시스템 모델링을 위한 설계 프로세스 단계별 UML 활용 방안 연구

On System Modeling Framework for System Design Processes Using UML

장 재덕* 이 재천(아주대학교)
Jang, Jae-Deuck Lee, Jae-chun

Abstract The MBSE design method using UML has not been well exhibited throughout the overall system design processes. There has been many different modeling usages of UML for system requirements proposed by many till now. The extension of UML to system modeling must demonstrate consistent representation rule throughout the overall processes. This study proposes the method of using UML with overall consistency to implement system design processes. To validate and verify the proposed UML approaches, a case study of ATACMS Block IA missile system design was implemented and the resulting products were shown. Phases of the systems engineering design process, logical solution development, and physical solution development, were demonstrated

1. 서 론

현대에는 시스템 설계에서 소프트웨어 분야의 비중이 점차로 확대되어 가고 있고, 많은 부분이 하드웨어에서 소프트웨어로 대체됨에 따라 시스템 엔지니어링과 소프트웨어 엔지니어링의 통합 모델링 언어 및 도구는 필요성이 증대되고 있다. 통합 모델링 언어는 폭 넓은 정보를 통합하고 재사용할 수 있는 효과적이고 정확한 방법을 사용해야 하고, 시스템 요구사항과 설계 문제를 쉽게 식별하고 해결할 수 있어야 한다. 또한 다양한 이해당사자와 참가자들 사이에 정확하고, 일관된 의사소통이 될 수 있도록 표현이 정확하고, 사용이 편리해야 한다. 이러한 통합 모델링 언어에 요구되는 것을 고려하여 INCOSE(International Council on Systems Engineering)와 OMG(Object Management Group)는 복잡한 시스템의 분석, 규격서, 설계 그리고 검증을 지원하는 공동 목적 시스템 모델링 언어를 제공하기 위한 확장된 UML 개발을 위해 협업 중이다. 향후, UML은 소프트웨어와 하드웨어가 통합된 모델링 중심의 시스템 엔지니어링 실현을 전환하기 위한 핵심 매개가 될 것이다. 그러나 현재 많이 사용하고 있는 UML 1.3 버전은 소프트웨어를 개발하기 위한 객체지향 개발 방법을 중심으로 지원되는 언어이므로 이를 시스템 엔지니어링에 바로 적용하기에는 부족한 부분이 존재한다. 모델 기반 시스템 엔지니어링의 성공적인 구현을 위해 많은 모델링 기법이 사용되어 왔고, 소프트웨어와 시스템 통합의 필요성이 강조됨에 따라 UML을 표준 모델링 언어로써 채택하고 사용하기 위한 노력이 이루어져 왔다. 소프트웨어 개발에서 시스템 개발로 확장되는 UML이 시스템 엔지니어링 프로세스를 진행하는 가운데 어떻게 사용되어야 하고, 적용되어야 하는지를 살펴보도록 하겠다. 본 연구의 범위는 시스템 엔지니어링 설계 프로세스를 모델 기반으로 수행하는 방안으로서 UML 활용하여 프로세스 각 단계별 활동들에 대한 표현 기법과 표현 요소들의 일관성을 유지하는 방안을 ATACMS Block IA 유도탄 시스템에 개발에 적용하는 것이다.

이를 위하여 우선 시스템 엔지니어링 설계 프로세스와 프로세스 단계별 수행 절차를 분석한다. 시스템 엔지니어링 설계 프로세스는 시스템 엔지니어링의 대표적인 표준을 모델 기반으로 수행할 수 있도록 조정하여 설정한다. 조정한 시스템 설계 프로세스는 논리적 해결방안 개발, 물리적 해결방안 개발의 단계별 활동을 기본 업무로 설정한다.

시스템 설계 프로세스가 설정되면 각 단계별 활동을 수행할 수 있는 UML 도구 활용법과 표현법에 대해 개발하여야 한다. UML 활용 방안이 설정되면 이를 사용하여 시스템 설계 개발 사례를 설계 프로세스와 UML을 활용하여 구현한다. 사례로 구현되는 유도탄 시스템을 개발하기 위한 스키마 개발과 설계를 진행하고, 시스템이 설계되는 방법과 시스템의 특성에 대해서 논의한다.

2. 시스템 엔지니어링 설계 프로세스

MIL-STD 499B는 고객의 필요성과 요구사항들을 하나의 시스템 제품과 프로세스로 변환하기 위하여 결정권자에게 필요한 정보를 생성하며, 다음 개발 단계로 진입하기 위해 입력 자료를 제공하는 프로세스를 보여 준다. 따라서 프로세스는 한번에 한 단계씩 순차적으로 적용되고 각각의 개발 단계에 따라 부가적인 항목과 정의를 추가토록 한다. 시스템 엔지니어링 프로세스 모델은 입력과 출력물, 요구사항 분석, 기능 분석 및 할당, 요구사항 루프, 합성, 설계 루프, 검증, 시스템분석 및 통제와 같은 항목을 포함하고 있다. 본 논문에서는 시스템 엔지니어링의 요구사항 및 아키텍처 정의 단계로 표현되어 있는 시스템 엔지니어링 설계 프로세스에 중점을 두었다. 시스템 엔지니어링 설계 프로세스는 요구사항 분석, 기능 분석, 조합 그리고 시스템 분석 및 최적화의 4단계로 나누어지는데 반복적인 과정을 거침으로서 문제를 정의하고 해법을 개발해 나가는 과정이다. 시스템 엔지니어링 설계 프로세스는 4단계가 상호보완적으로 작용하여 최초

요구사항으로부터 하부시스템의 설계에 이용될 수 있는 규격서를 최종 산출물로 생성하는데 중요한 의미를 가진다. 이 4단계의 프로세스 중에서 논리적 해결방안 및 물리적 해별방안에 대한 UML 활용 방안에 대해 연구 하였다.

3. 시스템 엔지니어링 설계 프로세스 단계별 UML 활용 방안

가. 논리적 해결방안 단계에서의 UML활용

ATACMS Block IA의 임무는 ATACMS Block IA는 사정거리가 70km에서 300km내에 방호가 취약한 정지상태에 있는 소프트 표적(C2 Centers, 군수 지원 지역, 단거리 탄도 유도탄, 전방지역 무장/연료 재보급소, 헬리콥터 이/착륙 지역, 방공 지역(방공포명), 레이더 기지)을 파괴하거나 무력화 시킨다.

유도탄 시스템 기능 요구사항을 정의하기 위해 유도탄 시스템 정황에서 도출한 시스템 임무 프로파일에 대해 각 단계별 기능 시나리오를 작성하고 통합된 유도탄 시스템의 시나리오를 도출하였다. UML에서의 시스템 시나리오에 대한 표현은 액티비티 다이어그램(Activity Diagram)을 활용하여 구체화될 필요성이 있다. 일반적으로 액티비티 다이어그램은 이벤트의 흐름을 모델링하는 방식 중 하나로서 이벤트의 흐름에 따른 객체의 활동을 중심으로 표현한다. 액티비티 다이어그램을 활용한 시스템의 임무 및 시나리오를 정의 하였다. 그림 1은 시스템의 정황을 표현한 액티비티 다이어그램이다.

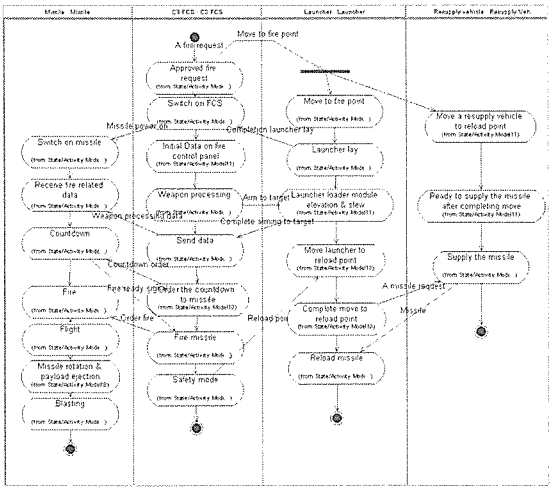

그림 1. 시스템의 정황(ATACM의 거동 모델)

그림 1과 같이 액티비티 다이어그램으로 구체화함으로써 구체화 된 시스템의 정황을 확인할 수 있다. 유도탄 시스템의 외부 시스템(External System)은 유도탄을 발사 시키고, 표적을 향해 방향을 잡아 주는 발사대(Launch), 사격 지휘체계와 사격 통제 장치를 자동으로 제어하는 FCS(Fire Control System), 사격에 필요한 군수지원 임무와 탄약을 재보급하는 재보급 차량(Resupply Vehicle)로 구성되어 있음을 알 수 있다. 이러한 시스템과 외부 시스템이 표현된 시스템 정황에서 시스템의 임무 프로파일을 이해할 수 있는 것이다. 시스템의 임무 프로파일은 다음 그림 2와 같다.

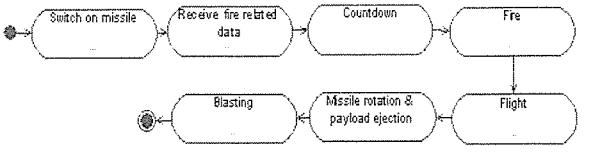

그림 2. 유도탄 시스템의 임무 프로파일 액티비티 다이어그램

시스템의 임무 프로파일은 각 단계마다 시스템 거동 다이어그램으로 표현되어 통합된 다이어그램으로 나타낼 수 있는데 여기에서 시스템의 하부 구성요소들이 등장하게 된다. 그림 3은 통합 시나리오 도출 과정을 보여 준다. 유도탄의 통합 시나리오는 액티비티 다이어그램으로 작성 하였다. 유도탄의 각 기능별 시나리오와 통합 시나리오를 살펴보면 유도탄 시스템의 하부 시스템 기능인 유도 조종부(Guidance and Control Section), 조정부(Control Section), 추진부(Propulsion Section), 탄두부(Warhead Assembly) 기능이 식별됨을 알 수 있다.

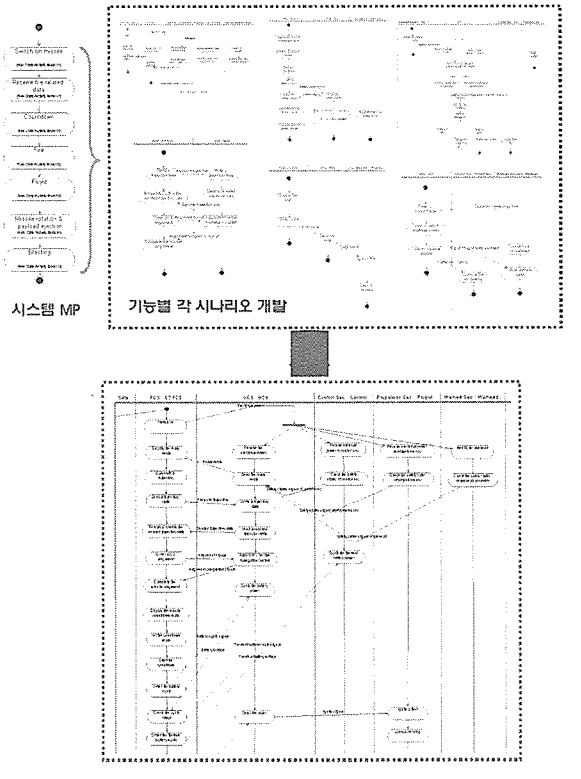

그림 3 시스템 통합 시나리오 개발

유도탄의 기능 인터페이스는 유도탄의 하부 시스템 사이에 주고 받는 아이템과 데이터를 정의하는 과정이다. 이미 유도탄 시스템의 시나리오를 정의함과 동시에 유도탄의 기능적 인터페이스를 정의 하였다. 그러나 UML 다이어그램 중 협력 다이어그램을 활용하면 각 하부 시스템을 중심으로 이들이 수행하는 기능과 아이템 및 데이터가 무엇인지를 보다 잘 식별 할 수 있다. 그림 4는 유도탄의 기능 인터페이스를 협력 다이어그램으로 표현한 것이다.

그림 4에 표현 된 것처럼 각 하부 시스템을 표현하는

협력 다이어그램의 객체/클래스의 수행하는 기능은 객체 상위에 재귀 링크로 (Link to Self) 표현 되었고, 클래스/객체 사이의 정보의 흐름을 보여주는 링크가 표현되어 있다. 순서에 따라 기능과 아이템 및 데이터에 숫자가 표현 되어 있다.

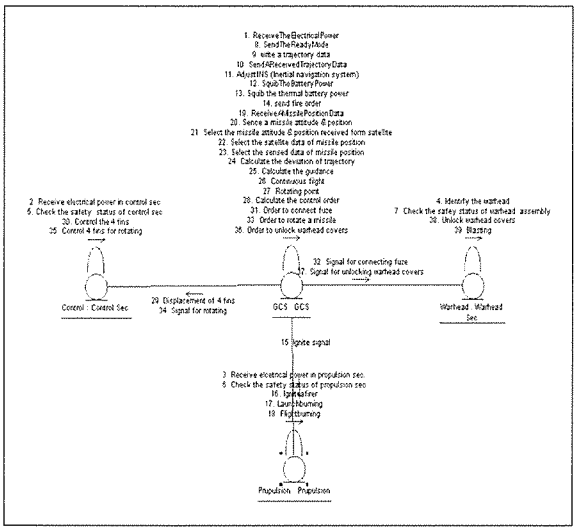

그림 4 유도탄 기능 인터페이스 정의 협력 다이어그램

유도탄의 기능 요구사항이 정의 되면 성능 요구사항이 기능에 할당 된다. 유도탄의 성능 요구사항인 "최대사거리 비행시 유도탄의 가속도"에 대한 성능 요구사항은 하부 시스템인 추진부의 성능 요구사항으로 할당 될 "추진부 전압", "추진부 발화물 점화 속도", "발사 연소 속도", "비행 연소 속도"의 성능 요구사항으로 분해 하였다. 이 성능 요구사항들은 추진부의 기능에 할당 된다. 그림 5는 추진부 기능 요구사항에 할당된 성능 요구사항을 유스케이스 다이어그램으로 표현한 것이다.

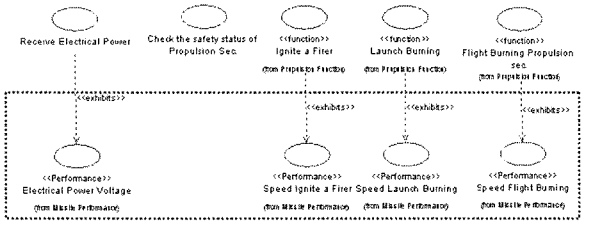

그림 5 기능 요구사항에 할당된 성능 요구사항

유도탄의 시스템 거동에 대한 시간선 분석을 실행 하였다. 유도탄 시스템의 기능 사이의 연속적인 관계와 기능의 반응 시간을 확인하고 거동의 일관성을 확인하기 위한 작업이다. 앞에서 표현한 유도탄의 협동 다이어그램은 시간선 분석이 가능한 시퀀스 다이어그램으로 자동 전환이 가능하다. 그림 6은 유도탄 시스템 거동의 시간선 분석을 위한 시퀀스 다이어그램이다.

시퀀스 다이어그램은 위에서 아래로 시간의 흐름에 따라 각 하부 시스템이 수행하는 기능 및 메시지 순서를 확인 할 수 있다.

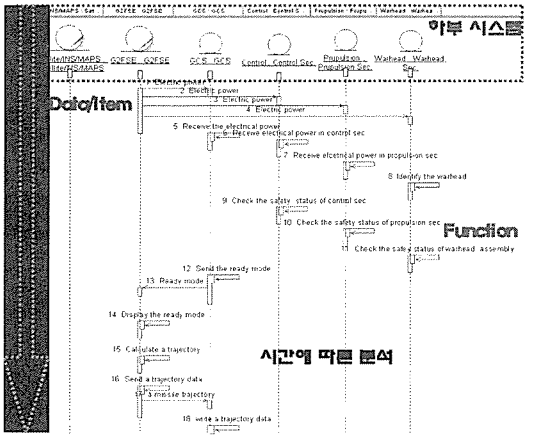

그림 6 유도탄 시스템 거동의 시간선 분석

이렇게 시간선 분석을 통해 논리적인 시스템 아키텍처를 구현하고 아래 그림 7과 같이 기능 아키텍처의 개층 구조를 유스케이스 다이어그램으로 표현하였다.

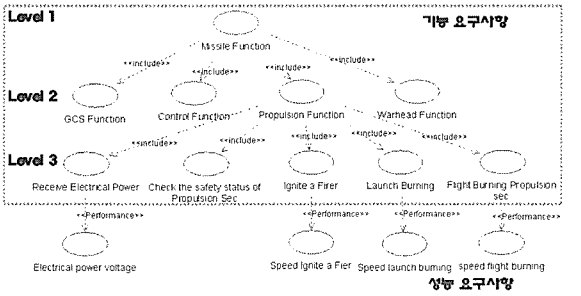

그림 7 유도탄의 기능 아키텍처

라. 물리적 해결방안 단계에서의 UML 활용방안

유도탄 시스템의 하부 시스템을 정의한다. 기능 아키텍처에서 정의된 기능적 분류를 통하여 기능 요구사항에 대한 시스템 구성요소에 대한 대안을 식별할 수 있다. 그러나 본 논문의 유도탄 시스템의 하부 시스템을 유도무기 시스템의 기능 아키텍처와 대응되는 유도 조종부, 조종부, 추진부 탄두부로 표현하였다. 앞에서 설명한 UML 활용법에서 하부 시스템을 표현하는 방안을 컴퍼넌트 다이어그램과 클래스 다이어그램의 두 가지로 설명 하였다. 아래 그림 8은 컴퍼넌트 다이어그램으로 유도탄의 하부 시스템을 표현한 것이고, 그림 9는 클래스 다이어그램으로 표현한 것이다.

컴퍼넌트 다이어그램으로 표현한 유도탄의 하부 시스템에 기능을 할당하는 것은 하부 시스템에 기능 및 속성(메시지)을 정의한 클래스를 유도탄의 각 하부 시스템에 매핑하는 것으로 표현 할 수 있다. 이는 향후 생성되는 소프트웨어 코드를 자동으로 생성할 수 있는 장점을 활용하기 위해서 이다.

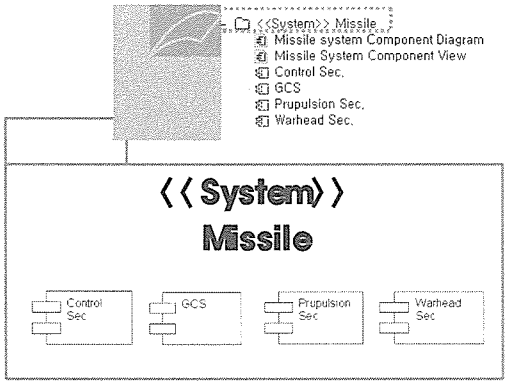

그림 8 컴퍼넌트 다이어그램으로 표현한 유도탄 하부 시스템

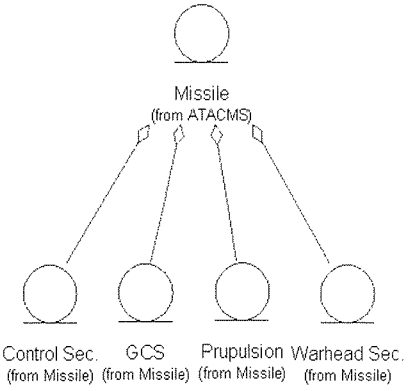

그림 9 클래스 다이어그램으로 표현한 유도탄 하부 시스템

클래스 매핑은 컴퍼넌트 세부항목의 실체화 탭에서 할 수 있다. 그림 10은 클래스를 컴퍼넌트에 매핑하는 세부항목이다. 그림 10에서 확인 할 수 있듯이 유도 조정부 컴퍼넌트에는 유도조정부 클래스가 매핑 되었고, 추진부 컴퍼넌트에는 추진부 클래스가 매핑 되었다.

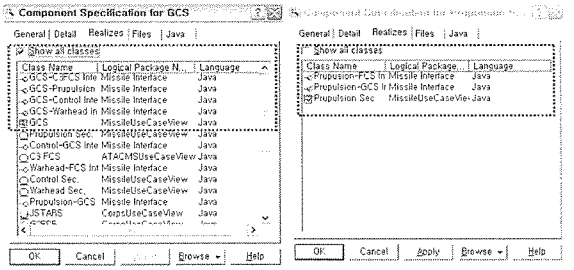

그림 10 컴퍼넌트에 클래스 매핑

클래스 다이어그램 다이어그램으로 표현한 유도탄의 하부 시스템에 대한 기능 할당은 기능으로 표현된 유스케이스와 하부 시스템 사이의 관계를 정의하는 작업이다. 아래 그림 11은 클래스 다이어그램을 활용한 하부 시스템에 대한 기능 요구사항 할당을 나타낸다.

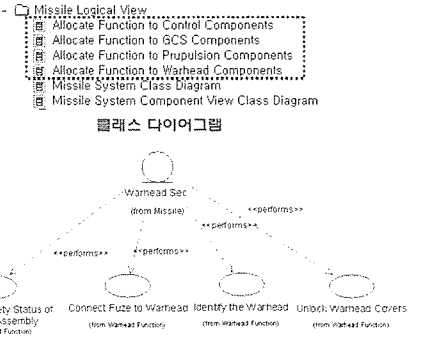

그림 11 클래스 다이어그램을 활용한 기능 할당

그림 11에서는 유도탄의 하부 시스템인 탄두부는 전원을 공급 받으면 탄두부를 식별하는 기능, 탄두 조립품 안전 상태를 점검하는 기능, 탄두에 휴즈를 연결하는 기능, 탄두 덮개를 개방하는 기능이 탄두부에 할당되어 수행 되고 있음을 표현한다. 하부 시스템에 제약 요구사항을 할당하는 과정 또한 두 가지로 표현 될 수 있다. 컴퍼넌트 다이어그램을 활용할 경우에는 노트 및 컴퍼넌트의 상세 항목에 기입하는 방법이 있다. 그러나 이 방안은 요구사항의 추적성을 확보하고 UML 표현법의 일관성을 유지하는데 문제점이 있다. 클래스 다이어그램에 표현한 하부 시스템에 할당 되는 제약 요구사항은 유스케이스로 표현되고 유스케이스는 하부 시스템에 할당된다. 이는 클래스 다이어그램에서 추적성을 계속적으로 유지할 수 있고, UML 표현법의 일관성을 유지할 수 있는 방안이다. 그림 12는 클래스 다이어그램을 활용한 제약 요구사항의 할당이다.

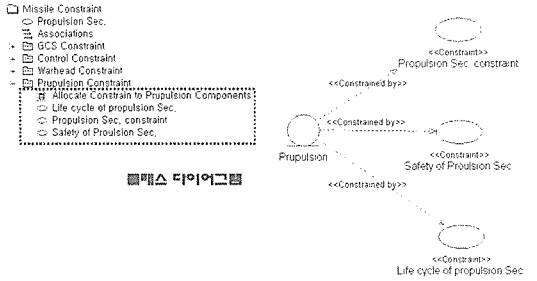

그림 12 클래스 다이어그램을 활용한 제약 요구사항 할당

그림 12의 예제에서 보듯이 추진부의 제약 요구사항은 추진부 구성, 추진부 안전, 추진부 순환 주기에 대한 제약 요구사항이 존재함을 알 수 있다. 하부 시스템에 대한 물리적 인터페이스를 정의 해주는 단계에서도 마찬가지로 컴퍼넌트 다이어그램과 클래스 다이어그램으로 나타낼 수 있다. 이 두 가지 방안은 표기법의 차이만 있을 뿐 거의 유사한 표현을 하고 있다. 클래스 다이어그램으로 표현할 때는 하부 시스템의 기능과 인터페이스로 운영되는 기능을 비교해서 식별 할 수 있는 장점이 있다. 다음 그림 13과 14는 컴퍼넌트 다이어그램과 클래스 다이어그램을 활용한 인터페이스 표기법이다.

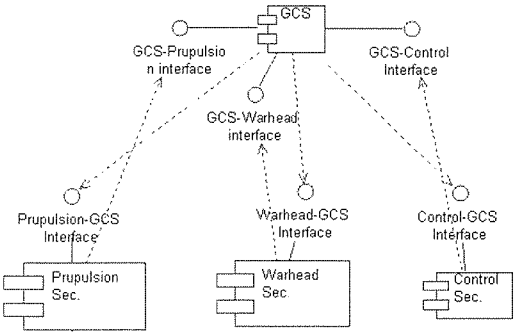

그림 13 컴퍼넌트 다이어그램을 활용한 인터페이스 정의

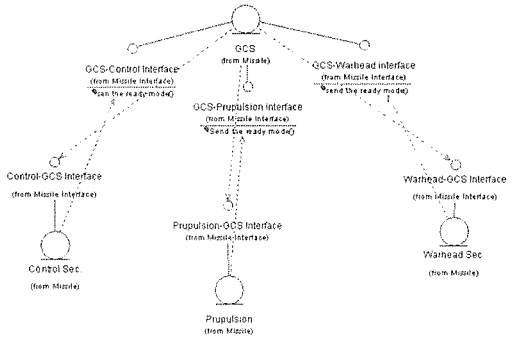

그림 14 클래스 다이어그램을 활용한 인터페이스 정의

조종부, 추진부, 탄두부는 유도 조정부와 입출력을 위한 인터페이스가 하나씩 존재하며 유도 조정부는 다른 하부 시스템과의 입출력을 위해 3가지의 인터페이스가 존재함을 알 수 있다. 클래스 다이어그램에 표현된 인터페이스와 하부 시스템을 비교하면 입출력을 위한 기능을 확인할 수 있다. 그림 15는 하부 시스템의 기능과 인터페이스를 통한 기능 수행의 관계를 표현한 클래스 다이어그램이다. 그림 15에서 확인 하듯이 추진부는 추진부의 안전 상태를 확인해서 유도 조정부에 신호를 보내야 한다. 따라서 출력에 대한 인터페이스에 안전 상태를 전송하는 기능을 표현하였다. 물리적 인터페이스가 설정되면 시스템의 물리적 아키텍처가 완성된다.

유도탄의 물리적 아키텍처의 표현은 인터페이스가 정의 되어 있고 기능 및 제약 요구사항이 할당된 아키텍처를 형성하게 된다.

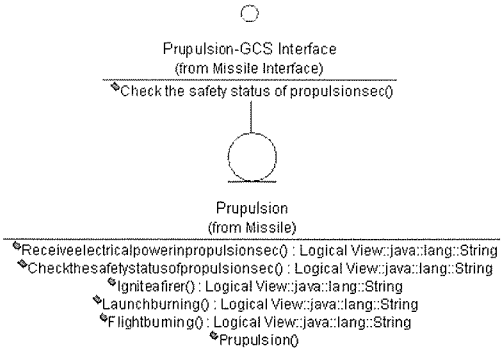

그림 15 인터페이스의 수행 기능 확인

3. 맺음말

본 논문에서는 UML을 사용하여 모델기반 시스템 엔지니어링을 구현할 때 활용되는 표기법 및 UML 요소들의 개념이 검증 될 수 있도록 ATACMS Block IA를 예제로서 구현하였다. 이 논문을 통해 UML의 효과적인 활용 방안을 제시하고자 하였으며, 적극적인 활용 방안을 모색하고자 하였다. 또한 UML 요소들이 소프트웨어 모델링에서 사용되는 기본 방침을 지키면서 시스템 모델링으로의 확장을 위한 방안을 제시함으로써 소프트웨어 단계까지의 일관성을 유지하는 것에 주의를 기울였다. 물론 UML이 시스템 엔지니어링 통합 모델링 언어로서 표준화가 되어 있지 못한 것이 사실이지만 현재 이러한 연구가 지속적으로 수행되고 있으며, 앞으로는 시스템 엔지니어링에 적합한 언어로서 표준화 되고 정착화 될 것이다. 현재 가장 많이 활용하고 있는 UML1.3 버전이 다소 동적인 거동을 표현하는데 어려움이 있고, 요구사항 사이의 추적성을 확보하는데 불편한 점이 존재하는 것은 사실이다. 하지만 모든 모델링 언어와 도구는 엔지니어의 표현 역량에 따라 그 효과성이 달라지는 것은 자명한 사실이다. 따라서 소프트웨어 모델링의 강점을 가진 UML을 보완하고 그 요소들의 표기법을 명확하게 한다면 시스템 엔지니어링 프로세스 단계별 구현을 위한 모델링 도구로서 전혀 손색이 없다.

참고문헌

[1] Army Modernization Plan, A-47, 2002

[2] 김현남, UML 2.0, Youngjin.com Y, Korea, pp.315, 2004

[3] Jon Holt., UML for Systems Engineering : Watching the Wheels. The Institution of Electrical Engineers. London, United Kingdom. pp. 223. 2001.

[4] Dennis M. Buede., The Engineering Design of Systems : Models and Methods. John Wiley & Sons, INC. Canada. pp.137

[5] Craig Larman., Applying UML and Patterns An Introduction to Object-Oriented Analysis and Design and The Unified Process. Prentice Hall PTR. PP. 58. 2002.