-Improvement Class Diagram of IFF HCI Interface SW

3.2.1 Introduction of the Device Control Strategy Interface

장비 종속적인 기능을 분리하기 위해 장비 제어 전략 인터페이스를 도입하였다. 이 인터페이스는 장비 제어와 관련된 모든 공통 동작을 추상 메서드로 정의한다. 상태 처리, 제어 메시지 전송, 장애 처리 등 장비 제어의 핵심 기능들이 이 인터페이스를 통해 표준화된다.

3.2.2 Redefining Responsibilities of the Manager Class

관리자 클래스는 전략 패턴을 적용하여 장비 종속 기능을 전략 객체에 위임한다. 개선된 관리자 클래스는 표준 인터페이스 구현, 이벤트 처리, 데이터 흐름 제어 등 장비 독립적인 핵심 기능에만 집중한다. 이를 통해 단일 책임 원칙을 준수하고 모듈의 응집도를 높였다.

개선된 아키텍처에서는 ISubFunctionManager와 함께 CIFFInfoManager 등 Manager 클래스 계층을 세분화하여 구현하였다.

3.2.3 Systematic Implementation of Concrete Strategy Classes

각 장비 타입별로 구체적인 전략 클래스를 구현하여 해당 장비의 제어 알고리즘, 상태 처리 방식, 장애 처리 절차 등을 캡슐화하였다.

또한, ICmsInfoMaker와 CCmsInfoMaker 등 Maker 클래스 패턴을 적용하여 다양한 메시지 변환 전략을 구성하였다. 이러한 구조는 새로운 장비가 추가될 경우 기존 코드 수정 없이 새로운 전략 클래스만 추가하면 되므로 개방-폐쇄 원칙을 충족한다.

3.3 Operation of the Revised Architecture

개선된 아키텍처에서는 장비 제어 관리자가 런타임에 적절한 전략 객체를 선택하여 장비 제어 기능을 수행한다. 예를 들어, 시스템이 특정 장비 타입을 감지하면 해당 장비에 맞는 전략 객체를 동적으로 생성하고 관리자에 주입한다. 이후 모든 장비 제어 요청은 이 전략 객체를 통해 처리되므로, 장비 변경 시 해당 객체만 교체하면 된다.

3.4 Systematic Application of SOLID Principles

본 연구에서 제안하는 아키텍처는 다음과 같이 SOLID 원칙을 적용하였다.

- 단일 책임 원칙 (SRP): 각 전략 클래스는 특정 장비 제어에만 책임을 가지며, 관리자 클래스는 전략 조율과 시스템 통합에만 책임을 가진다.

개선된 아키텍처에서는 Manager 클래스가 기능 영역

을 전담하고 Maker 클래스가 메시지 변환을 전담하도록 구현하였다.

- 개방-폐쇄 원칙 (OCP): 개선된 아키텍처에서는 새로운 장비 추가 시 기존 코드 수정 없이 전략 클래스를 추가하여 확장 가능하도록 구현하였다. 예를 들어 새로운 IFF 모드 코드나 메시지 포맷이 추가될 경우 Maker 클래스만 추가하면 된다.

- 리스코프 치환 원칙 (LSP): 모든 전략 클래스는 동일한 인터페이스를 구현하므로 상호 교체가 가능하며 시스템 동작에 영향을 주지 않는다.

개선된 아키텍처에서는 모든 ISubFunctionManager 구현체가 인터페이스 계약을 준수하도록 하였다.

- 인터페이스 분리 원칙 (ISP): 세분화된 전략 인터페이스를 통해 불필요한 의존성을 제거하고 관련 기능만 그룹화한다.

개선된 아키텍처에서는 IInfoTransmitter와 IFunctionMediator 등 세분화된 인터페이스를 추가하였다.

- 의존성 역전 원칙 (DIP): 고수준 모듈은 구체 클래스가 아닌 추상 인터페이스에 의존하여 유연성과 테스트 용이성을 확보한다.

개선된 아키텍처에서는 CFunctionManager가 중앙 조정자로서 모든 서브시스템을 통제하고, 저수준 모듈이 아닌 인터페이스에 의존하도록 구현하여 컴포넌트 간 결합도를 최소화하고 유지보수성을 향상시켰다.

3.5 Expected Effects and Practical Benefits

본 연구에서 제안하는 아키텍처를 통해 다음과 같은 효과를 기대할 수 있다.

- 장비 교체 시: 해당 전략 클래스만 교체하여 최소한의 코드 변경으로 신속하게 대응 가능하며, 시스템 중단 시간을 최소화할 수 있다.

- 신뢰성 시험: 변경된 전략 클래스만 집중 테스트하여 시험 기간을 획기적으로 단축하고, 테스트 커버리지를 높일 수 있다.

- 재사용성: 장비 독립적인 관리자 클래스를 다른 프로젝트나 시스템에서 재사용할 수 있어 개발 효율성이 극대화된다.

- 확장성: 새로운 장비나 기술이 추가되더라도 전략 클래스만 추가하면 되므로 시스템의 수명 주기를 연장할 수 있다.

결론적으로, 본 연구에서 제안하는 표준 아키텍처는 기존의 장비 종속적 구조에서 벗어나 유연하고 확장 가능한 소프트웨어 구조를 구현하는 효과적인 방법론이다. 이는 함정 전투체계 표준 아키텍처(NSCP)에 즉시 적용 가능하며, 장비 및 기술 변경 시 UI 모듈 교체와 알고리즘의 전략적 변경을 용이하게 하여 방위산업 경쟁력 강화에 기여할 수 있을 것으로 기대된다.

IV. Evaluation

본 연구에서는 제안된 객체지향 표준 아키텍처가 함정 전투체계 IFF 전시 소프트웨어의 유지보수성을 개선하는지를 검증하기 위해 기존 아키텍처와 비교 분석을 수행하였다. 분석 기준은 코드 복잡도(Cyclomatic Complexity, MI(Maintainability Index), 코드 규모(LOC)로 설정하였다[13].

4.1 Subjects for Comparison

본 연구에서는 장비 변경 시 수정 범위를 정량적으로 평가하기 위해 개선 전 프로젝트와 본 논문에서 제안한 방법의 아키텍처를 적용하여 여러 모듈에 작업량이 분산된 구조를 가진 개선 후 프로젝트를 비교하였다.

4.2 Evaluation Methodology

각 평가 지표는 코드 복잡도 Cyclomatic Complexity(CC)와 Maintainability Index(MI), LOC를 활용하여 코드 복잡도를 산출하였다[14-15].

CC는 그래프 이론 기반의 지표로 프로그램 제어 흐름의 복잡성을 나타내며, 다음식으로 계산된다[16].

$$CC = E - N + 2P$$

(E: 간선 수, N: 노드 수, P: 연결 요소 수)

MI는 Haley의 공식에 기반하여 코드의 가독성과 유지보수 용이성을 측정한다[17].

$$MI = 171 - 5.2 \times log_2(LOC) - 0.23 \times CC - 16.8 \times log_2(operators)$$

LOC 측정: 공백 및 주석을 제외한 실제 코드 라인 수를 산출하여 코드 규모를 비교하였다.

4.3 Experimental Results of Comparative Analysis with the Conventional Architecture

4.3.1 Cyclomatic Complexity

Cyclomatic Complexity(CC)는 프로그램의 제어 흐름을 기반으로 복잡성을 정량화하는 지표로, 값이 높을수록 코드의 분기와 경로가 많아 유지보수가 어려워진다.

Table 2. Complexity Calculation

CC	Pre -Improvement	Post -Improvement
Total Complexity	1,835	3,167
Average Complexity	114.69	73.65
Maximum Complexity	669	528

Table 2에 제시된 결과와 같이, 개선 전 프로젝트의 총 복잡도는 1,835였으나 개선 후에는 3,167로 증가하였다. 이는 전체 모듈 수가 증가하면서 총합 지표가 커진 것으로, 단순히 총 복잡도만으로는 유지보수성 개선 여부를 판단하기 어렵다. 따라서 평균 복잡도와 최대 복잡도를 함께 고려할 필요가 있다.

평균 복잡도의 경우 개선 전에는 114.69였으나 개선 후에는 73.65로 약 35.8% 감소하여 모듈 단위의 복잡도가 전반적으로 낮아졌음을 보여준다. 또한 최대 복잡도 역시 669에서 528로 약 21% 감소하여 가장 복잡한 모듈의 제어 흐름이 단순화되었음을 실증하였다.

종합적으로 볼 때, 총 복잡도의 증가는 모듈 수 증가에 따른 자연스러운 현상이며, 평균 및 최대 복잡도의 감소는 제안된 표준 아키텍처가 모듈 단위의 복잡도를 완화하고 유지보수성을 개선하는 데 실질적으로 기여했음을 보여준다. 이는 모듈 분산화와 인터페이스 중심 설계가 복잡도 완화에 효과적임을 입증하는 결과라 할 수 있으며, 나아가 제안된 아키텍처가 코드 품질 지표 전반에서 개선 효과를 보였음을 뒷받침한다.

4.3.2 MI

MI(Maintainability Index)는 코드의 복잡도, 크기, 가독성을 종합적으로 반영하여 유지보수 용이성을 수치화하는 지표로, 값이 높을수록 유지보수성이 우수함을 의미한다[18].

Table 3. MI Calculation Results

MI Grade	Pre -Improvement	Post -Improvement
Good (20+)	64.1%	70.5%
Moderate (10-20)	0%	2.2%
Poor (0-10)	35.9%	27.3%

Table 3에 제시된 결과와 같이, 개선 전 MI 분포는 Good 64.1%, Moderate 0%, Poor 35.9%로 나타나 유지보수성이 양호한 모듈과 취약한 모듈이 극단적으로 분포하는 양상이 관찰되었다. 반면 개선 후에는 Good 70.5%, Moderate 2.2%, Poor 27.3%로 변화하였다. 이는 Good 등급 모듈 비율이 6.4%p 증가(상대적 약 10% 개선)하고, Poor 등급 모듈 비율이 8.6%p 감소(상대적 약 24% 개선)한 결과로, 유지보수성의 전반적 향상과 아키텍처 균형성 확보를 실증하였다.

이러한 변화는 단순히 평균 지표의 개선을 넘어, 극단적 분포 완화와 위험 분산 효과를 동시에 확보한 결과라 할 수 있다. 즉, 개선 전에는 유지보수성이 높은 모듈과 매우 낮은 모듈이 양극화되어 있었으나, 개선 후에는 중간 수준 모듈이 등장함으로써 전체 시스템의 안정성이 강화되었다.

이는 개발자가 장비 교체나 UI 변경 시 특정 모듈에 과도한 부담이 집중되는 문제 발생 가능성을 감소시켜, 유지보수 작업을 보다 균형 있게 분산할 수 있음을 의미한다. 따라서 MI 개선은 단순히 수치상의 향상에 그치지 않고, 개발 비용 절감, 시험 기간 단축, 시스템 확장성 확보라는 실질적 장점으로 이어진다.

특히 수출 사업 환경과 같이 다양한 장비와 UI 변경이 반복되는 상황에서, 이러한 개선은 유지보수성 향상뿐 아니라 경제성과 신뢰성 확보에도 기여하며, 제안된 표준 아키텍처의 학문적·실무적 기여를 뒷받침한다.

4.3.3 Comparison of Code Size

LOC(Line of Code)는 소프트웨어 규모와 유지보수성을 직관적으로 나타내는 지표로, 코드 수정 범위와 개발 비용을 추정하는 데 중요한 역할을 한다. 특히 지나치게 큰 모듈은 유지보수성 저하의 주요 원인이 되므로, LOC 분포 분석을 통해 모듈 크기의 균형성과 안정성을 평가할 수 있다. 따라서 LOC는 Cyclomatic Complexity 및 Maintainability Index와 함께 유지보수성 평가의 핵심 지표로 활용된다[19].

Table 4. LOC Comparison Analysis

LOC	Pre -Improvement	Post -Improvement
Min LOC	167	0
Max LOC	5,122	2,484
Standard Deviation	1,112.4	414.7
First Quartile	511.5	97
Third Quartile	1,808.2	576
Inter -quartile Range	1,296.8	479

LOC 비교 분석 결과, 개선 전 아키텍처에서는 최대 5,122 LOC에 달하는 대형 모듈이 존재하고 표준편차가 1,112.4로 매우 높아 모듈 크기 분포가 불균형하게 나타났다. 반면 개선 후 아키텍처에서는 최대 LOC가 2,484로 절반 이하로 감소하고, 표준편차가 414.7로 줄어들어 모듈 크기의 균형성이 크게 향상되었다. 또한 사분위수 범위(IQR)가 1,296.8에서 479로 축소되어, 전체 모듈의 크기 분포가 안정화되었음을 확인하였다. 이는 제안된 표준 아키텍처가 모듈 단순화와 균형화에 실질적으로 기여했음을 보여준다.

4.3.4 Software Quality Comparison Based on Standard Metrics

Table 5. Summary of Architecture Improvement Effects.

EVALUATION METRIC	KEY MEASUREMENT	CORE CONTRIBUTION
Code Complexity	Avg. complexity ↓ 35.8%	Increases developer productivity & reduces errors.
Maintainability (MI)	Poor grade modules ↓ 24%	Lowers maintenance risk & balances workload.
Structural Stability (Coupling/ Cohesion)	Max. module size ↓ 51%	Improves reliability by eliminating oversized modules.
	Std. deviation ↓ 63%	Enhances system stability & predictability.

앞선 분석 결과를 종합하여 제안된 아키텍처의 품질 개선 효과를 Table 5 와 같이 요약할 수 있다. 본 연구에서는 개선전, 후의 소프트웨어 품질 특성을 정량적으로 비교 평가하기 위해 IEEE/ISO에서 표준으로 정의한 대표적인 메트릭을 활용하였다. 분석에는 코드의 복잡도, 유지보수성, 결합도, LOC 지표가 포함되었으며, 각 지표의 수치를 통해 두 시스템의 구조적 특징과 품질 수준을 객관적으로 파악하고자 하였다.

4.3.4.1 Improved Development Efficiency through Reduced Complexity

Cyclomatic Complexity는 코드 내 의사 결정점의 수를 측정하여 정량적 복잡도를 나타내는 지표이다. 본 연구 결과, 제안된 아키텍처는 모듈의 평균 복잡도를 35.8% 감소시키고 최대 복잡도 역시 21% 개선했다. 이는 전략 패턴과 인터페이스 중심 설계를 통해 복잡한 조건문을 대체하고 모듈의 단일 책임을 명확히 함으로써 달성된 결과이다. 따라서 개발자는 더 단순한 제어 흐름을 가진 코드를 다루게 되므로, 신규 기능 추가나 버그 수정 시 예측 가능성이 높아지고 실수를 줄일 수 있어 전반적인 개발 효율성과 코드 품질이 직접적으로 향상됨을 입증하였다.

4.3.4.2 Risk Mitigation and Enhanced Stability via Improved Maintainability

Maintainability Index(MI)는 유지보수성의 양적 측면뿐만 아니라 질적 측면까지 평가하는 중요한 지표이다. 본 연구에서는 'Good' 등급 모듈의 증가와 'Poor' 등급 모듈의 감소라는 분포의 개선을 통해 질적인 향상을 실증했다. 이는 시스템 내에 존재하던 유지보수하기 까다로운 모듈들이 사라지고 전체적으로 균형 잡힌 구조로 변했음을 의미한다. 이에 따라 장비 변경이나 기능 수정과 같은 유지보수 작업 시, 특정 모듈에 부담이 몰리는 현상이 방지되어 개발 리소스를 효율적으로 분배할 수 있으며, 시스템 전체의 안정성을 장기적으로 보장할 수 있다.

4.3.4.3 Achieving Extensibility through Reduced Coupling and Increased Cohesion

본 연구에서는 LCOM과 같은 직접적인 결합도·응집도 지표를 별도로 측정하지는 않았으나, 코드 규모(LOC) 분석 결과는 구조 개선의 핵심인 결합도 감소와 응집도 증대를 간접적으로나마 명확히 시사한다. 거대 모듈(5,122 LOC)의 소멸과 표준편차의 급격한 감소는 모듈들이 잘 정의된 단일 책임을 수행하게 되었음(높은 응집도)과, 모듈 간 상호작용이 표준화된 인터페이스로 제한되었음(낮은 결합도)을 강력히 뒷받침한다. 이러한 구조적 안정성은 새로운 장비나 기능 추가 시 기존 코드를 수정하지 않고 확장할 수 있는 유연한 기반을 제공하며, 이는 방위산업과 같이 변화가 잦은 환경에서 시스템의 수명 주기를 연장하는 데 결정적인 역할을 수행한다.

4.4 Summary of Results

본 연구에서는 제안된 표준 아키텍처의 효과를 종합적으로 검증하기 위해 CC, MI, LOC를 산출하여 비교하였다. 그 결과, 제안된 아키텍처는 코드 수정 범위를 줄이고 유지보수성을 향상시켜 실질적인 개선 효과를 보여주었다. 또한 장비 교체나 기술 변경·추가 시 발생하는 개발 비용을 절감할 수 있음을 확인하였다. 나아가 전투체계 소프트웨어의 확장성과 경제성을 동시에 확보할 수 있는 실질적인 방안임을 시사한다.

4.5 Discussion and Conclusion

비교 분석 결과, 제안된 객체지향 표준 아키텍처는 함정 전투체계 IFF 전시 소프트웨어의 유지보수성을 다각적으로 개선하였다. Cyclomatic Complexity(CC)는 평균 35.8% 감소하여 코드 제어 흐름의 단순성이 확보되었고, Maintainability Index(MI)는 Good 등급 모듈 비율이 6.4%p 증가하고 Poor 등급 모듈 비율이 8.6%p 감소하여 유지보수성의 균형성이 향상되었다. 또한 LOC 분석에서는 최대 LOC가 절반 이하로 줄고 표준편차 및 사분위수 범위가 크게 축소되어 모듈 크기의 균형성과 안정성이 개선되었음을 확인하였다.

이러한 결과는 제안된 아키텍처가 장비 교체 및 기능 변경 시 코드 수정 범위를 줄이고 유지보수 시험 부담을 완화하는 데 실질적으로 기여함을 보여준다. 특히 모듈 분산화와 인터페이스 중심 설계는 전투체계 소프트웨어의 확장성과 재사용성을 높여, 다양한 장비 환경과 수출 사업에서 발생하는 반복적 수정 부담을 효과적으로 완화할 수 있다.

따라서 본 연구는 단순한 코드 품질 지표 개선을 넘어, 전투체계 소프트웨어의 경제성과 확장성을 동시에 확보할 수 있는 실질적 방안을 제시한다는 점에서 학문적·실무적 의의가 크다. 다만 본 연구는 특정 프로젝트와 가상 시나리오 기반으로 평가가 이루어진 한계를 지니므로, 향후 연구에서는 다양한 모듈과 실제 운용 데이터를 활용한 장기적 검증이 필요하다.

V. Conclusions

본 연구는 IFF 전시 소프트웨어의 유지보수성 향상을 위해 객체지향 표준 아키텍처를 제안하고, 기존 구조와의 비교 평가를 통해 그 효과를 검증하였다. 평균 CC는 35.8% 감소, MI는 Good 모듈 비율이 6.4%p 증가하고 Poor 모듈 비율이 8.6%p 감소, 최대 LOC는 절반 이하로 줄어드는 등 정량적 개선 효과가 확인되었다. 이는 모듈 분산화와 인터페이스 중심 설계가 유지보수 효율성과 확장성 제고에 실질적으로 기여함을 보여준다.

특히 수출 사업 환경에서 빈번히 발생하는 장비 교체와 UI 변경 요구에 대응할 수 있는 범용 아키텍처를 제시했다는 점에서 실무적 의의가 크다. 동시에 국내 사업에서는 장기화되는 신뢰성 시험 부담을 완화할 수 있다는 점에서도 의미가 있다.

본 연구의 평가는 특정 IFF 시스템에 국한되어 있어 그 결과를 전투체계 전체로 일반화하는 데에는 신중한 접근이 필요하다. 다만, 본 연구의 핵심 기여는 특정 사례의 성공 그 자체가 아니라 장비 종속성 문제를 해결하기 위한 설계 방법론에 있다. 이러한 설계 원리는 현대 전투체계 HCI에서 공통적으로 나타나는 실시간 데이터 통신, 메시지 변환, 이벤트 처리 등의 문제에 대해 범용적으로 적용될 수 있는 잠재력을 가진다.

향후 연구에서는 제안된 아키텍처를 본 연구의 대표 모델 IFF HCI와 유사한 구조를 가진 다양한 센서/무장 연동 전투체계 HCI의 모듈로 확장 적용하고, 실제 수출 프로젝트와 국내 운용 데이터를 활용한 장기적 검증을 통해 연구 결과의 일반성과 실효성을 강화할 예정이다.

REFERENCES

[1] Dong-Uk Kim, Hyung-Rok Jung, Young-Il Song, "A Study on Cost Behavior of Korean Defense Industry," Korean Journal of Management Accounting Research, Vol. 11, No. 2, pp. 55-82, 2011.

[2] Dong-Uk Kim, Hyung-Rok Jung, "A Study on Cost Structure of Korean Defense Firms," The Quarterly Journal of Defense Policy Studies, Vol. 28, No. 3, pp. 109-143, 2012.

[3] Ki-Chang Kim, Yong-Hwan Kim, Ju-Hwan Shin, Ki-Jun Han, "A Case study on Application for Software Reliability Model to Improve Reliability of the Weapon System" Journal of KISS: Software and Application, Vol. 38, No. 8, pp. 405-418, Aug. 2011.

[4] Seung-Young Lee, Dong-Hwan Kim, "Policy Trends of Defense Software Quality Improvement" Communications of KIISE, Vol 38, No.9, pp. 23-35, Jan 2020.

[5] 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, Nov. 2020. DOI: 10.9708/JKSCI.2020.25.11.147.

[6] Ki-Tae Kwon, Ki-Pyo Kim, Hwan-Jun Choi, "Design of the Scalable Naval Combat System Software using Abstraction and Design Pattern" Journal of the Korea Society of Computer and Information, Vol. 24, No. 7, pp. 101-108, Jul. 2019. DOI: 10.9708/JKSCI.2019.24.07.101.

[7] Alan Dix, Janet Finlay, Gregory Abowd, Russell Beale, Human-Computer Interaction, Pearson Education, 3rd ed., 2004.

[8] 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

[9] C. S. Back, and J. H. 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, Jan. 2021. DOI: 10.9708/JKSCI.2021.26.01.147

[10] Galleon Embedded Computing, "NGVA Data Model and OpenDDS," https://galleonec.com/ngva-data-model-and-opendds/

[11] R. C. Martin, "Agile Software Development, Principles, Patterns, and Practices," Prentice Hall, pp. 95-145, 2002.

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

[13] Yeun, K., Park, S., & Kim, Y. (2025). A Study on Efficient Design of Hull-Mounted Sonar Interface Control Unit in Naval Combat Management System. Journal of the Korea Society of Computer and Information , 30(8), 189-197. 10.9708/jksci.2025.30.08.189

[14] A. Tahir, S. Counsell, and S. G. MacDonell, "An empirical study into the relationship between class features and test smells," Proceedings of the 23rd Asia-Pacific Software Engineering Conference (APSEC), pp. 137-144, Hamilton, New Zealand, Dec. 2016. DOI: 10.1109/APSEC.2016.029.

[15] I. Yanakiev, B.-M. Lazar, and A. Capiluppi, "Applying SOLID principles for the refactoring of legacy code: An experience report," Journal of Systems and Software, Vol. 220, Art. no. 112254, Feb. 2025. DOI: 10.1016/j.jss.2024.112254.

[16] Thomas J. McCabe, "A Complexity Measure," IEEE Transactions on Software Engineering, Vol. SE-2, No. 4, pp. 308-320, 1976.

[17] Paul Oman, Jack Hagemeister, "Metrics for Assessing a Software System's Maintainability," Proceedings of the Conference on Software Maintenance (ICSM), pp. 337-344, 1992.

[18] Tjaša Heričko, Boštjan Šumak, "Exploring Maintainability Index Variants for Software Maintainability Measurement in Object-Oriented Systems," Applied Sciences, Vol. 13, No. 5, Article 2972, pp. 1-20, 2023.

[19] Adrian S. Barb, Colin J. Neill, Raghvinder S. Sangwan, Michael J. Piovoso, "A statistical study of the relevance of lines of code measures in software projects," Innovations in Systems and Software Engineering, Vol. 10, No. 4, pp. 243-260, 2014.

Authors

Sung-Uk Jo received the B.S. degree in Computer Science from Kyungpook National University, Korea, in 2017. He has been working at Hanwha Systems Co., Korea, since 2017, where he is involved in the development of naval combat software. His research interests include naval combat, export-oriented software in naval combat systems, and related defense technologies. Sung-Uk Jo work includes Management Systems, Interface Control Unit Software, Software Reuse, MFC Development, and related areas.

Su-Seok Park received the B.S degrees in Computer Science and Engineering from Kumoh National Institute of Technology, Korea. He is currently working in Hanwha Systems Co. from 2009.

He is interested in Naval Combat Management System Software, Software Engineering.

Young-San Kim received B.S degree in Computer Science and Electrical Engineering from Handong University, Pohang, Korea. He is currently working in Hanwha Systems Co. from 2009.

He is interest in Naval Combat Management System Software, Human Computer Interface and System Engineering.