The Journal of The Institute of Internet, Broadcasting and Communication (IIBC)
Vol. 17, No. 4, pp.1-10, Aug. 31, 2017. pISSN 2289-0238, eISSN 2289-0246

https://doi.org/10.7236/JIIBC.2017.17.4.1
JIIBC 2017-4-1

REST 아키텍처를 이용한 함정 통합 서비스 프레임워크

Integrated Service Framework for Naval Ship using REST Architecture

김규백*

Kyubaek Kim*

요 약 미래의 해양 전장 환경에 대비하고 다양한 요구사항을 만족하기 위해 함정을 구성하는 여러 체계의 기능이 고도화되고 다양화되고 있다. 함정의 승조원 수는 제한되어 있어 결국 한명의 승조원이 담당하는 임무와 운용 화면의 수는 늘어나고 있다. 본 논문에서는 REST 아키텍처를 이용하여 함정 내 각 체계와의 연동을 통해 체계의 정보와 서비스를 자함 객체로 통합하고 HTTP와 JSON 기반의 API 형태로 제공하는 함정 통합 서비스 프레임워크를 제안한다. 그리고 이를 전투체계 테스트베드에서 구현하고 성능을 측정하여 해군함정에 적용 가능함을 입증하였다. 본 논문에서 제시된 프레임워크를 실제 함정에 도입한다면, 체계 간의 기능이 통합된 화면의 설계가 가능하게 되어 승조원이 효율적으로 함정을 운용할 수 있을 것이다.

Abstract The functions of various systems inside the naval ships have been improved and diversified to prepare for the future battlefield and satisfy new requirements. The number of missions and operating displays with which a crew has to deal has increased. In this paper, Integrated Service Framework for Naval Ships is proposed. The proposed framework is based on REST architecture and it interfaces the networks of each systems in the ship. The connected data and functions are rebuilt as an object and they are provided as HTTP and JSON based APIs. The prototype of the framework was implemented in a combat system test-bed and its feasibility was proved with performance tests. With the proposed framework, operational efficiency for a naval ship can be achieved by designing new integrated displays and HCI which utilize whole ship's capabilities.

Key Words : Naval Combat Systems, System Integration, REST, OpenAPI

Ⅰ. 서 론

우리나라 영해를 수호하는 해군 함정은 외부로 보이는 선박플랫폼 외에도 그 내부에 수많은 체계와 장비로 구성되며 해군 승조원들은 이를 제어함으로서 임무를 수행해오고 있다. 90년대 이후의 급속한 정보통신기술의 발달은 해군 함정에도 영향을 주어 각종 장비의 전자화로 나타났고 네트워크 기술에 힘입어 각 주요기능을 중심으로 몇 개의 체계로 통합되었다. 함정 내 발전, 배전 및 추진 관련 기능 및 장비는 통합전기추진체계(IPS, Integrated Power System)[1] 또는 통합플랫폼관리체계(IPMS, Integrated Platform Management System)로, 항해 센서, 항해레이더, 전자해도 등 함정의 항해관련 기능과 장비는 통합항해체계(INS, Integrated Navigation System)[2]로 그리고 탐색/추적 레이더, 소나 등 표적을 탐지하는 센서와의 연동, 함포, 유도탄 등의 무장 및 이들

*정회원, 국방과학연구소 연구원
접수일자: 2017년 6월 9일, 수정완료: 2017년 7월 9일
게재확정일자: 2017년 8월 11일

Received: 9 June, 2017 / Revised: 7 July, 2017
Accepted: 11 August, 2017
*Corresponding Author: kyubaek@add.re.kr
Dept. Agency for Defense Development, Korea

을 통제하는 전투정보실 내의 정비와 콘솔은 함정전투체계(Naval Combat System)[3][4]로 통합되어 해군 함정을 움직이는 각각의 단일 체계로 발전하고있다. 이러한 체계들의 발전은 전자화, 자동화 통해 해군함정에 비약적인 성능향상을 이루었다.

최근 동북아에서의 해군력 급증과 미래 전장 환경에 대비하기 위한 새로운 요구사항은 신규 함정의 건조뿐만이 아니라 기존 함정들에게도 더 많은 임무의 수행을 요구하고 있다. 그러나 해군 승조원의 수는 제한되어 있다. 결국 함정 당 승조원 수는 감소하며 한명의 승조원이 점점 더 많은 임무를 수행해야하는 상황이다[5]. 더 많은 임무는 더 많은 운용 화면을 담당하게 되는 것을 의미하며, 제한된 수의 콘솔로 이를 운용할 경우 승조원은 여러 개의 화면을 지속적으로 전환하며 임무를 수행해야 하며 이는 임무에 집중도 하락으로 이어질 수 있다.

또 다른 문제로 해군 함정의 긴 수명주기와 그 기간 중에 발생하는 운용 화면에 대한 요구사항을 함정에 신속하게 반영하는 것이 불가능하다. 해군 함정은 한번 취역하면 30년 가까이 운용된다. 긴 함정의 수명주기 중 운용화면에 대한 변경이나 기능추가 같은 요구사항은 언제든 발생할 수 있음에도 이를 반영하기 위해선 각 체계 개발 주체의 재참여가 필수적이며 참여하지 않는 경우 해당 체계를 새로이 제작해야 한다. 또한 하나의 체계 외의 기능에 대해서는 같은 함정 내에 정보가 있음에도 불구하고 개발시점에서 체계 간 연동이 고려되지 않았다면 이후에 한 화면에 동시 전시는 불가능하다. 또한 최근 모바일 기기와 사물인터넷(IoT, Internet of Things)의 등장과 발달로 해군 함정에도 이를 도입하기 위한 연구[6]가 시작되고 있지만 이미 개발이 완료된 체계에 대해서는 체계의 수정 없이는 새로운 기술 및 장비의 도입이 불가능하다.

본 논문에서는 이러한 문제를 해결하기 위해서 함정 내의 각 체계서비스와 화면을 분리하는 함정 통합 서비스 프레임워크를 제시한다. 각 체계의 서비스를 통합하여 API(Application Program Interface) 형태로 모든 기능을 제공하여 운용 화면 및 단말이 독립적으로 설계될 수 있는 방안을 제시한다. 이를 위해 최근 인터넷 업체에서 자사 서비스를 Open API형태로 제공할 때 널리 쓰이는 REST(Representational state transfer) 아키텍처[7]를 해군 함정에 도입하여 각 단일체계 기능에 접근 가능한 일원화 된 표준 프로토콜 기반 API를 설계하고자 한다.

본 논문의 내용은 다음과 같이 구성된다. 2장에서는 유사한 문제를 해결하기 위한 미 해군의 사례 및 관련 기술을 설명하고 3장에서 함정 내 각 체계를 통합하고 API를 설계하는 방안을 기술한다. 4장에서는 테스트베드 구현을 통해 제시된 프레임워크의 실현가능성을 보이며, 마지막 5장에서 결론을 맺는다.

II. 관련 연구

해군 무기체계의 획득 및 운용비용의 절감, 보다 효율적인 함정의 운용을 위해 미 해군은 1990년대부터 개방형 아키텍처를 도입하여 상용 제품 및 기술을 무기체계에 빠르게 적용해 나가고 있다. 또한 개방형 아키텍처를 더욱 발전시켜 현재는 함정 내의 모든 컴퓨팅 자원을 통합하여 함정 내의 다양한 전투/비전투 관련 체계를 하나로 통합하여 운용할 수 있는 통합 함정 컴퓨팅 환경(TSCE, Total Ship Computing Environment)이라는 개념을 정립하였으며 최근 TSCE 개념이 적용된 DDG-1000 줌월트급 구축함이 취역하였다[8].

TSCE의 기반 하드웨어 구성은 그림 1과 같다. TSCE는 임무영역의 운용자 화면이 있는 전시처리계층, 대용량 정보를 처리하는 코어계층, 외부 장비들과 연동을 위한 연동처리계층으로 구성되며 다양한 인터페이스를 통해 연결된 외부장비를 적응계층을 통해 IP 네트워크로 통합하고 이를 코어계층의 상용 블레이드 서버를 통해 처리하여 표현계층에서 화면으로 전시하고 있다. 현재의 TSCE는 기존 장비의 SW를 그대로 재활용하기 위해 코어계층의 서버 풀을 가상머신 호스트로 구성하고 기존의 각 체계 컴퓨팅 환경을 그대로 가상머신으로 포팅 하여 동작시키고 가상머신의 화면을 표현계층의 장비를 통해

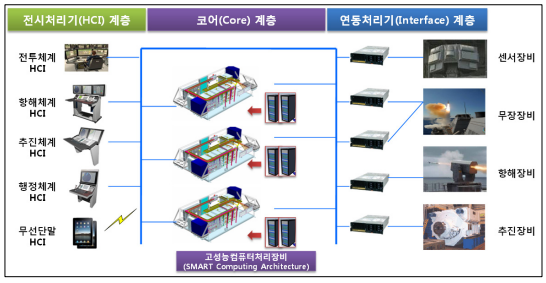

그림 1. TSCE 개요
Fig. 1. TSCE Overview

원격에서 접속하여 전시한다. 이를 통해 TSCE에서는 운용자 화면의 공간적 제약이 사라지며 줌월트급은 기존의 전투정보실, 함 조종실, 사령소 등의 역할을 SMC(Ship's Mission Center) 한 곳으로 모아 300명 이상의 승조원이 필요한 기존의 이지스(Aegis) 구축함보다 훨씬 적은 140명의 인원으로 함정 운용을 가능케 하였다.

TSCE는 가상화와 원격접속 기술을 기반으로 운용 화면의 공간적 통합은 이루었지만 각 체계의 기존 운용 화면을 그대로 사용하고 있기 때문에 승조원 한명 당 운용 화면의 수는 증가했으며, 증가한 운용 화면을 전시하기 위해 한 대의 운용자 콘솔은 3개의 모니터로 운용화면을 전시한다. 따라서 승조원의 임무집중도 향상과 보다 더 효율적인 함정 운용을 위해선 함정 내 각 체계간의 기능 및 화면의 통합이 필수적이며 고정된 수의 운용자 콘솔에서 탈피하여 담당 임무의 성격에 따라 그 수와 형태를 유연하게 가져갈 수 있는 구조를 필요로 한다.

REST는 일종의 소프트웨어 아키텍처로 시스템이 제공하는 기능을 자원으로 정의하고 자원에 대한 접근방법으로 URI 주소를 지정하는 방법을 제시한다. 전송 프로토콜로 HTTP 표준을 그대로 사용하며 이를 통해 자원에 접근하고 조작을 지원한다[7]. 따라서 REST 아키텍처는 빠르고 간결하며 웹서버와 같이 쉽게 확장이 가능하다. 또한 일관된 인터페이스는 서버와 클라이언트가 각각 독립적으로 개발될 수 있도록 한다. 이러한 특징으로 기존에 쓰이던 SOAP(Simple Object Access Protocol)을 빠르게 대체하고[9] 있으며 인터넷상의 많은 정부사이트, 포털사이트, SNS업체 등에서는 REST 형식의 OpenAPI를 제공하여 외부 개발자로 하여금 다양한 응용 프로그램을 개발할 수 있게 한다. 뿐만 아니라 기존의 서비스 인프라가 구축되어 있는 곳에서도 REST 아키텍처를 도입하여 REST와 OpenAPI의 장점을 활용하려는 다양한 시도가 이루어지고 있다[10][11].

Ⅲ. 함정 통합 서비스 프레임워크

본 논문에서 제안하는 함정 통합 서비스 프레임워크는 해군 함정 내의 여러 체계들의 서비스를 네트워크로 연동하고 이를 REST 아키텍처 디자인 가이드[12]를 적용하여 자원(Resource)화하여, 클라이언트가 단일 서비스 게이트웨이를 통해 쉽게 자원에 접근할 수 있도록 설계되었다. 그림 2는 함정 통합 서비스 프레임워크의 구성을 나타낸다.

1. 체계별 인터페이스 연동

함정의 여러 체계의 기능을 통합하여 단일 API 서비스로 제공하기 위해서 가장 우선적으로 해결되어야 할 과제는 서비스 게이트웨이와 각 체계간의 연동이다. 각

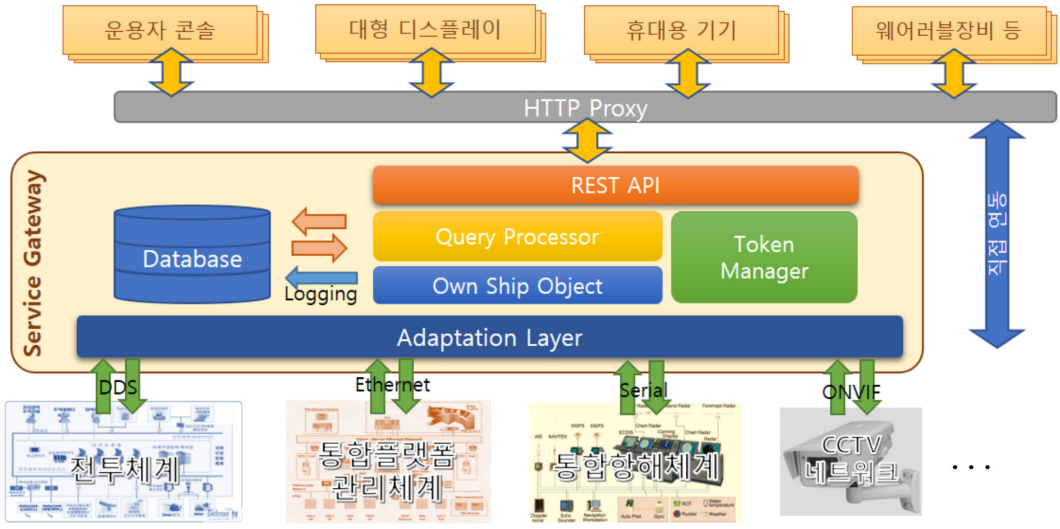

그림 2. 함정 통합 서비스 프레임워크 구조
Fig. 2. Architecture of Integrated Service Framework for Naval Ship

체계로부터 생산된 정보를 하나의 서비스 게이트웨이로 모으기 위해서 서비스 게이트웨이는 각 체계 네트워크의 공통 노드로써 존재해야 한다. 이를 위해 서비스 게이트웨이에는 각 체계별 네트워크 노드 역할을 수행하며 해당 체계를 상위 레이어에 추상화하여 제공하는 어댑터가 필요하다. 각 체계별 어댑터는 해당 체계의 물리/네트워크 계층을 담당하는 하드웨어 장비(NICs, CAN, 시리얼 컨트롤러 등)를 통해 각 체계와 물리적 연동을 하며 각 체계 내에서 생산된 메시지를 수신한다.

2000년대 이후의 국내에서 개발된 함정전투체계는 이더넷을 기반의 DDS(Data Distribution Service) 통신 미들웨어를 이용해 노드간의 통신을 수행한다[13]. 따라서 서비스 게이트웨이의 전투체계 어댑터 역시 전투체계 네트워크에 이더넷으로 연결되며 하나의 DDS 노드로 동작한다. 실제 전투체계 네트워크에서 각 노드는 자신의 기능과 관련된 토픽만 구독한다. 하지만 전투체계 어댑터는 서비스 게이트웨이를 통해 전투체계 대부분의 서비스를 제공할 수 있어야 하므로 가능한 모든 토픽을 구독하고 수신된 메시지를 파싱하여 상위 레이어로 전달한다. 반대로 상위 레이어에서 수신한 전투체계로의 제어 명령은 어댑터가 해당 토픽을 생성해 발행한다.

통합항해체계(INS)는 시리얼 기반의 NMEA 0183, 이더넷 기반의 IEC 61162-450 등의 표준 프로토콜을 중심으로 체계 네트워크가 구성되며 통합플랫폼관리체계(IPMS)는 제조사의 이더넷 기반 독자 프로토콜을 이용해 구성된다. 이들 체계를 위한 어댑터도 전투체계 어댑터와 마찬가지로 시리얼(RS-232/422), 이더넷을 통해 각 체계 네트워크와 연결되며 프로토콜 문서 및 연동통제문서를 참고하여 송수신 메시지를 해석하여 각 체계의 정보를 상위 레이어와 교환한다.

CCTV 네트워크는 통합항해체계나 통합플랫폼관리체계에 포함된 경우도 있지만 보안용이나 특정 장비 감시 등 필요에 따라 별도의 CCTV 네트워크가 함 내외에 구축될 수 있다. CCTV는 ONVIF(Open Network Video Interface Forum) 표준 프로토콜이 널리 사용되므로 서비스 게이트웨이에는 ONVIF 프로토콜과의 연동도 고려되어야 한다. ONVIF 프로토콜은 크게 CCTV 영상 전달용 프로토콜과 CCTV를 제어하는 프로토콜로 구성되어 있다. CCTV 제어용 프로토콜은 SOAP으로 구현되어 있고 CCTV의 영상 소스를 요청하면 영상의 스트리밍 URL(Uniform Resource Locator)을 반환한다. 그리고 각 CCTV는 해당 URL를 통해 영상을 RTP/RTSP 프로토콜로 제공한다. 즉 CCTV 네트워크는 이미 HTTP 프로토콜을 기반으로 설계되어 있어 별도의 어댑터 없이 서비스 게이트웨이 앞 단에 HTTP Proxy를 두어 CCTV 네트워크 액세스가 가능하다. 하지만 CCTV 제어를 다른 체계의 기능제어와 같은 형태로 제공하기 위해 어댑터가 SOAP 클라이언트가 되어 SOAP 메시지를 Unmarshall 하여 CCTV 상태정보를 상위 레이어로 제공하고 상위 레이어에서 수신한 제어명령은 SOAP 메시지로 Marshall하여 해당 CCTV로 전송하는 것도 가능하다.

2. 자함 객체(Own Ship Object)

함정 내 다양한 장비에서 생산되고 어댑터를 통해 수집된 수많은 정보를 클라이언트로 제공하기 위해서는 모든 정보가 REST 자원의 형태로 정의되어야 한다. 각각의 체계는 센서, 신호처리장치 등의 실제 하드웨어 장비로 구성이 되어 있으며, 각 장비는 장비 자체에 대한 속성과 상태정보를 가지고 있다. 따라서 체계 하드웨어 계층구조를 기반으로 REST 자원의 계층구조를 모델링한다. 최상위 개체는 자함(ship)이 되며 자함은 전투체계(cs), 통합항해체계(ins), 통합플랫폼관리체계(ipms) 등의 체계를 하위 개체로 가진다. 다음 레벨의 개체는 각 체계의 주요 구성 장비로 이루어지며 콘솔, 캐비닛, 각종 센서류, CCTV 등이 포함될 수 있다. 콘솔, 캐비닛, CCTV 등과 같이 복수 개의 장비가 존재하는 경우, 하나의 공통 개체를 정의하고 하위에 개별 장비를 두고 ID를 할당해 개별 장비를 구분한다.

각 체계에는 구성장비의 상태정보과 기능 외에도 정보 자체가 자원인 경우가 존재한다. 예를 들어 자함 운동정보는 현재 위치의 위경도 좌표, 침로, 헤딩, 속력, 함의 롤/피치 등으로 구성되며 이는 다양한 센서로 부터 획득한 정보를 종합한 결과이다. 전투체계의 표적정보 역시 다중 센서를 통해 획득한 접촉정보를 종합하여 생산된 정보이다. 또한 항해체계의 항해계획, 전투체계의 전술구역, 교전계획 등의 정보는 이들을 저장하고 있는 장비의 속성정보 보다는 정보 그 자체가 의미를 가지고 기능을 제공한다. 이러한 정보는 독립된 자원으로 정의하여 관리한다. 이들 정보를 생산한 센서나 저장된 위치는 속성정보에 출처(Source)로 표현하여 관리한다.

각 개체의 명칭은 URI(Uniform Resource Indicator)로 표현하기에 적합하고 해군의 다양한 함정에 일관된

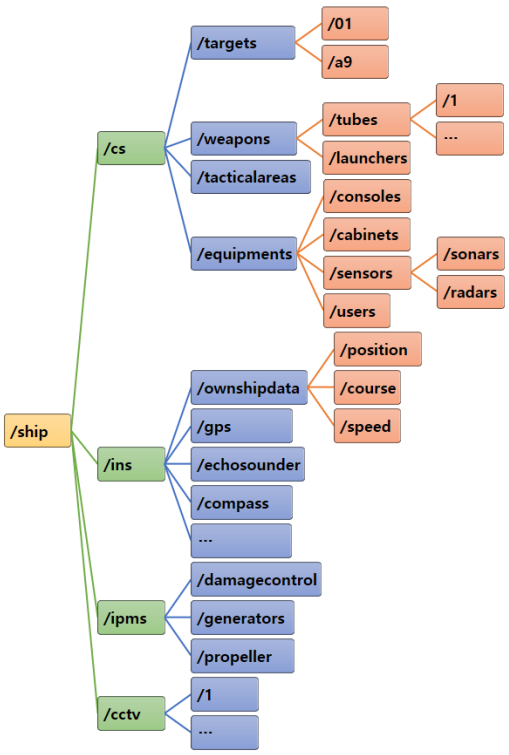

그림 3. 자함 객체 계층 구조
Fig. 3. Hierarchical Structure of Own Ship Object

자원 모델로서 적용하기 위해 몇 가지 규칙이 필요하다. 각 개체는 영문자로 표현되어야 하며 가능하면 소문자만 구성하고 공백문자는 사용할 수 없다. URI 문법 표준인 RFC 3986에서 URI의 Scheme과 Authority 부분을 제외한 나머지 부분에 대해서는 대소문자를 구분하여 사용할 수 있도록(Case Sensitive) 규정하고 있어 대소문자를 혼용하여 자원의 이름을 정의하면 대소문자만 다른 같은 명칭이 존재하게 되어 혼동을 발생시킬 위험이 있다. 또한 자원의 이름은 제조사명이나 장비 모델명이 아닌 일반 명칭으로 정의하고 해당 장비의 구체적인 정보는 해당 자원의 속성으로 기술토록 한다. 한 개체가 여러 개의 하위 개체를 포함(collection)한다면 해당 개체의 이름은 복수형(-s)으로 정의하여 다른 정보를 포함하는 개체임을 알 수 있도록 한다. 그리고 모든 자원은 해당 자원까지의 계층구조를 나열해 URI 형태를 이루게 한다. 자함이 탐지한 01번 표적은 (1)처럼 표현한다.

/ship/cs/targets/01 (1)

이와 같이 자원의 계층구조와 명칭을 정의하고 각 자원이 가지는 세부정보, 기능은 자원의 속성으로 표현하여 함정 내의 각 체계별 정보를 하나의 자함 객체로 통합한다. 각 체계별 어댑터를 통해 획득한 정보는 자함 객체내의 해당하는 항목에 맵핑되어 저장 또는 갱신되며 반대로 자함 객체내의 항목을 변경할 경우 어댑터를 통해 해당되는 체계의 서비스/장비로 명령 메시지가 송신된다. 자함 객체내의 각 항목이 갱신될 때마다 최종 갱신 시간을 속성으로 관리하여 최종 클라이언트로 하여금 정보의 획득 시점을 알 수 있도록 한다. 자함 객체가 함정의 모든 정보를 통합 관리하며 클라이언트로 정보를 제공함으로 추후 함정 내에 각종 모바일 기기, 웨어러블 기기가 도입되어도 자함 객체가 캐시로 작용해 각 체계의 오버헤드를 방지할 수 있다.

3. API 설계

자함 객체에 수집된 함정 내 각 체계 정보를 외부로 제공하고 명령을 수신하여 각 체계로의 전달하기 위해 REST 인터페이스를 기반으로 통합 API를 설계한다. HTTP 프로토콜을 이용해 자함 객체의 각 자원을 URI 주소를 통해 접근할 수 있으며 해당 자원에 대해 HTTP 메서드(GET, POST, PUT, DELETE)를 이용해 제어하게 된다. 각각의 HTTP 메서드는 CRUD(Create, Read, Update, Delete)에 각각 대응하므로 HTTP 프로토콜에 이미 정의된 메서드만을 이용해서 자원에 대한 모든 조작이 가능하다. 따라서 클라이언트는 HTTP 메서드와 URI를 이용해 기본적인 요청 메시지를 생성하여 서비스 게이트웨이로 전송하는 것으로 자함에 대한 모든 정보를 다룰 수 있게 된다. 표 1은 표적정보에 대한 HTTP 메서드별 동작 예시이다.

수신한 요청 메시지에 대해 서비스 게이트웨이는 그 결과를 응답 메시지 내에 HTTP 상태코드로 응답한다. HTTP 표준에는 다양한 상황에 대한 상태코드가 정의되어 있으며 이 상태코드를 그대로 활용하여 자함 객체에 대한 액세스 결과를 표 2의 코드로 응답하도록 한다. 자함 객체 내의 모든 자원에 대한 제어 결과를 하나의 공통된 표준 코드 체계를 통해 보고함으로써 클라이언트의 예외처리부는 간결하게 작성될 수 있다. 응답 메시지와 요청 메시지 중에서 POST, PUT, DELETE와 같이 메서드와 함께 부가정보가 함께 전달해야 하는 경우에는 HTTP 메시지에 본문에 해당 내용을 채워 전달한다. 함

표 1. 자원별 HTTP 메서드 동작 예시
Table 1. Example HTTP Methods for each Resource

요 청 / 자 원	HTTP POST [CREATE]	HTTP GET [READ]	HTTP PUT [UPDATE]	HTTP DELETE [DELETE]
/ship/cs/targets	새 표적 생성 요청	표적 목록의 요청	복수개의 표적 정보를 갱신 요청	표적 전체 삭제 요청
/ship/cs/targets/01	-	01번 표적에 대한 정보 요청	01번 표적에 대한 정보 갱신 요청	01번 표적 삭제 요청

표 2. HTTP 응답 코드를 이용한 API 호출 결과
Table 2. HTTP Response Code represents API Call Result

HTTP 응답 코드	API 호출 결과
200: Success	API 요청 수행 성공
201: Created	새 자원 생성 성공
202: Accepted	API 요청 접수 (수행 처리 완료 전 응답)
400: Bad Request	잘못된 요청(요청 메시지 오류)
401: Unauthorized	권한 없음
403: Forbidden	금지된 자원
404: Not Found	없는 자원에 대한 접근
405: Method not Allowed	자원에 대해 해당 메서드가 지원되지 않음
410: Gone	삭제된 자원에 대한 접근
500: Internal Server Error	내부 서버 오류로 요청 수행 불가
307: Temporary Redirect	별칭을 통해 자원을 요청한 경우

정 통합 서비스 프레임워크 내의 모든 메시지 본문은 JSON(JavaScript Object Notation) 형식을 따른다. JSON은 자바스크립트(Javascript)에서 객체를 텍스트 형태로 표현하는 방법으로 사람과 기계 모두에게 쉽게 읽힐 수 있는 포맷으로 자바스크립트의 객체표현법으로 소개되었지만 현재는 인터넷상에서 자료교환용 포맷으로 널리 쓰이고 있으며 대부분의 언어에서 JSON 입력 및 출력을 지원하고 있다. 이런 특징으로 함정 통합 서비스 프레임워크의 데이터포맷으로 도입하였다. 자함 객체의 어떤 자원을 GET 메서드로 요청한 경우, 해당 자원의 속성정보를 속성명과 속성의 값을 이용해 JSON 형식의 문장으로 만들어 응답메시지의 본문으로 삽입한다. 표 3은 한 표적에 대한 요청과 응답 메시지의 예시이다. PUT, UPDATE 메서드와 같이 대상 자원의 URI와 함께 생성/갱신해야할 내용이 필요한 경우는 요청 메시지의 본문에 필요한 속성 정보를 JSON 형식으로 삽입해 전달

표 3. 요청 및 응답 메시지 예시
Table 3. Example of Request and Response Messages

HTTP 요청 메시지

```
GET /ship/cs/targets/02 HTTP/1.1
Host: 10.0.1.21
```

HTTP 응답 메시지

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 518
Date: Thu, 06 Apr 2017 04:01:09 GMT

{
  "index": 2,
  "isActive": true,
  "targetNumber": "02",
  "targetName": "",
  "mainTracker": "RADAR",
  "targetSource": [
    {
      "sensor": "RADAR",
      "id": 3
    },
    {
      "sensor": "IPS",
      "id": 2
    }
  ],
  "trackType": "Point",
  "trackClass": "Surface",
                ----- 후 략 -----
```

한다. 자함 객체 내의 자원 중 하위 자원을 포함하는 항목(collection)을 호출한 경우에는 포함된 하위 자원의 목록을 JSON 형식으로 클라이언트에 반환한다. 이와 같이 HTTP 표준 프로토콜과 간결하고 다양한 플랫폼을 지원하는 JSON을 활용해 API를 설계함으로써 함정을 구성하는 체계들의 내부 통신 인터페이스와 직접 연동하지 않고도 다양한 하드웨어 및 소프트웨어 플랫폼을 활용한 클라이언트 서비스를 언제든 개발할 수 있을 것이다.

4. 자함 데이터베이스(OSDB, Own Ship Database)

자함 객체를 통해 함정 내 각 체계의 정보를 하나의 객체로 통합하고 이 객체를 하나의 JSON 문장으로 기술 가능하다는 것은 함정관련 정보 전체를 하나의 데이터로 저장하는 것도 가능하다는 것을 의미한다.

서비스 게이트웨이의 내부 혹은 외부에 자함 객체를 저장하는 자함 데이터베이스를 두고 일정 시간 간격 혹은 자함 객체의 내용이 변경되는 주기에 맞춰 데이터베이스에 기록한다. 자함 객체는 다양한 형태의 정보로 구성된 객체이므로 정형화된 자료를 저장하는 관계형 데이터베이스(RDB, Relational DataBase)를 이용해 테이블을 설계하고 자함 객체를 관리하는 데에는 무리가 있다. 따라서 JSON과 같이 semi-semantic한 문서의 저장 및 질의(Query) 처리에 특화된 문서기반 데이터베이스(Document Oriented Database)를 이용하여 자함 객체를 객체 구조 그대로 저장하고 조회할 수 있게 한다. 또한 저장된 자함 객체를 클라이언트가 호출할 수 있도록 (2)와 같이 시간관련 질의를 GET 메서드 호출시 URI의 파라미터로 수신토록 하여 자함 데이터베이스에서 해당 정보의 과거 이력을 추출하여 제공하게 한다.

/ship/cs/targets/01?from=20170401&to=20170405 (2)

이를 통해 기록 기능이 없는 장비의 정보도 기록/조회가 가능하며 각 체계, 각 기능별로 수행하던 기록 기능을 일원화하였기에 이를 이용한 새로운 서비스도 만들 수 있다. 즉 NoSQL로 대표되는 빅데이터 기술을 자함 데이터베이스를 통해 각 해군 함정에도 도입이 가능하며 함정의 임무수행 기간 전체 정보를 그대로 저장하고 임무 완료 후 사후분석을 통해 함정의 운용, 고장분석 및 각 체계/기능의 개선 요소를 도출하는 등 다방면으로 활용될 수 있을 것이다.

5. 보안 및 인증

URI를 통한 자원 접근과 HTTP 및 JSON을 이용한 본문 전달은 REST 기반 설계의 핵심이며 이들은 모두 평문(Clear Text)를 기반으로 하고 있어 바이너리 기반 메시지 프로토콜에 비해 인간이 쉽게 의미와 사용법을 파악할 수 있고 이는 설계, 구현 및 유지/보수 시에 큰 장점이 될 수 있다. 하지만 이는 동시에 메시지를 중간에 가로채어 쉽게 분석이 가능하며 해커가 다양한 공격방법과 함께 전체 체계를 교란시키는데 남용될 가능성 이 있다[14]. 이를 방지하기 위해 네트워크 레이어 상에서 SSL을 이용해 서버-클라이언트 간의 통신을 암호화하는 HTTPS 프로토콜을 사용한다.

일반적인 인터넷에서의 HTTPS 프로토콜에서는 클라이언트가 서버에 연결 요청 시 서버는 자신의 서버 인증서를 전달하고 클라이언트는 서버 인증서를 신뢰기관에 정상 발급여부를 확인하는 절차를 포함한다. 하지만 함정의 네트워크는 외부와 차단된 인트라넷이므로 신뢰기관으로 인증서 확인요청이 불가능하다. 따라서 서버 인증서 정보를 함정 내에서 사용할 클라이언트 장비에 각각 내장시켜 서버 인증서가 유효한지 체크하게 한다. 이를 통해 중간자 공격(Man In the Middle Attack)을 방지하고 인가된 장비만 함정 내에서 네트워크에 연결할 수 있도록 한다. 전투체계의 무장관련 정보 등 매우 민감한 정보에 대해서는 개별 필드를 암호화 하는 방법으로 추가적인 보안을 제공할 수 있다.

또 다른 보안 문제로 권한과 인증이 있다. 함정 내의 모든 정보가 통합되어 있는 자함 객체에서 무장정보와 같은 민감한 자원은 정해진 인원만 접근이 가능해야 한다. 또한 하나의 자원에 대해 권한의 차이를 두어 정보 조회는 누구나 가능하지만 수정/삭제는 권한을 가진 사용자만 수행토록 통제할 수 있어야 한다. 이를 위해 함정 통합 서비스 플랫폼에서는 전투체계에 존재하는 운용자 관리 기능과 토큰인증을 이용해 자함 객체 내 자원의 권한 및 접근을 통제할 수 있다. 그림 4는 API 클라이언트

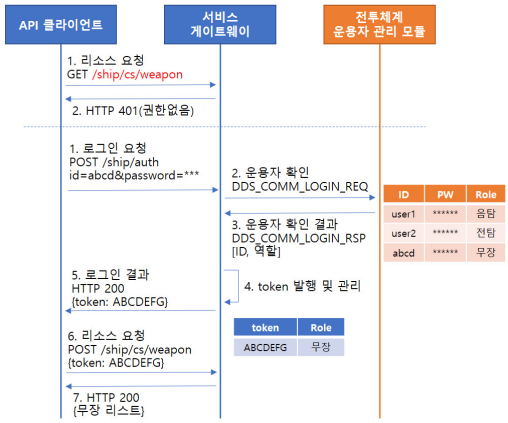

그림 4. 인증토큰을 이용한 API 요청 절차
Fig. 4. Procedure of API Request Calls with Authentication Token

가 전투체계의 운용자 정보를 이용해 로그인하고 권한이 제한된 자원에 접근하는 절차를 나타낸다. API 클라이언트는 자함 객체에 로그인을 위한 URI를 통해 운용자 정보와 패스워드를 전달한다. 서비스 게이트웨이는 전달받은 정보를 이용해 전투체계로 로그인 요청을 한다. 유효한 운용자 정보로 로그인이 이루어진 경우 토큰을 발행하고 운용자 권한과 함께 서비스 게이트웨이 저장하며 토큰을 API 클라이언트로 응답한다. 이후 API 클라이언트는 자원 접근 시 토큰을 메시지에 포함하여 접근 요청을 하고 서비스 게이트웨이는 토큰의 유효성을 판단한 뒤 해당 자원을 클라이언트로 전송한다. 만약 토큰 정보가 없거나 유효하지 않으면 HTTP 응답코드 401을 반환하여 접근한 자원에 대한 권한이 없음을 통보한다. 인증 토큰을 이용한 권한 제어 방식을 이용해 기존에 로그인 및 권한 제한이 없는 기능에 대해서도 인증절차를 추가해 보안을 강화할 수 있다.

Ⅳ. 구현 및 평가

제안된 함정 통합 서비스 프레임워크의 실현 가능성을 검증하기 위해 실제 구현을 하고 성능을 측정하였다. 전투체계 및 통합플랫폼관리체계 등과 연동 기능을 시험하기 위해 실제 함정 탑재용 장비로 구성된 함정 전투체계 테스트베드에서 통합플랫폼관리체계 시뮬레이터 및 상용CCTV 제품 등을 이용해 구축하였다. 개발환경은 표 4과 같이 구성하여 서비스 게이트웨이 프로토타입을 구현하였다. JSON 형태의 메시지를 직접 다룰 수 있는 Node.js를 서비스 게이트웨이의 기반으로 선택하였다. 서버 하드웨어를 함정 전투체계 테스트베드의 스위치와 이더넷으로 연결하였다. 통합플랫폼관리체계 시뮬레이터와 CCTV 네트워크와 이더넷으로 연결하고 항해시뮬레이터와는 RS-485 시리얼을 통해 연결하였다. 각 어댑터는 Node.js 내에 이벤트 핸들러 형식으로 구현하였다. 전투체계의 DDS 메시지는 RTI 사에서 개발하여 배포하는 rticonnextdds-connector 라이브러리를 이용해 Node.js에서 DDS 토픽 메시지를 수신할 수 있게 하였고 그 외의 연동은 Node.js에서 기본 제공하는 TCP 소켓과 시리얼 라이브러리를 사용하여 체계별 어댑터를 구현하였다. 각 어댑터는 새로운 메시지를 수신할 때마다 메시지를 파싱하여 자함 객체 내의 해당하는 값을 갱신토록

표 4. 개발환경
Table 4. Development Environment

항목	내용
서버 하드웨어 (서비스 게이트웨이)	CPU: Intel Xeon E3-1246v3 3.5Ghz RAM: 16GB ECC OS: Linux(Ubuntu 17.04, 64bit) NIC: 1 Gigabit LAN x 2 Serial-to-USB Convertor
테스트 클라이언트	Laptop PC CPU: Intel Core i7-3537U 2.0Ghz RAM: 8 GB OS: Windows 10(64bit) NIC: 1 Gigabit LAN
개발언어	Javascript (Node.js v4.7.2) HTML5 (client)
DBMS	MongoDB v3.2.11-2
오픈 라이브러리	restify v4.3.0 rticonnextdds-connector 등
성능측정도구	Apache JMeter v3.2 r1790748

하였다. 자함 객체는 자바스크립트 객체로 구현하여 Node.js에서 바로 JSON 형태로 출력이 가능한 장점을 취하고 별도의 처리 부하가 발생하지 않도록 하였다. 자함 객체를 저장하기 위한 데이터베이스로는 문서기반 데이터베이스인 MongoDB를 사용하였다. API 요청 수신 및 처리부는 restify 라이브러리를 활용하여 REST API를 구현하였다.

서비스 게이트웨이의 구현과 동시에 서비스 게이트웨이에서 제공되는 API를 통해 함정 내 정보를 요청하고 화면에 전시할 클라이언트 역시 구현하였다. 기존 체계들에서 사용되어온 특정 기반 하드웨어/소프트웨어에서 벗어나고 인터넷 표준기반의 REST의 장점을 드러내기 위해 HTML5를 이용하여 그림 5와 같이 웹앱(WebApp)

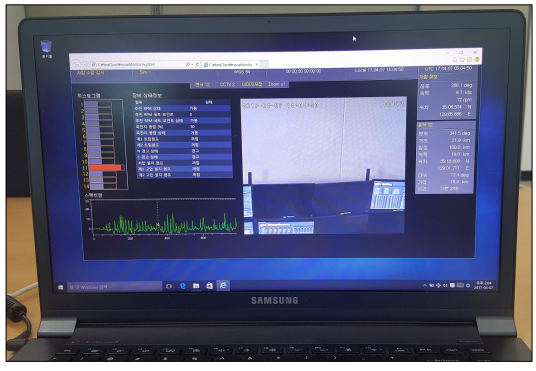

그림 5. 테스트 클라이언트 구현
Fig. 5. Implementation of Test Client

표 5. 응답시간 측정 시험 결과(밀리초)
Table 5. Result of Response Time Test(milli-second)

Test Case	Avg.	Min.	Max.	Std. Dev.
자함객체 정보 호출	1	< 1	15	0.65
자함 DB 정보 호출	6	1	19	2.7
자함 객체 + SSL	4	< 1	96	6.88
자함 DB + SSL	11	3	112	9.23

으로 구현하였다. 이 테스트 클라이언트는 전투체계를 통해 함정의 여러 지점의 자함소음 측정정보를 통합플랫폼관리체계를 통해 해당 지점의 장비 작동상태를, CCTV를 통해 해당 지점의 영상을 함께 전시한다. 이를 통해 서비스 게이트웨이가 함정 내의 각 체계의 정보를 수집하여 REST 아키텍처에 따라 설계된 API로 제공이 가능함을 확인하였다.

구현된 API가 HTTP 프로토콜을 기반으로 제공되기 때문에 인터넷 서비스를 테스트하는 도구를 이용해 구현된 서비스 게이트웨이의 성능을 측정할 수 있었다. 테스트 도구로는 Apache 재단의 JMeter를 사용하였고 서비스 게이트웨이에 접속하는 클라이언트의 수를 100개로 설정하고 10초 동안 각 클라이언트가 초당 10번씩 정보요청을 하여 초당 1000개의 메시지를 서비스 게이트웨이로 전달한 뒤 응답 메시지가 도착하기까지의 응답시간을 측정하였다. 그 결과는 표 5와 같이 측정되었다. 이 수치는 클라이언트의 요청이 서비스 게이트웨이로 도달하는 시간, 요청 처리시간, 응답이 다시 클라이언트로 도달하는 시간의 합을 의미한다. 가장 처리부하가 심한 HTTPS 프로토콜에서의 자함 데이터베이스 정보 조회가 평균 11ms에 표준편차 9.23ms로 측정되어 통합 서비스 프레임워크가 도입되어도 기존의 환경에서 발생하는 시간 지연에 큰 영향을 미치지 않음을 확인하였다. 만약 예상보다 더 많은 클라이언트가 접속하는 상황이 발생한다면 서비스 게이트웨이를 복수로 운용하고 인터넷 웹서버와 같은 방식의 Load Balancing을 통해 해결할 수 있을 것이다.

V. 결론 및 향후 계획

본 논문에서는 해군 함정을 구성하는 전투체계, 항해체계, 추진체계, 기타 각종 네트워크 기반 서비스를 하나로 통합하고 통합된 서비스를 단일 표준 API 형태로 제공하는 프레임워크를 제안하였다. 각 체계의 노드로서 체계와 직접 연동되어 정보를 교환하는 어댑터, 어댑터를 통해 수신한 각종 함정 내 정보를 하나의 객체로 표현하는 자함 객체를 설계하였다. 자함 객체를 서비스 클라이언트에 제공하기 위해 URI 형태로 자원 접근방법을 정의 하였으며 HTTP와 JSON을 이용한 API 설계방법을 기술하였다. 또한 자함 데이터베이스를 이용하여 실시간 정보뿐만 아니라 과거의 함정정보도 다룰 수 있게 하였으며 해군 함정의 특성상 요구되는 보안 기능도 SSL통한 네트워크 메시지 암호화와 전투체계의 운용자 관리 기능 연계를 통한 사용자 인증 및 권한 부여를 통해 충족하고자 하였다. 제안된 프레임워크의 실현 가능성을 확인하기 위하여 전투체계 테스트베드에서 구현하였으며 실제로 프레임워크가 작동하고 이를 통해 여러 체계에서 정보를 수신해 새로운 서비스화면을 개발할 수 있음을 보였다. 이를 통해 기존의 각 체계 개발 조직은 내부 서비스의 개발 및 개선에 집중하고 운용 단말 및 화면은 독립적으로 소요군의 운용 요구사항과 일관된 디자인 철학을 가지고 통합 개발이 가능 할 것으로 기대된다.

앞으로 본 논문에서 소개한 자함 객체를 더욱 확장하여 다양한 해군 함정을 포괄하는 표준 함정 객체모델로서 발전시키고자 한다. 이를 기반으로 함정 간 정보공유, 함정의 통합관리의 새로운 모델을 제시하고자 한다. 함정 내에서는 구성 체계 간 시너지를 낼 수 있는 기능 간 연계 방안을 도출해 새로운 운용개념, 화면, 운용기기 등을 발굴해 나가고자 한다. 또한 저장된 자함의 정보를 자동으로 분석하여 상태이상을 판단하고 조치방안을 제시하는 방법도 연구하고자 한다.

References

[1] H. Baek, K. Jung, M. Lee, J. Choi, “A study on the arrangement of integrated power system for warship,” Journal of the Korean Society of Marine Engineering, Vol. 38, No. 9 pp. 1070~1074, 2014. DOI: https://doi.org/10.5916/jkosme.2014.38.9.1070

[2] J. Park, S. Sung, Y. Lim, C. Yun, S. Kim, “An Implementation of Integrated Interface System for the Digital Ship,” Journal of the Korea Institute of

Information and Communication Engineering, Vol. 16, No. 6, pp. 1158~1166, Jun. 2012.
DOI: https://doi.org/10.6109/jkiice.2012.16.6.1158

[3] J. Ryu, "Current Status and Growth Strategy of Domestic Combat System Development," 21th Policy Forum, Defense Acquisition Program Administration, Jun. 2016.

[4] S. Ko, D. Park, "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.

[5] K. Shin, "A Study on the Republic of Korea Navy's Strategy for Dealing with the Maritime Security Threats against the East Asia Region," Dissertation of Doctor of Philosophy in Maritime Security and Policy, Korea Maritime and Ocean University, 2016.

[6] J. Kim, "Application of IOT on Naval combat system," Information and Control Symposium, pp. 280~281, Oct. 2015.

[7] R. Fielding, "Architectural Styles and the Design of Network-based Software Architectures," Dissertation of Doctor of Philosophy in Information and Computer Science, University of California, IRVINE, 2000.

[8] S. Lee, Y. Kim, T. Ha, "Total Ship Computing Environment: Development Trend and Future Development Direction," Defense & Technology, The Korea Defense Industry Association, Vol. 434, pp. 62~77, Apr. 2015.

[9] F. Belqasmi, J. Singh, S. Y. B. Melhem, R. H. Glitho, "SOAP-Based vs. RESTful Web Services: A Case Study for Multimedia Conferencing," IEEE Internet Computing, Vol. 16, Issue 4, pp. 54~63, Jul. 2012.
DOI: https://doi.org/10.1109/MIC.2012.62

[10] D. Lee, "Open API Software Framework for Information Processing of RCS-e Presence Feature," The Journal of The Institute of Internet, Broadcasting and Communication(JIIBC), Vol. 16, No. 5, Oct. 2016.
DOI: https://doi.org/10.7236/JIIBC.2016.16.5.77

[11] Y. Jeon, S. Im, H. Hwang, "Design of Open Gateway Framework for Personalized Healing Data Access," The Journal of The Institute of Internet, Broadcasting and Communication(JIIBC), Vol. 15, No. 1, Feb. 2016.
DOI: https://doi.org/10.7236/JIIBC.2015.15.1.229

[12] M. Masse, "REST API Design Rulebook: Designing Consistent RESTful Web Service Interfaces," O'Reilly Media Inc, 2011.

[13] D. Kim, S. Heo, "Distributed Contol Networks of Naval Combat Systems," Journal of the Korea Institute of Information and Communication Engineering, Vol.13, No.2, pp.41-47, 2013.

[14] M. Lee, J. Park, "An analysis on invasion threat and a study on countermeasures for Smart Car," Journal of the Korea Academia-Industrial cooperation Society(JKAIS), Vol. 13, No. 3, pp. 374-380, 2017.
DOI: https://doi.org/10.5762/KAIS.2017.18.3.374

저자 소개

김 규 백(정회원)

- 2008년 : 한국과학기술원 정보통신공학부 학사
- 2010년 : 한국과학기술원 전산과 공학석사
- 2010년~현재 : 국방과학연구소 연구원