JKSCI

한국컴퓨터정보학회논문지
Journal of The Korea Society of Computer and Information
Vol. 27 No. 5, pp. 109-116, May 2022
https://doi.org/10.9708/jksci.2022.27.05.109

A Study on Applying a Consistent UML Model to Naval Combat System Software Using Model Verification System

Seung-Mo Jung*, Woo-Jin Lee*

*Student, School of Computer Science and Engineering, KyungPook National University, Daegu, Korea
*Professor, School of Computer Science and Engineering, KyungPook National University, Daegu, Korea

[Abstract]

Recently, a model-based development method centered on highly readable and standardized UML (Unified Modeling Language) models has been applied to solve unclear communications in large-scale software development. However, it is difficult to apply consistent UML models depending on software developers' proficiency, understanding of models and modeling tools. In this paper, we propose a method for developing a Model Verification System to apply an consistent UML model to software development. Then, the developed Model Verification System is partially applied to the Naval Combat System Software development to prove its function. The Model Verification System provides automatic verification of models created by developers according to domain characteristics. If the Model Verification System proposed in this paper is used, It has the advantage of being able to apply the consistent UML model more easily to Naval Combat System Software Development.

▸ **Key words**: Unified Modeling Language, Model Driven Development, Model Verification System, Modeling Tool, Naval Combat System Software

[요 약]

최근 대규모 소프트웨어 개발하는 데 있어 불명확한 의사소통을 해결하기 위해 가독성이 높은 표준화된 UML(Unified Modeling Language) 모델 중심의 모델 기반 개발 방법이 적용되고 있다. 하지만 소프트웨어 개발자들의 숙련도, 모델 및 모델링 도구의 이해도에 따라 대규모 소프트웨어에 일관성 있는 UML 모델을 적용하기에는 어려움이 발생한다. 이에 본 논문에서는 소프트웨어 개발에 일관성 있는 UML 모델을 적용하기 위한 모델 검증 시스템 개발 방법을 제시한다. 그리고 개발된 모델 검증 시스템을 함정 전투체계 소프트웨어 개발에 일부 적용하여 기능을 입증한다. 모델 검증 시스템은 개발자들이 작성한 모델들을 도메인 특성에 맞게 자동으로 검증할 수 있는 기능을 제공한다. 본 논문에서 제안한 모델 검증 시스템을 사용하면 함정 전투체계 소프트웨어 개발에 좀 더 쉽게 일관성 있는 UML 모델을 적용할 수 있는 장점을 가진다.

▸ **주제어**: 통합모델링언어, 모델기반개발, 모델검증시스템, 모델링 도구, 함정전투체계 소프트웨어

• First Author: Seung-Mo Jung, Corresponding Author: Woo-Jin Lee
*Seung-Mo Jung (seungmo1025@knu.ac.kr), School of Computer Science and Engineering, KNU / Hanwha Systems
*Woo-Jin Lee (woojin@knu.ac.kr), School of Computer Science and Engineering, KNU / SWRC
• Received: 2022. 03. 23, Revised: 2022. 04. 26, Accepted: 2022. 04. 27.

Copyright © 2022 The Korea Society of Computer and Information
http://www.ksci.re.kr pISSN:1598-849X | eISSN:2383-9945
www.kci.go.kr

I. Introduction

최근 대규모 소프트웨어를 개발하는 데 있어 고객과 개발자 간의 의사소통, 개발자들 간의 의사소통은 매우 중요한 요소로 부각 되고 있다[1][2]. 소프트웨어 개발과정에서 의사소통이 불명확하다면 개발된 제품의 신뢰성 저하 및 심각한 오류를 발생시킬 수 있기 때문이다. 고객과 개발자 간의 불명확한 의사소통은 정확한 요구사항 전달에 대한 문제를 발생시키고 개발자들 간의 불명확한 의사소통은 소프트웨어 기능 통합에 대한 문제를 발생시킨다. 또한, 불명확한 의사소통으로 인해 발생한 문제점들을 해결하기 위해 개발기간 증가에 따른 개발 비용 증가로 의사소통은 소프트웨어 제품개발에 중요한 요소이다.

이러한 문제점들을 해결하기 위해 최근 소프트웨어 개발에서 불명확한 의사소통을 해결하기 위한 연구가 활발히 진행되고 있다. 개발 초기부터 불명확한 의사소통을 배제하여 신뢰성이 높은 효율적인 소프트웨어를 개발하기 위해 많은 연구를 하고 있다[3][4]. 현재까지 연구된 방법에는 여러 가지가 있겠지만 그중 그래픽 한 객체들을 사용하여 가독성을 높인 UML(Unified Modeling Language)을 이용한 모델 기반 개발 방법이 주목받고 있다[5][6]. 모델 기반 개발 방법이란 가독성이 높은 표준화된 모델 중심의 개발 방식으로 작성된 모델로부터 프로그램 코드뿐만 아니라 다양한 문서를 자동으로 생성하는 개발 방식이다[7]. 가독성이 높은 표준화된 UML 모델 객체 들을 활용하여 소프트웨어를 개발하면 불명확한 의사소통을 감소 시켜 신뢰성이 높은 제품을 생산할 수 있다는 장점이 있다.

함정 전투체계 (Naval Combat Management System/CMS)를 개발하기 위한 함정 전투체계 소프트웨어 개발에서도 원활한 의사소통에 장점을 가진 표준화된 UML 모델 객체들을 활용한 개발 방법이 기본적으로 적용되고 있다. 함정 전투체계란 공중, 해상 및 수중으로부터의 복합적 상황에 대응하기 위하여 함정이 보유한 모든 센서, 무장 등을 최대한 효율적으로 통제/분배하는 자동화 체계로 체계를 작동하기 위한 다양한 소프트웨어들이 탑재되어 있다.

하지만 표준 UML 모델 객체를 특정 도메인 특성을 가진 함정 전투체계 소프트웨어에 적용하기 위해서는 UML 모델 적용 범위와 수많은 소프트웨어에 통일성 있는 UML 적용, 일관성 있는 UML 적용 등 어려움 들이 발생한다. 이를 해결하기 위해 개발 초기에 문서방식의 모델 적용 가이드라인을 제공하지만 개발자 마다의 개발 숙련도와 모델 및 모델링 도구의 이해도 차이에 따라 달리 적용될 수 있다는 단점으로 많은 수의 개발자들이 서로 협력하여 개발하는 대규모 소프트웨어 개발에는 한계가 발생한다. 또한, 모델의 이해도가 높은 소프트웨어 개발자가 적용된 모델들의 신뢰도를 높이기 위해 수동으로 검토를 하지만 대규모 소프트웨어에 적용된 UML 모델들을 소수의 인원이 검토하기에는 많은 시간이 걸리는 단점이 있다.

이에 본 논문에서는 함정 전투체계 소프트웨어 개발에 일관성 있는 UML 모델을 적용하기 위한 모델 검증 시스템 (MVS : Model Verification System) 개발방안을 제시한다. 모델 검증 시스템은 개발자들이 작성한 UML 모델들을 입력으로 받아 도메인 특성에 맞게 자동으로 검증할 수 있는 기능을 제공한다. 이를 통해 함정 전투체계 소프트웨어 모델 적용의 정확성 및 일관성 있는 UML 모델들을 적용 하는데 그 목적을 가진다. 본 논문에서 제시한 모델 검증 시스템의 기능을 소프트웨어 개발에 활용한다면 더욱 신속한 시간에 일관성 있는 UML 모델들을 소프트웨어 개발에 적용할 수 있다.

본 논문의 구성은 다음과 같다. 2절에서는 모델 검증 시스템을 개발하는데 필요한 개발환경에 대해서 간략히 설명하고 3절에서는 개발환경과 모델 접근 방법을 활용하여 모델 검증 시스템에 대한 개발과정을 설명한다. 그리고 4절에서는 개발된 모델 검증 시스템에 대한 기능적인 시험과 함정전투체계 소프트웨어 일부 모듈에 직접 적용하면서 기능을 입증한다. 마지막 5절에서는 결론 및 추후 수행과제로 이 논문을 마무리한다.

II. Preliminaries

이 절에서는 UML 모델 검증에 대한 관련 연구와 모델 검증 시스템을 개발하는데 필요한 개발환경에 대해서 설명한다.

1. Related Work for UML Verification

UML(Unified Modeling Language)은 풍부한 구성요소를 가지므로 개발하고자 하는 시스템을 바라보는 관점에 따라 다양하고 상세하게 표현할 수 있다. 하지만 표현된 UML 모델로부터 작성된 다이어그램에 대한 일관성은 보장하지 못한다[8]. UML에서 사용하는 모델들은 표준이라는 장점이 있지만 그 모델들을 사용하는 개발자들의 개발 숙련도, 모델 및 모델링 도구에 대한 이해도에 따라 달리 적용될 수 있는 단점을 가진다. 따라서 이를 해결하기 위해 UML 다이어그램의 일관성을 검증하기 위한 많은 연

www.kci.go.kr

구들이 진행되었고 검증하는 방법에는 크게 정형명세기법, 확장UML기법, 비매개표현기법 등이 있다[9][10].

- 정형명세기법 : 다이어그램의 일관성 검증에 가장 많이 이용되고 있는 방법으로 수리, 논리에 기반하여 소프트웨어 시스템을 명세, 개발, 검증하는 방법으로 주로 시퀀스 다이어그램, 상태 머신 다이어그램의 일관성 검증에 사용된다[10].
- 확장UML기법 : UML 다이어그램에 Constraint나 Stereotype과 같은 매개적인 표현을 추가하여 일관성을 검증하는 방법으로 주로 시퀀스 다이어그램, 콜로보레이션 다이어그램, 상태 머신 다이어그램의 일관성 검증에 사용된다[11].
- 비매개표현기법 : 매개적인 표현 없이 상호 변경 규칙 또는 일관성 점검규칙을 정하고 점검 대상 다이어그램에 점검규칙을 적용하여 일관성을 검증하는 방법으로 주로 클래스 다이어그램, 시퀀스 다이어그램, 상태 머신 다이어그램의 일관성 검증에 사용된다[12].

UML 다이어그램의 일관성을 검증하는 방법에는 여러 검증 방법들이 있지만 대부분 학문적으로만 연구가 되어 왔고 특성화된 도메인을 가지고 있는 산업 현장 소프트웨어 개발에 적용하기에는 한계가 있다.

2. Background

2.1 Components of Unified Modeling Language

UML(Unified Modeling Language)은 요구 분석, 시스템 설계, 시스템 구현 등의 시스템 개발 과정에서 개발자 간의 의사소통을 원활하게 하기 위한 표준화된 모델링 언어로 이 표준안은 OMG(Object Management Group)에서 관리한다. UML은 기본 요소를 구성하는 4개의 사물과 사물 간의 관계를 나타내는 4개의 관계, 사물과 관계를 도형으로 표현하는 9개의 다이어그램으로 구성되어 있다. Fig. 1은 UML의 구성요소를 도식화하여 보여준다.

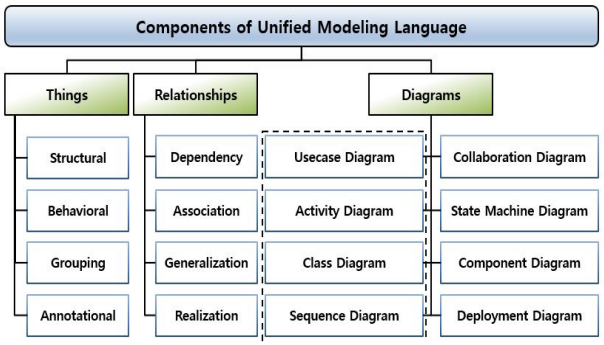

Fig. 1. Components of Unified Modeling Language

함정전투체계 소프트웨어 개발에 사용하는 UML 다이어그램은 유스케이스 다이어그램, 액티비티 다이어그램, 클래스 다이어그램, 시퀀스 다이어그램이며 각 다이어그램 설명은 아래와 같다.

- 유스케이스 다이어그램(Usecase Diagram) : 시스템과 사용자의 상호작용을 표현해주는 다이어그램
- 액티비티 다이어그램(Activity Diagram) : 유스케이스 내부에 대한 구체적인 흐름을 표현하기 위해서 사용되며 전체적인 제어 흐름을 보여주는 다이어그램
- 클래스 다이어그램(Class Diagram) : 시스템을 구성하는 클래스들 사이의 관계를 표현해주는 다이어그램
- 시퀀스 다이어그램(Sequence Diagram) : 객체를 정의하고 객체 간의 상호작용 메시지를 시간의 흐름에 따라 나타내는 다이어그램

2.2 Component of Naval Combat System

함정 전투체계는 함정에서 두뇌와 같은 역할을 수행한다. 즉, 함정에 탑재된 센서 장비, 무장 장비, 기타 장비들을 연동/제어하여 작전에 필요한 정보들을 종합적으로 수집하고 그 수집된 정보들을 활용하여 전술상황평가, 교전 등의 기능을 효율적으로 수행할 수 있도록 지원하는 일련의 자동화된 무기체계이다[13].

함정 전투체계는 센서(Sensor), 무장(Weapon), 데이터링크(Data Link), 지휘무장통제체계(Combat Fire Command System, CFCS)로 구성되어 있으며 기능을 수행하기 위해 다양한 소프트웨어들이 탑재되어 유기적으로 실행되고 있다. 탑재된 소프트웨어를 개발하는데 표준화된 UML 객체들을 활용한 모델링 과정이 포함되어야 한다. Fig. 2는 함정 전투체계 시스템의 구성요소를 보여준다.

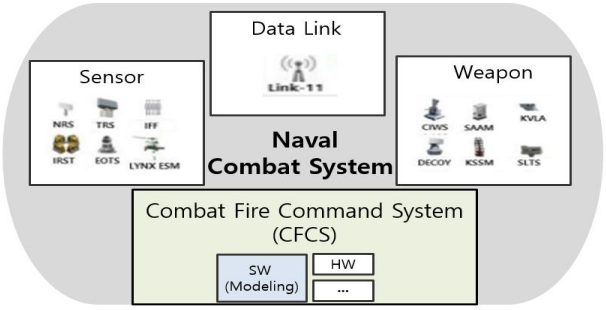

Fig. 2. Components of Naval Combat System

2.3 Modeling Tool for Naval Combat System

OMG에서 정의된 표준 UML 모델들을 시스템 개발에 사용하기 위해 다양한 모델링 도구들이 있다. 대표적인 모델링 도구에는 Rational Rhapsody, Rational Rose, Enterprise Architect, StarUML, Visio 등이 있다.

www.kci.go.kr

모델 검증 시스템에서 모델 검증에 사용한 모델링 도구는 IBM사의 Rational Rhapsody이다. Rational Rhapsody는 UML의 장점을 살릴 수 있도록 고안된 실무형 개발 프로세스로써, UML을 가장 잘 적용할 수 있다고 알려져 있다[14]. 또한, UML 표기법을 시각화하여 모델로 제공하고 모델에 대한 코드를 자동으로 생성시켜 주어 설계와 구현을 모델링 도구에서 직접 할 수 있는 장점이 있다. 이러한 장점 때문에 대규모 소프트웨어 개발에 모델을 적용하기 위해 많이 사용되고 있다. Rational Rhapsody의 기본 프로세스는 ML3, ML2, ML1, ML0 계층을 통해서 표준 UML 모델을 제공하고 모델로부터 소스 코드를 생성하게 만들어 준다. Fig. 3은 Rational Rhapsody의 구조와 프로세스를 도식화하여 보여준다.

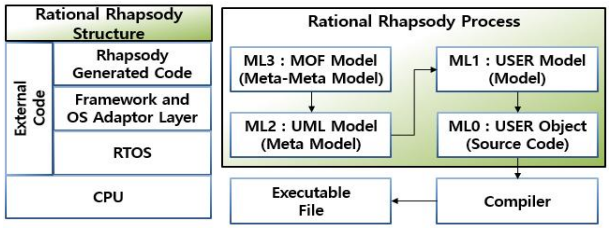

Fig. 3. Rational Rhapsody Structure and Process

Table 1은 Rational Rhapsody의 프로세스 중에서 ML0 ~ ML3의 4단계를 간단히 설명한 표이다.

Table 1. Rational Rhapsody Process (Layer)

Layer	Discription
ML3 (Meta-Meta Model)	Define the language used to set the meta model.
ML2 (Meta Model)	Define the language used to set the model.
ML1 (Model)	Define the language used to model domain areas.
ML0 (Source Code)	Define the created model as source code.

본 논문에서 개발된 모델 검증 시스템은 Rational Rhapsody로 작성된 Meta Model에 접근하여 모델 정보들을 수집한다. 그리고 그 정보들을 융합한 다음 도메인 특성을 반영하여 UML 모델들을 자동으로 검증한다.

III. The Proposed Scheme

이 절에서는 모델 검증 시스템에 대한 개발과정을 설명한다. 모델 검증 시스템은 도메인 특성화된 모델들을 검증 시나리오에 따라서 자동으로 검증하는 시스템이다. 개발된 모델 검증 시스템을 활용하면 함정 전투체계 시스템에 좀 더 신속하게 일관성 있는 UML 모델을 적용할 수 있다.

1. Naval Combat System SW Modeling Process

함정전투체계 소프트웨어 개발에서 원활한 의사소통의 장점을 가진 표준화된 UML 모델 객체들을 활용한 개발 방법을 기본적으로 적용하고 있다. Fig. 4는 함정 전투체계 소프트웨어의 모델링 적용과정을 설명한 그림이다.

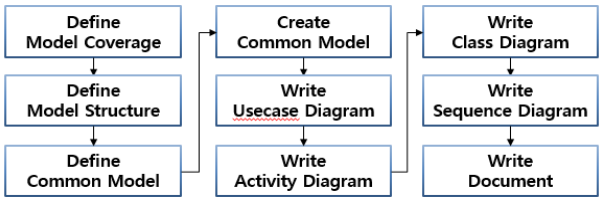

Fig. 4. Naval Combat System Software Modeling Process

함정전투체계 소프트웨어 모델링 과정은 9가지로 구성되어 있다. 먼저 모델 적용 범위를 정의해야 한다. 사업에서 요구하는 모델이 무엇인지 모델로 표현하는 범위를 정한다. 모델 적용 범위가 정해지면 모델 구조를 정의해야 한다. 모델 구조 정의는 표현이 필요한 모델들을 어디에 작성할지에 대한 통일적인 구조를 정의한다. 그리고 공통적으로 사용할 도메인 특성을 반영한 공통 모델을 정의한다. 함정 전투체계 소프트웨어에서는 Actor, Stereotype, Event 모델들이 공통 모델이다. Actor와 Stereotype 모델은 사업에서 필요한 모델들로 정의하고 Event 모델은 MOMAT에 있는 메시지들로 정의한다. 함정 전투체계에서는 모듈간의 메시지를 송/수신 위해 MOMAT(Message Oriented Management Analysis Tool)을 사용하여 수많은 메시지에 대한 정의 및 관리를 한다[15]. 그리고 정의된 공통적인 모델들을 개발자들이 모델 작성할 때 사용하기 위해 생성하는 과정을 거치게 된다.

이러한 과정이 끝나면 개발자들은 공통 모델들을 활용하여 담당 모듈에 대한 다이어그램 모델들을 작성한다. 시스템 요구사항을 표현하는 유스케이스 다이어그램 작성, 시스템의 처리 흐름을 순서에 따라 표현하는 액티비티 다이어그램 작성, 시스템에서 식별된 클래스간의 상호 작용을 표현하는 클래스 다이어그램 작성, 시스템에서 시계열로 정렬된 객체 상호작용을 표현하는 시퀀스 다이어그램을 작성한다. 그리고 작성된 다이어그램 모델들은 최종적으로 소프트웨어 요구사항 명세서(SRS), 소프트웨어 설계 명세서(SDD) 문서에 작성하고 모델링 과정을 마무리한다.

모델 개발 방법을 적용하기 위해서 사용한 모델링툴은 Rational Rhapsody이며 Fig. 5는 함정전투체계 소프트웨어에 모델링을 적용하기 위한 UML 모델 구조를 보여준다.

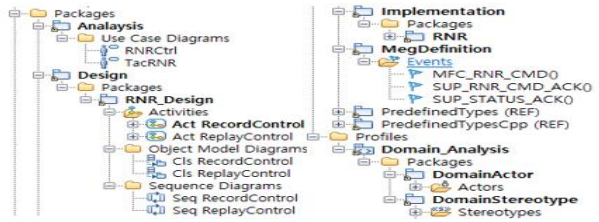

Fig. 5. UML Structure of Naval Combat System Software

함정 전투체계 소프트웨어의 UML 모델 구조는 4개의 모델 패키지와 1개의 프로파일로 구성되어 있다.

- Analysis : 소프트웨어 요구사항 모델 (User Model) 정의 영역으로 유스케이스 다이어그램을 정의
- Design : 소프트웨어 설계 모델 (User Model) 정의 영역으로 액티비티 다이어그램, 설계용 클래스 다이어그램, 시퀀스 다이어그램을 정의
- Implementation : 소프트웨어 구현 모델 (User Model) 정의 영역으로 구현에 필요한 클래스 다이어그램을 정의
- MegDefinition : 이벤트 모델 (Common Model) 정의 영역으로 소프트웨어 개발에 필요한 이벤트들을 정의
- Domain_Analysis : 프로파일 모델 (Common Model) 정의 영역으로 도메인에 필요한 Actor, Stereotype 등 도메인에 필요한 공통 모델들을 정의

2. Model Verification System (MVS)

모델 검증 시스템 (MVS)의 주된 기능은 개발자들이 작성한 모델들을 검증 요구사항에 맞게 자동으로 검증하는 시스템이다. 개발자가 작성한 UML 모델의 Meta Model 정보들을 읽어와 그 정보들을 융합하여 모델 규칙에 맞는지 자동으로 검증하고 그 검증 결과를 개발자한테 알려준다. 만약 검증 결과가 실패이면 실패 정보들을 전시하여 개발자가 수정할 수 있도록 지원해 준다. Fig. 6은 모델 검증 시스템의 기본 프로세스를 도식화하여 보여준다.

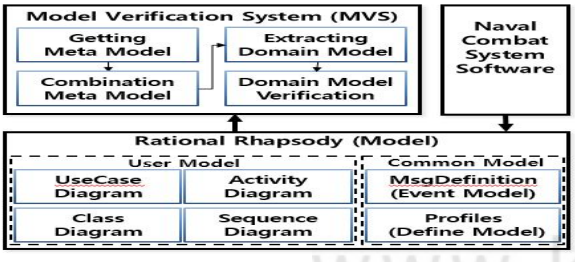

Fig. 6. Model Verification System Process

모델 검증 시스템의 프로세스는 크게 4가지로 구분되어 있다. Getting Meta Model은 모델링 도구로부터 User Model과 Common Model의 Meta Model 정보를 얻어오는 과정이고 Combination Meta Model은 얻어온 Meta Model의 정보들을 통합하는 과정이다. Extracting Domain Model은 통합된 Meta Model로부터 도메인에 필요한 모델들을 추출하는 과정이고 Domain Model Verification은 추출된 도메인 모델에서 모델 규칙 및 모델 검증 요구사항에 맞게 검증하는 과정이다.

2.1 Requirement Analysis for MVS

요구사항 분석 단계에서는 모델 검증 시스템 개발에 필요한 요구사항에 대해서 분석하였다. 모델 검증 시스템은 개발자가 작성한 사용자 모델에 대한 정확성, 사용자 모델과 사용자 모델 간의 일관성, 사용자 모델과 공통 모델 간의 일관성을 보장해야 한다. 이를 위해 4가지 다이어그램에 대한 검증 요구사항들을 수립하였고 Fig. 7은 모델 검증 시스템에 대한 요구사항을 도식화한 그림이다.

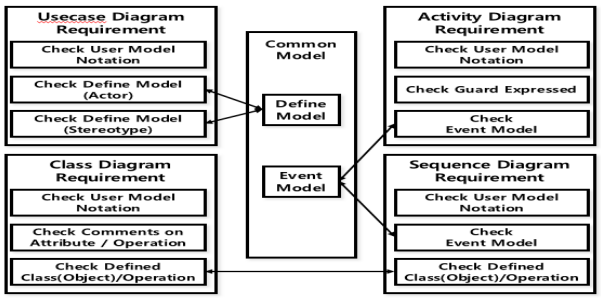

Fig. 7. Model Verification System Requirements

먼저 유스케이스 다이어그램의 검증 요구사항은 Boundary 안의 표현 규칙에 대한 기본적인 UML 표기법 검사, Define Model에 정의된 Actor와 Stereotypes 사용 여부를 검사해야 한다. 액티비티 다이어그램의 검증 요구사항은 시작점과 종료점 표현 여부, 조건문에 가드 표현 여부에 대한 기본적인 UML 표기법 검사, SendAction에서 정의된 Event Model 사용 여부 검사이다.

클래스 다이어그램의 검증 요구사항은 클래스명의 규칙 통일, 기본적인 클래스의 구조가 통일되었는지에 대한 기본적인 UML 표기법에 대한 검사, 클래스의 변수 및 함수에 주석이 제대로 표현되었는지를 검사, 클래스 다이어그램에 정의된 모든 클래스(객체), 함수가 시퀀스 다이어그램에 표현되어 있는지에 대한 검사이다. 마지막 시퀀스 다이어그램의 검증 요구사항은 Interaction Operation에서 opt와 alt의 표현 규칙에 대한 기본적인 UML 표기법 검

www.kci.go.kr

사, Event, Reply Message에서 공통 모델에 정의된 Event Model 사용 여부 검사, 시퀀스 다이어그램에 표현되어 있는 모든 클래스(객체), 함수가 클래스 다이어그램에 정의되어있는지에 대한 검사이다.

2.2 Design and Implementation for MVS

모델 검증 시스템은 모델접근언어를 활용하여 설계/구현을 하였다. 모델접근언어는 메타모델에 접근하여 모델의 정보를 얻어올 수 있는 함수들을 제공한다. 모델접근언어는 IRPModelElement를 SuperClass로 상속받은 수 많은 클래스들로 구성되어 있다. 모델 검증 시스템은 모델 접근언어를 활용하여 개발하였으며 개발자들이 작성한 모델들과 공통 모델들의 메타 정보를 얻어와서 도메인 특성에 맞게 자동으로 검증하는 기능을 수행한다. Fig. 8은 모델 검증 시스템 설계에 대한 클래스 다이어그램을 보여준다.

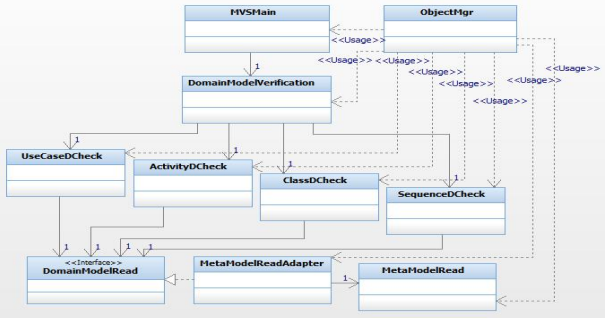

Fig. 8. Class Diagram of Model Verification System

모델 검증 시스템의 클래스 다이어그램에서 주요 기능을 담당하는 클래스에 대한 설명은 아래와 같다.

- UseCaseDCheck : 유스케이스 다이어그램의 검증을 담당하는 클래스로 개발자들이 작성한 유스케이스 다이어그램에서 액터, 유스케이스, 관계 모델 정보들을 읽어오고 공통모델의 Define Model에서 정의된 액터와 스테레오타입 모델 정보들을 읽어온다. 읽어온 모델 정보들을 활용하여 유스케이스와 유스케이스 간에 관계에 Associaton의 존재 여부검사, Define Model에 정의된 액터와 스테레오타입이 유스케이스 다이어그램에서 사용했는지에 대한 일관성을 검사한다.
- ActivityDCheck : 액티비티 다이어그램의 검증을 담당하는 클래스로 개발자들이 작성한 액티비티 다이어그램에서 시작점, 종료점, 조건문, SendAction 모델 정보들을 읽어오고 공통 모델의 Event Model에서 이벤트 정보들을 읽어온다. 읽어온 정보들을 활용하여 시작점/종료점 표현 여부, 조건문 표현 여부가 제대로 되었는지 검사하고 Event Model에 정의된 이벤트들이 SendAction에서 제대로 사용했는지를 검사한다.
- ClassDCheck : 클래스 다이어그램의 검증을 담당하는 클래스로 개발자들이 작성한 클래스 다이어그램에서 클래스(객체) 정보, 변수 정보, 함수 정보의 모델 정보들을 읽어오고 시퀀스 다이어그램에서 클래스(객체) 정보와 함수 정보를 읽어온다. 읽어온 정보들을 활용하여 클래스의 이름 규칙, 공통 클래스 간의 연관관계, 변수의 주석 여부, 함수의 주석 여부를 검사하고 시퀀스 다이어그램에서 읽어온 정보들과 비교하여 클래스 다이어그램에서 정의된 클래스(객체)와 함수가 시퀀스 다이어그램에 모두 표현되었는지를 검사한다.
- SequenceDCheck : 시퀀스 다이어그램의 검증을 담당하는 클래스로 개발자들이 작성한 시퀀스 다이어그램에서 클래스(객체), 함수, Interaction Operator, Event, Reply Message 모델 정보를 읽어오고 클래스 다이어그램에서 클래스(객체), 함수 정보를 읽어온다. 또한. 공통 모델의 Event Model에서 이벤트 정보들을 읽어온다. 읽어온 정보들을 활용하여 Interaction Operator의 정확한 표현 여부를 검사하고 시퀀스 다이어그램에서 표현된 클래스(객체)와 함수가 모두 클래스 다이어그램에 정의되어있는지를 검사한다. 또한, Event Model에 정의된 이벤트들이 Event와 Replay Message에서 제대로 사용했는지를 검사한다.
- MetaModelRead : 개발자들이 작성한 User Model과 Common Model의 Meta Model에 접근하여 모델 정보를 얻어오는 기능을 담당한다.

IV. Test and Evaluation

이 절에서는 개발된 모델 검증 시스템에 대한 기능을 입증하기 위해 테스트 모델에 대한 기능적인 시험과 함정 전투체계 시스템 소프트웨어에 적용하여 결과를 도출하였다.

1. Functional Test for Model Verification System

기능적인 면을 시험하기 위해 검증할 수 있는 모든 경우에 대한 잘못 작성된 테스트 모델을 작성하였다. 유스케이스 다이어그램에서는 유스케이스 간의 Association 관계를 표시하였고 Defined Model에 정의되지 않은 Actor와 Stereotype를 사용하였다. 액티비티 다이어그램에서는 시작점 및 조건문 한쪽에 가드를 표현하지 않았다. 또한, Send Action에 Event Model에서 정의되지 않은 Test_Event를 사용하였다. 클래스 다이어그램에서는

www.kci.go.kr

TestClass3의 이름 규칙 및 CTestClass1과 TestClass3의 관계를 Composition에서 Association으로 잘못 표현하였다. 또한 Test_Oper1 함수에서는 주석을 표시하지 않았고 시퀀스 다이어그램에는 표현되지 않은 TestClass3 클래스와 Test_Oper1 함수를 정의하였다. 시퀀스 다이어그램에서는 Interaction Operator에 opt 대신 alt로 잘못 표시하였고 Event Model에 정의되어 있지 않은 Test_Event를 Event로 표시하였다. 또한, 클래스 다이어그램에 정의되지 않은 Test_Oper5 함수를 표현하였다. 그림 Fig. 9 는 모델 검증 시스템의 기능을 테스트하기 위한 테스트 모델들의 예제들을 보여준다.

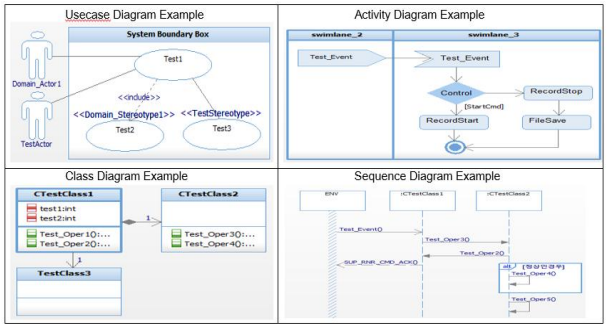

Fig. 9. Test Model Example

작성된 테스트 모델을 입력으로 모델 검증 시스템을 실행시키면 먼저 모델의 구조를 읽어와서 트리 구조로 보여준다. 그 트리 구조에서 검증받고 싶은 부분을 선택하고 해당 모델 검증을 선택하면 검증을 시작한다. 모델 검증이 끝나고 나면 검증 결과를 전시하고 검증 결과에서 수정할 부분이 있으면 검증내용을 참고하여 해당 모델들을 수정하면 된다. 시험 결과 잘못 작성된 부분들을 모두 검출하는 것을 확인하였다. Fig. 10은 모델 검증 시스템으로 테스트 모델들을 입력으로 받아 검증한 결과를 보여준다.

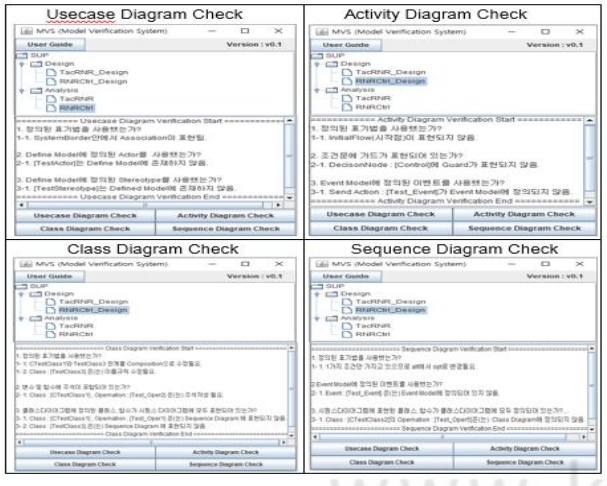

Fig. 10. Result of Model Verification System

2. Evaluation for Model Verification System

함정전투체계 소프트웨어의 수많은 기능 중 체계지원 일부 기능에 적용하면서 모델 검증 시스템의 기능을 입증하였다. 체계지원은 기록 및 재생, 전술정보 기록 및 재생 등 함정전투체계의 정보들을 쉽게 분석하기 위한 지원을 담당하는 소프트웨어들로 구성되어 있다. 모델 검증 시스템의 기능을 평가하기 위해 6개 MODEL과 4개 CASE로 모델들을 정의하였다. CASE1은 개발자들이 작성한 User Model에 대한 정확성 검증이고 CASE2는 User Model과 User Model 간의 일관성 검증이다. 그리고 CASE3은 User Model과 Define Model 간의 일관성 검증이고 CASE4는 User Model과 Event Model 간의 일관성 검증이다. Model은 M1은 Usecace Model, M2는 Activity Model, M3은 Class Model, M4는 Sequence Model, M5는 Define Model, M6은 Event Model로 정의하였다.

모델 검증 시스템을 활용하여 체계지원의 4개 모듈의 모델을 검증한 결과 CASE2와 CASE4의 검증 결과가 높게 나왔다. 다 수의 모델들을 사용하여 연관성이 있는 모델들의 일관성을 검증하는 CASE일수록 높게 측정되었다. 모델 검증 시스템을 활용하여 체계지원의 4개 모듈의 모델들을 검증한 결과 수동으로 검증한 결과보다 좀 더 신속하고 정확하게 모델들을 검증하는 것을 확인하였다. 이는 모델 검증 시스템이 검증 시나리오에 따라 개발자들이 작성한 모델들을 자동으로 검증하기 때문에 모델 검증 시간 및 모델 검증 비용이 줄어든다. Table 2는 개발한 모델 검증 시스템으로 체계지원 4개 모듈의 작성된 모델들에 대한 검증 결과를 CASE 별로 수치화한 표이다.

Table 2. Result of Model Verification System

	CASE1 (M1~M4)	CASE2 (M3,M4)	CASE3 (M1,M5)	CASE4 (M2,M4,M6)
RNRCtrl	0	3	2	3
TacRNR	0	3	1	2
RecordCtrl	0	1	0	1
ReplayCtrl	0	2	0	1

V. Conclusions

본 논문에서는 함정 전투체계 소프트웨어 개발에 일관성 있는 UML 모델을 적용하기 위한 모델 검증 시스템 개발방안을 제안하였고 개발환경 및 개발과정에 대해서 설명하였다. 그리고 개발된 모델 검증 시스템을 함정 전투체계 소프트웨어 모듈에 일부 적용하면서 기능 또한 입증 하

www.kci.go.kr

였다. 모델 검증 시스템을 활용하면 개발자들이 작성한 모델들과 도메인에 정의된 모델들을 자동으로 검증하기 때문에 더욱 신속하고 정확하게 일관성 있는 UML 모델들을 적용할 수 있다. 추후 연구과제에서는 도메인 특성에 맞는 UML 모델 생성 기능 추가 및 다양한 사례연구를 통해 함정 전투체계 소프트웨어 개발에 유연하게 적용하기 위한 더 많은 연구가 필요하다.

REFERENCES

[1] Jang, Chang Ki, "Effect of Collaboration Tools on Stakeholders' Communication in Software Development Project," The Graduate School of Information Yonsei University, 2022.

[2] Seok-Kwan Kim, Gab-Sang Ryu, "Research for improving quality of SI(System integration) development project," The Journal of Korea Institute of Information, Electronics, and Communication Technology, Vol.11 No.3, pp. 215-220. Jun. 2018.

[3] Hyunsik Kim, Seokjun Hong, Dongho Kwon, Juhyun Kim, and Jungmok Ma, "The Study on Automated Artillery Fire Direction for Future Warfare with UML," Korean Journal of Computational Design and Engineering, Vol. 23, No. 4, pp. 394-403. Dec. 2018.

[4] Yun-Ho Kim, "Information Structuring of Diagram Repository for UML Diagrams," Journal of the Korea Institute of Information and Communication Engineering, Vol. 23, No. 12, pp.1588~1595, Dec. 2019.

[5] 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. 23, No. 10, pp. 115-123, Oct. 2016.

[6] G. Martin, "UML for Embedded Systems Specification and Design : Motivation and Overview," Proceedings 2002 Design, Automation and Test in Europe Conference and Exhibition, 2002, pp. 773-775, Mar. 2002. DOI: 10.1109/DATE.2002.998386.

[7] M. U. Khan, K. Geihs, F. Gutbrodt, P. Gohner, and R. Trauter, "Model-Driven Development of Real-Time Systems with UML 2.0 and C," MBD-MOMPES'06, pp. 10-42, Mar. 2006. DOI: 10.1109/MBD-MOMPES.2006.21.

[8] Ha Il-Kyu, "Improvement of Consistency for UML Diagrams with Cross Checking Rules," Journal of information and communication convergence engineering. vol. 16, no. 6, pp. 1291-1299, Jun. 2012.

[9] Francisco J. Lucas, Fernando Molina and Ambrosio Toval, "A systematic review of UML model consistency management," Information and Software Technology. vol. 51, no. 12, pp. 1631-1645, May. 2009.

[10] M. Usman, A. Nadeem, "A survey of Consistency Checking Techniques for UML Models," 2008 Advanced Software Engineering and Its Applications, pp. 57-62, Dec. 2008. DOI: 10.1109/ASEA.2008.40.

[11] G. Engles, J. H. Hausmann, R. Heckel and S. Sauer, "Testing the Consistency of Dynamic UML diagrams," Proc. Sixth International Conference on Integrated Design and Process Technology (IDPT 2002), 2002.

[12] B. Graaf and A-V. Deursen, "Model-Driven Consistency Checking of Behavioral Specifications," MOMPES'07, pp. 115-126, Mar. 2007. DOI: 10.1109/MOMPES.2007.12.

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

[14] Modeling tool Rational Rhapsody from IBM, Homepage : https://www.ibm.com/kr-ko/products/systems-design-rhapsody

[15] S. Kyoung-Sub, K. Dong-Seong, C. Yoon-Suk, "A Design of Message Oriented Management and Analysis Tool for Naval Combat Systems", Journal of the Institute of Electronics and Information Engineers, Vol. 51, NO. 2, pp. 197-204, Feb. 2014.

Authors

Seung-Mo Jung received the M.S. degree in Electronic, Electrical, Control & Instrumentation Engineering from Hanyang University, Korea, in 2010. He is currently pursuing the Ph.D. degree in the School of Computer Science and Engineering at KyungPook National University from 2021. Also, He is currently working in Hanwha Systems Co. from 2010. He is interested in Combat System Software, Software Modeling and Model Driven Development.

Woo-Jin Lee received the M.S. and Ph.D. degrees in Computer Science from Korea Advanced Institute of Science and Technology (KAIST), Korea, in 1994 and 1999, respectively.

Dr. Lee joined the faculty of the Department of Computer Science at Kyungpook National University, Korea, in 2002. He is currently a professor in the School of Computer Science and Engineering at Kyungpook National University, Korea. He is interested in Software Testing, Software Modeling and Embedded Systems.

www.kci.go.kr