3

무기체계 임베디드 소프트웨어 개발 프로세스

국방대학교 ▌윤희병*
국방과학연구소 ▌오행록 · 조병인

1. 서 론

최근 그 중요성이 증대되고 있는 무기체계 임베디드 소프트웨어를 효율적으로 획득·관리하고 개발하기 위해서는 소프트웨어의 재사용과 무기체계간 상호효용성을 확보할 수 있는 개발 프로세스가 요구되고 있으나, 각 군 및 연구기관에서는 무기체계 임베디드 소프트웨어 개발을 위해 대부분 구조적 개발방법론을 사용하고 있으며, 일부 무기체계에서 객체지향 개발방법론을 도입하는 중이다.

국방 분야의 자동화 정보체계 소프트웨어는 1995년 국방과학연구소에서 발표한 국방 소프트웨어 개발방법론인 국방 컴포넌트 기반(CBD: Component Based Development) 방법론인 '국방 CBD 방법론(AddMe)'[1]을 표준 절차로 준수하도록 하고 있으나, 무기체계 임베디드 소프트웨어 개발방법론에 대해서는 별도로 규정되어 있지 않아 산출물의 재사용이나 상호운용성 증진에 어려움이 있었다.

소프트웨어 재사용을 고려하고 무기체계 간 상호운용성을 확보하기 위해서는 기존의 '국방 CBD 방법론'을 기반으로 무기체계 임베디드 소프트웨어에 적용할 수 있는 '무기체계 임베디드 소프트웨어 개발 프로세스'가 필요하다. 국방 CBD 기반 임베디드 소프트웨어 개발 프로세스는 이해하기 쉬운 단계로 구성되어야 하며, 일반적인 소프트웨어 개발 프로세스 활동을 따르면서 임베디드 소프트웨어 개발절차의 모든 단계를 이행하고 위험요소를 최소화하여 무기체계의 특성 및 요구사항이 정확하게 반영된 개발 프로세스가 되어야 한다.

따라서 본 논문에서는 먼저 컴포넌트 기반 소프트웨어 개요 및 개발과정, 국내·외 CBD 기반 개발 프로세스 비교 분석 그리고 제품계열 기반 개발 프로세스에 대한 비교 분석 등을 실시한다. 그런 다음 무기체계 임베디드 소프트웨어에 대한 개념 및 특징을 연구하고, 마지막으로 본 연구의 핵심분야인 CBD 기반 무기체계 임베디드 소프트웨어 개발 프로세스 설계시 고려요소와 세부 설계절차(기본, 단계, 활동 설계를 제시하며 더 세부적인 작업 설계는 제외함)를 제시한다.

이러한 CBD 기반 무기체계 임베디드 소프트웨어 개발 프로세스로 개발된 산출물을 재사용함으로써 개발비용과 개발기간을 줄일 수 있어 높은 품질의 무기체계 임베디드 소프트웨어를 획득할 수 있는 장점이 있을 것이다.

2. 컴포넌트 기반 소프트웨어 개요 및 개발 프로세스

2.1 CBD 방법론 개요

컴포넌트란 특정한 기능을 수행하기 위해 독립적으로 개발, 보급되고 잘 정의된 인터페이스를 가지며, 다른 부품과 조립되어 애플리케이션 시스템을 구축하기 위해 사용되는 소프트웨어 단위 또는 부품을 말한다. 이러한 컴포넌트 개념을 이용한 CBD 방법론이 등장하게 된 배경은 크게 다음과 같은 네 가지로 요약할 수 있다.

첫째는 언제, 어디서, 누구나 필요한 정보를 쉽게 얻을 수 있는 인터넷 환경이 보편화되어 시시각각 변화하는 사용자의 요구를 여러 플랫폼 상에서 분산된 객체 및 모듈을 이용하여 구현이 가능해졌고, 둘째는 소프트웨어의 Plug and Play 기술의 발전으로 소프트웨어 개발시 컴포넌트를 부품 조립하듯이 통합하여 생산이 가능해졌기 때문이다. 셋째는 정보기술 및 이기종 컴퓨터 간의 연동기술이 발전되어 인터넷상의 다양한 소프트웨어 부품이 개방화되었으며 시스템의 신속한 구축, 변경 확장의 용이성 및 재사용성 증진을 위해 기종에 상관없이 표준화된 소프트웨어 부품을 이용하는 경향이 증가하였고, 마지막으로 객체지향 패러다임을 기반으로 하는 재사용 기술의 발전과 소프트웨어 조립을 위한 컴포넌트의 개념이 등장하였기 때

* 종신회원

문이다[2].

컴포넌트 기반 개발(CBD)이란 개발 생명주기의 모든 활동, 즉 요구분석, 아키텍처, 설계, 시험, 배포, 기술적 인프라 지원 및 프로젝트 관리 등이 컴포넌트를 기반으로 이루어지는 소프트웨어 개발 방법을 말한다. 컴포넌트 기반 개발방법은 크게 컴포넌트 개발(CD: Component Development)과 컴포넌트 기반 소프트웨어 개발(CBSD: Component Based Software Development)로 구분할 수 있다. 컴포넌트 개발은 여러 애플리케이션에 공통으로 사용되는 소프트웨어 컴포넌트를 확보하기 위해 컴포넌트를 개발하는 것이며, 컴포넌트 기반 소프트웨어 개발은 CD에서 개발한 컴포넌트를 조립하여 소프트웨어를 개발하는 것이다.

2.2 CD와 CBSD 개발절차

CD는 완전한 소프트웨어 시스템을 만드는 것이 아니라 다른 소프트웨어 시스템에 포함될 컴포넌트를 만드는 것이다. 컴포넌트 개발방법에는 변경, 래핑, 생성의 세 가지 방법이 있다. 변경은 기존의 확보된 컴포넌트의 기능을 확장하는 방법이며, 래핑은 기존의 레거시 소프트웨어를 인터페이스 명세 언어(XML 등)를 이용하여 컴포넌트를 만드는 방법이고, 생성은 필요한 컴포넌트가 존재하지 않을 경우 필요한 컴포넌트를 새롭게 만드는 방법이다. 이러한 CD의 일반적 절차는 도메인 분석 → 도메인 설계 → 컴포넌트 추출 → 컴포넌트 설계 → 컴포넌트 구현 → 컴포넌트 인증으로 이루어지며 그림 1과 같다.

CBSD는 여러 애플리케이션에 공통적으로 사용되는 소프트웨어 컴포넌트를 획득, 재사용하여 소프트웨어를 개발하는 방법으로 CBSD를 개발하는 일반적인 절차는 요구분석 / 컴포넌트 획득 → 컴포넌트 기반 설계 → 컴포넌트 조립 → 시험 단계로 이루어지며 그림 2와 같다.

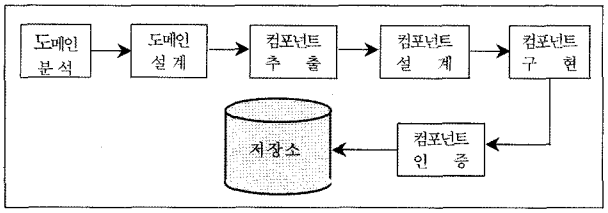

그림 1 CD 개발절차

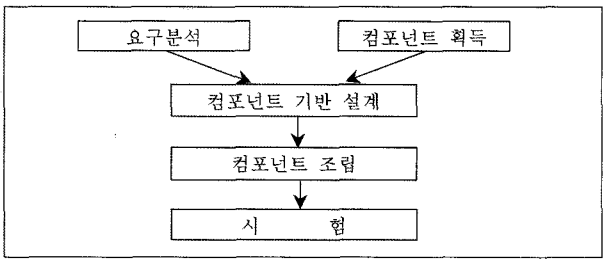

그림 2 CBSD 개발절차

2.3 CBD 방법론의 종류

CBD 방법론은 크게 국내에서 개발한 방법론과 국외에서 도입한 방법론의 두 가지로 구분할 수 있다.

국내 개발 CBD 방법론은 연구소 주도로 개발한 방법론으로 국방 정보체계를 대상으로 하는 국방 CBD 방법론과 ETRI에서 주관하여 개발한 마르미-III[3]가 있다. 국방 CBD 방법론은 ADD에서 개발하여 2005년 국방 표준으로 배포한 국방 정보체계 CBD 방법론이며, 마르미-III는 ETRI 주관으로 학계 및 산업계가 협력하여 개발한 방법론으로 아키텍처 기반의 반복, 점진적인 개발방법론이다.

국외 CBD 방법론은 많은 종류가 있으나 본 논문에서는 CBD 방법론의 기본이 되는 RUP와 학계 주도로 개발한 방법론으로 CBD 기반 방법론의 바이블처럼 연구되는 Catalysis, 그리고 임베디드 소프트웨어의 컴포넌트 기반 개발을 위해 유럽에서 제안된 DESS에 대하여 분석한다.

RUP[4]는 1993년 Jacobson의 OOSE방법론의 개념을 확장한 컴포넌트 기반 개발방법론으로 산업분야의 목적에 부합하는 특성을 갖는 아키텍처 중심 개발방법론이다. Catalysis[5]는 주로 학계에서 연구대상으로 채택한 방법론으로 CBD 방법론의 기초개념과 기법을 연구하는 교범처럼 사용이 되며 다른 방법론에 영향을 많이 주고 있으나 이론적인 부분이 많아 실용화되지는 못한 방법론이다. DESS[6]는 실시간 임베디드 소프트웨어 개발을 위해 벨기에, 프랑스, 체코 등 6개국이 공동 개발한 컴포넌트 기반 임베디드 소프트웨어 개발방법론으로 V모델과 반복, 점진적인 개발방법을 채택하고 있다.

3. 국내 · 외 CBD 방법론 비교 분석

컴포넌트 기반 개발방법론은 현재 많은 종류가 있으나 본 장에서는 국내에서 개발한 마르미-III와 국방 CBD 방법론을, 국외는 DESS, RUP, Catalysis에 대해 표기법, 개발절차, 테스트 절차에 대한 비교 분석결과를 제시한다.

3.1 표기법 비교 분석

국방 CBD 방법론은 개발절차를 IDEF0 표기법을 이용하여 작성, 이해하기 쉽게 상세히 설명되어 있으며

비즈니스 분석을 위한 모델링 기법으로 UML을 적용하고 있다. 활동 간의 계층적 관계를 계층도로 표현하고 있으며 복잡한 활동을 보다 작고, 단순하며, 상세한 활동으로 표현하기 위해 분해도를 사용하고 있다. 또한 분석하고자 하는 업무는 배경도(Context Diagram)로 해당 업무와 외부 인터페이스를 표현하여 전체의 주제를 표현하고 있다[1].

마르미-III는 UML을 기본 표기법으로 사용하고 있으며 개발절차를 서술식으로 상세히 표기하여 기술적 표기법을 모르는 초보자도 쉽게 이해가 가능하다. 프로그램 평가 검토 기법(PERT)으로 각 단계마다 점검 목록을 제시하고 있으며, 계층도를 이용하여 각 활동과 작업 간의 관계를 표현하고 있다[3].

DESS는 UML을 기본 표기법으로 사용하고 있으며 DESS 프로젝트의 배경도를 포함하여 명세, 개발, 컴포넌트 문서는 UML을 이용하여 표기하고 있다. 개념적인 소프트웨어 프로세스는 변형된 DFD(Data Flow Diagram)를 사용하고 있으며 개발절차는 변형된 UML 표기법과 서술식을 병행 사용하여 이해도를 높이고 있다[6].

RUP는 UML을 기본 표기법으로 사용하고 있으며 Catalysis도 UML을 기본 표기법을 사용하고 있으나 모델링의 부가 설명을 위해 심볼과 표기 형태를 다소 변경하였다[4].

3.2 개발절차 비교 분석

국방 CBD 방법론의 개발절차는 총 4단계, 12활동, 37작업, 41산출물로 구성된다. 단계는 분석 → 설계 → 구현 및 테스트 → 인도의 순으로 진행되며 분석 단계는 요구사항 정의, 아키텍처 정의, 요구사항 분석의 세 가지 활동, 설계는 개략 설계 및 상세 설계의 두 가지 활동, 구현 및 테스트는 테스트 준비, 구현, 통합 테스트, 시스템 테스트의 네 가지 활동, 그리고 인도는 시스템 설치 및 인수 지원이라는 두 가지 활동으로 구성된다. 국방 CBD 방법론은 정보체계 개발방법론으로 하드웨어 개발과 관련한 작업이 없으며, 시스템과 관련된 작업은 도메인 모델링, 현행 시스템 분석, 시스템 아키텍처 정의 작업으로 구성된다. 구현 및 테스트 단계에서 시스템을 구현하며 작업의 흐름은 DB 구축 → 컴포넌트 구현 및 테스트 → 사용자 인터페이스 구현이며 컴포넌트 구현시 .NET과 J2EE를 지원한다[1].

마르미-III의 구조는 단계 → 활동 → 작업 → 절차이며 소프트웨어 개발을 위한 개발절차와 개발을 지원하고 관리하기 위한 관리절차로 구성된다. 소프트웨어 개발절차는 요구획득 → 아키텍처 → 점진적 개발 → 인도로 구성되고 총 4단계, 20활동, 75작업, 95산출물이 있다. 그리고 매 단계마다 점검 사항을 두어 체계적이고 효율적인 프로젝트 관리를 지원한다. 시스템 관련 작업은 요구사항 획득과 아키텍처 단계에서 수행하나 하드웨어 동시 설계 및 통합 작업을 지원하지 않는다. 컴포넌트 구현작업은 .NET과 J2EE를 지원하며 프로세스 전 단계에 걸쳐 컴포넌트 재사용 개념이 잘 정의되어 있다[3].

DESS 개발방법론은 평행적으로 작용하는 3개의 V 워크플로우, 즉 실행 워크플로우, V&V 워크플로우, 요구관리 워크플로우, 23단계, 32산출물로 구성된다. DESS는 실시간 임베디드 소프트웨어 개발방법론으로 요구공학 단계에서 하드웨어를 고려하고, 시스템 설계 이후 하드웨어 개발을 소프트웨어 개발과 병행한다. 하드웨어의 개발은 소프트웨어 통합에 맞추어 종료하며 시스템 통합시 하드웨어 통합을 병행한다[6].

RUP는 5활동, 25작업, 34산출물로 구성되며, 핵심 프로세스 워크플로우 활동은 비즈니스 모델링 → 요구사항 → 분석 및 설계 → 구현 → 테스트로 구성된다. 각 단계는 수평축의 동적구조에 따라 작업의 반복을 통해 점진적으로 개발되며 각 작업별 입력물과 산출물의 정의가 명확하지 않다. 그리고 RUP는 정보체계 개발방법론으로 하드웨어 동시 개발의 개념을 포함한 시스템 요구사항 분석 및 시스템 설계가 고려되지 않았으며 시스템 배경도 이해, 하부 시스템 설계 및 구현은 하드웨어가 포함된 시스템이 아닌 소프트웨어 자체 시스템으로 하드웨어와 소프트웨어의 통합 및 테스트 절차가 없다[4].

Catalysis는 개발절차가 단계-활동-작업으로 명확히 구분되어 제시되어 있지 않으며, 개발절차는 "요구사항, 시스템 명세, 아키텍처 설계, 구축/통합/테스트" 4단계로 구분할 수 있다. 각 단계를 지원하는 활동에 대한 세부 작업이 명확히 제시되지 않으며 시스템 명세 단계에서 하드웨어와 소프트웨어의 영역에 대한 고려는 있지만 이후 단계는 하드웨어 설계 및 동시 개발에 관한 절차가 없다. 또한 구현 및 테스트 단계에서의 세부 작업 및 지침이 없으며 소프트웨어 구축을 위한 활동은 비선형적, 반복적, 병행적으로 수행된다[5].

3.3 테스트 절차 비교 분석

국방 CBD 방법론의 테스트 단계는 총 5활동, 8작업, 7산출물로 구성되어 있으며, 테스트 프로세스 범위는 공학적 생명주기를 제공하고 하드웨어 통합 및 하드웨어와 소프트웨어 통합에 따른 테스트 지원 없이 소프트웨어 자체 테스트만 지원한다. 그리고 임베디드

소프트웨어 개발방법론의 일반적인 테스트 프로세스와 비교시 필드 테스트, 인증 테스트, 리그레션 테스트에 대한 지원이 없다. 컴포넌트 테스트 작업에는 하드웨어 컴포넌트 테스트 지원 사항이 없으며, 통합 테스트 작업에는 하드웨어 통합 테스트, 하드웨어/소프트웨어 통합 테스트 지원이 없다[1].

마르미-III의 테스트 단계는 총 5활동, 13작업, 13산출물로 구성되며, 테스트는 공학적 주기의 테스트인 컴포넌트 테스트, 통합 테스트, 시스템 테스트 및 인수 테스트만을 수행한다. 임베디드 소프트웨어 특징을 지원할 수 있는 하드웨어 통합 및 하드웨어와 소프트웨어 통합에 따른 테스트 그리고 필드 테스트, 인증 테스트, 리그레션 테스트에 대한 지원이 없다. 통합 테스트 작업은 소프트웨어 컴포넌트의 통합 테스트만을 지원하며, 하드웨어 컴포넌트의 통합 테스트와 하드웨어-소프트웨어 간의 통합 테스트에 대한 지원이 없다[3].

DESS에서 테스트는 V&V 워크플로우 V에서 지원하며 실행 워크플로우 V와 평행하게 진행된다. 테스트 활동은 전체 과정 중 2단계이며 7개 작업, 7개 산출물이 있다. V&V 워크플로우 V는 하드웨어와 소프트웨어가 동시에 개발되는 임베디드 시스템의 특징을 반영하여 하드웨어/소프트웨어 통합 테스트, 리그레션 테스트를 지원하나 하드웨어 통합 테스트, 필드 테스트는 지원하지 않는다. V&V 워크플로우 V에서 좌측은 벨리데이션(validation) 단계이며 우측은 베리피케이션(verification) 단계이다. DESS 개발방법론의 컴포넌트 개발 관련 활동은 실행 워크플로우 V에서 명세/설계 활동, 구현, 검증 활동으로 지원하고 V&V 워크플로우 V에서 컴포넌트 테스트 활동으로 지원한다[6].

RUP의 테스트 작업은 정적 구조의 테스트 단계에서 이루어지며 총 7개 작업, 7개 산출물로 구성된다. 하드웨어 테스트, 하드웨어 소프트웨어 통합테스트를 지원하지 않으며, 통합 테스트는 작업 반복시 마다 수행하고 시스템 테스트는 마지막 반복에서 1회만 수행한다. 도입 단계에서 테스트 계획을 수립하며, 정련 및 구축 단계에서 시스템 구현에 따른 통합 및 시스템 테스트에 초점을 두고, 전이 단계에서는 결함의 탐지 및 수정, 회귀 테스트에 주력한다[4].

Catalysis의 테스트 작업은 설계 단계와 구현/통합/테스트 단계에서 지원하며 테스트는 총 7개 작업과 8개 산출물로 구성된다. 비즈니스 모델링과 도메인 모델링 기반의 요구사항 명세서는 모든 테스트 작업의 기본 입력물이며 아키텍처 단계에서 컴포넌트 리뷰 작업을 수행한다. 테스트 아키텍처 설계시 하드웨어와 소프트웨어 영역을 고려 설계하며 시스템 테스트는 소프트웨어 통합 테스트만 언급하고 있으며, 하드웨어와 소프트웨어 통합 테스트에 관한 상세한 절차 및 지침은 없다[5].

4. 제품계열 방법론 비교 분석

본 장에서는 무기체계 임베디드 소프트웨어 개발 프로세스 중 컴포넌트 개발 프로세스를 위해 제품계열 방법론의 구성 및 프로세스, 개발의 다양성 및 핵심자산 측면에서 비교 분석한 결과를 제시한다.

4.1 개발방법론 구성 및 프로세스 비교 분석

개발방법론 구성을 먼저 비교하면, 마르미-EM을 제외한 대부분의 제품계열 개발방법론들은 핵심자산 개발과정의 영역공학과 제품개발 과정의 응용공학으로 구성된다[7]. 마르미-EM[8]은 6개의 개발 경로와 네 가지의 작업 지침으로 구성되며, FAST[9]는 영역 자격과 응용공학 환경, SEI SPL[10]은 관리, Bosch[11]는 관리와 핵심자산 저장소가 추가로 구성되어 있다.

개발 프로세스를 비교하면, 마르미-EM은 요구사항, 도메인, 아키텍처, 설계, 구현, 테스트의 6단계와 19개의 세부 활동으로 구성되고, FORM[12]은 2단계에 16개 활동, FAST는 3단계에 13개 활동, KobrA[13]는 2단계에 14개 활동, Bosch는 2단계에 6개 활동으로 구성되어 있으나, SEI SPL은 3개의 기본 활동만 제시하고 있다. 아키텍처, 통합, 테스트에 대한 구분을 살펴보면 마르미-EM은 아키텍처와 테스트, FORM과 SEI SPL 및 Bosch는 아키텍처, KobrA는 통합에 대해 별도로 구분하고 있다. 제품계열에 대한 반영은 마르미-EM이 도메인, FORM은 마케팅 및 제품계획(MPP), FAST는 영역 자격, KobrA는 배경 실체화, Bosch는 영역 분석에서 이루어지고 SEI SPL은 기본 활동에 전체적으로 반영된다. 각 방법론에서 임베디드SW 개발절차와 매핑이 안되는 부분을 보면 마르미-EM은 통합, FORM과 KobrA는 시험평가 및 인수 검증, Bosch는 평가 및 인수 검증, FAST는 통합에서 평가 및 인수 검증까지이며, SEI SPL은 전체적으로 매핑이 어렵다.

4.2 개발의 다양성 및 핵심자산 비교 분석

먼저 개발의 다양성 측면에서 비교 분석하면, 마르미-EM은 시스템 요구분석을 세분화하여 개발 대상의 특성에 따른 경로 설정이 가능하며, FORM과 FAST는 제품시장에 대한 사전 분석이나 영역 자격을 통한 경제적 측면 분석으로 비용 및 투자 요소 등의 시장 변

화에 쉽게 대응된다. 그리고 SEI SPL은 기본적이고 개념적인 절차 제시로 프로세스의 유연성과 융통성이 있으며, KobrA는 컴포넌트 기반으로 제품 개발의 체계적 접근을 통한 품질 향상이 가능하다. Bosch는 영역 분석과 아키텍처 설계에서 제품군의 기반과 확장성을 제공한다.

핵심자산 측면에서 비교 분석하면, 마르미-EM은 자산으로써 필요한 컴포넌트를 선정하고 관계를 설정하며, 구현 및 테스트 과정을 잘 명시하고 있다. FORM은 비기능적 품질요소에 영향있는 임베디드 시스템에서 품질요소를 유도하며, FAST는 영역 간 참조와 패밀리 변경을 통한 유지보수를 지원한다. SEI SPL은 폭넓은 지침의 제공 및 유지보수를 지원하며, KobrA는 자산의 가변성 표현 방법 제공 및 적용을, Bosch는 아키텍처에 대한 상세 지침을 제공한다.

5. 무기체계 임베디드 소프트웨어 특징

5.1 무기체계 임베디드 소프트웨어 개념

한국군의 무기체계 임베디드 소프트웨어에 대한 정의는 2002년 1월 국방부에서 제정한 '무기 및 비무기체계 임베디드 소프트웨어 개발관리 지침'[14]에 정의되어 있으나, 2006년 1월 1일 방위사업청 개청과 동시에 사용되지 않고 있다. 이 지침에 따르면 무기체계 임베디드 소프트웨어는 각종 무기 및 비무기체계에 내장되어 해당 장비의 임무에 전용으로 제공되는 소프트웨어라고 정의하고 있다.

미군의 경우에는 2000년 5월에 제정한 미 공군성 지침인 '소프트웨어-집약 시스템의 성공적인 획득 및 관리 지침서'[15]에 정의되어 있다. 이 지침서에 따르면 무기체계 임베디드 소프트웨어는 전체 시스템의 통합 컴포넌트로서 무기체계의 임무를 지원하거나 또는 그 임무에 치명적인 영향을 미치는 것으로 특별히 무기체계를 위해 설계되거나 또는 무기체계에 전용인 소프트웨어라고 정의되어 있다.

무기체계 소프트웨어에 대한 분류는 한국군의 경우 현재까지 없으나 2007년 3월부터 방위사업청에서 무기체계 임베디드 소프트웨어 정책연구과제를 통해 정립 중에 있다. 미군의 경우, 미공군성의 소프트웨어 기술 지원센터에서 제시한 자료에 소프트웨어 도메인을 무기체계 소프트웨어와 자동화 정보시스템 소프트웨어로 분류하고 있으며, 무기체계 소프트웨어는 임베디드 소프트웨어, C3 소프트웨어, 인텔리전스 소프트웨어, 기타 무기체계 소프트웨어로 분류하고 있다[15].

5.2 무기체계 임베디드 소프트웨어 특징

무기체계 임베디드 소프트웨어 특징을 도출하기 위해 다음 절차를 따르며 세부내용은 생략한다: ① 무기체계 특징 도출 → ② 분야별 임베디드 SW 특징 분석 → ③ 임베디드 SW 특징 도출→ ④ 무기체계 특징과 임베디드 소프트웨어 상관관계 분석 → ⑤ 무기체계 임베디드 SW 특징 도출.

도출된 무기체계 임베디드 소프트웨어 특징은 총 6개이며 실시간성, 테스트 어려움성, 고 신뢰성, 목적 한정성, 개발의 어려움성, 하드웨어 통합성 등이 있다.

먼저 실시간성은 무기체계 임베디드 소프트웨어가 경성 실시간성을 지원해야 하고 무기체계는 임무에 중요한 시스템으로 주어진 시간 내에 발생한 이벤트 처리를 완료해야 하고, 이벤트 처리 지연은 아군의 인명피해 및 전투력 손실의 원인이 된다는 특성이다.

둘째, 테스트 어려움성은 임베디드 소프트웨어의 테스트가 소프트웨어 단위 테스트 및 통합 테스트, 하드웨어 단위 테스트 및 통합 테스트, 소프트웨어-하드웨어 통합 테스트, 시스템 테스트, 인수/필드/인증/리그레션 테스트 등을 수행해야 한다는 특성이다.

셋째, 고 신뢰성이다. 이 특성은 무기체계 임베디드 소프트웨어의 고 신뢰성이 고가용성을 의미하고 긴박한 전장 속에서 소프트웨어의 고장은 무기체계의 전투력 상실을 의미한다는 특성이다. 또한 고 신뢰성을 만족하기 위해서는 사용자 간섭 없이 고장 예측, 회피, 감내, 제거가 가능하도록 설계되고 구축되어야 한다.

넷째, 목적 한정성은 무기체계 임베디드 소프트웨어가 특정 무기체계의 임무 목적에 맞게 개발되고 또한 범용성이 없다는 특성이다.

다섯째, 개발의 어려움성은 무기체계 임베디드 소프트웨어가 하드웨어와 동시에 설계 및 개발되므로 소프트웨어에 대한 전문지식뿐만 아니라 무기체계가 수행하는 임무와 관련된 도메인 지식과 하드웨어 특성에 대한 지식까지 필요하다는 특성이다.

여섯째, 하드웨어 통합성은 무기체계 임베디드 소프트웨어가 하드웨어와 동시에 설계 및 개발되고 통합되므로 하드웨어에 대한 전문지식과 소프트웨어에 대한 전문지식을 동시에 보유해야 한다는 특성이다.

6. 무기체계 임베디드 소프트웨어 개발 프로세스

6.1 설계 고려요소

무기체계 임베디드 소프트웨어의 개발 프로세스 설계를 위해 공통 부분, 단계 설계 그리고 활동 및 작업 설계 시로 구분하여 설명한다.

공통 고려요소로서, 개발자의 이해도 향상을 위해 IDEF0 표기법 사용, UML 다이어그램 지원, 서술식 표기법을 사용하며 개발조직 내 원활한 의사소통을 위한 용어 표준화를 지원한다. 또한 산출물로서 용어집을 작성한다.

단계 설계 시 고려요소로서, 먼저 임베디드 시스템 수명주기 적용을 위해 임베디드 소프트웨어의 특징을 반영하여 설계하며, 하드웨어에 의한 구현 기능과 소프트웨어에 의한 구현 기능을 명시하고 하드웨어와 소프트웨어의 동시 설계를 고려한다. 둘째로 프로세스 각 단계마다 점검사항을 두어 체계적이고 효율적인 프로젝트 관리 지원 및 각 작업에서의 오류를 최소화한다. 셋째로 방위사업청 규정과 지침을 준수하며 방위사업청 SW 개발 프로세스[16]의 절차를 준수한다.

활동 및 작업 설계 시 고려요소로서, 시스템, 하드웨어 및 소프트웨어 통합을 고려하고 임베디드 소프트웨어의 특성을 반영한 테스트 작업, 즉 HW/SW 통합 테스트, 필드 테스트, 인증 테스트 등을 지원한다. 국내·외 개발방법론 개발절차를 고려하며 재사용성을 고려한 컴포넌트 개발(CD)절차를 제시한다. 또한 위험요소 최소화 및 오류 최소화를 위한 점진적 개발을 도입하고 하드웨어에 의한 구현 기능과 소프트웨어에 의한 구현 기능을 명시한다.

6.2 개발 프로세스 개요

국방 CBD 기반 무기체계 임베디드 소프트웨어 개발 프로세스는 소프트웨어 개발 추세인 UML을 기반으로 하며 국방 무기체계(자동화 정보체계 제외) 도메인 내에서 재사용성을 높일 수 있도록 공통성 및 가변성을 정의하며, 임베디드 소프트웨어 생명주기 특징인 하드웨어와 소프트웨어 기능 결정과 동시 설계 기능을 반영한다.

또한 임베디드 시스템의 위험요소 최소화를 위해 프로토타이핑 기법, 점진적 개발을 적용한 하드웨어 프로토타이핑 단계를 제시하며, 각 활동의 작업마다 리뷰(Review) 또는 테스트 활동을 추가함으로써 피드백 기능을 제공하여 작업 결과를 검토할 수 있도록 한다.

그리고 아키텍처 정의를 통해 적합한 컴포넌트가 존재하는 지를 식별하여 재사용 컴포넌트를 반영하고 제품계열 개념의 반영을 통해 컴포넌트에 요구되는 공통성과 가변성을 분석하여 소프트웨어를 구현하므로 비즈니스 로직에 충실하면서 재사용성을 향상시키는 소프트웨어 개발 프로세스이다.

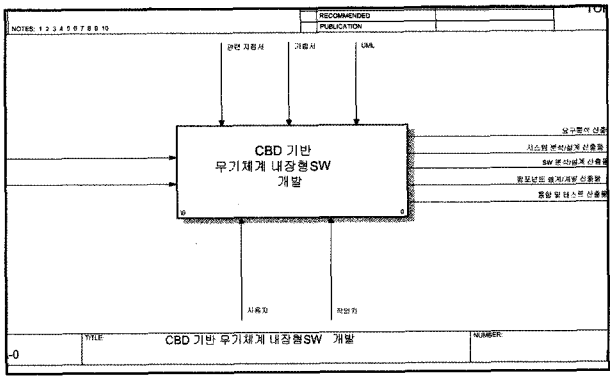

그림 3 기본 설계 구성도

6.3 개발 프로세스 설계

6.3.1 기본 설계

기본 설계에서는 IDEF0의 입력, 제어, 출력, 처리자원이라는 표기법을 이용하며 UML 2.0 기반의 각종 다이어그램을 사용하고 개발조직의 원활한 의사소통 지원을 위해 용어집을 작성한다. 이와 같은 기본 설계 구성도가 그림 3에 나타나 있다.

6.3.2 단계 설계

단계 설계에서는 임베디드 시스템 수명주기를 적용하여 하드웨어 개발단계를 제시하며 위험요소 최소화 및 오류 최소화를 위해 점진적 평가를 적용한다. 또한 방위사업청 규정과 지침을 준수하여 6단계의 개발 프로세스인 요구분석, 시스템 분석/설계, HW 프로토타입 개발, SW 분석/설계, 컴포넌트 설계/개발, 통합 및 테스트를 설계하며 그림 4에 단계 설계의 구성도가 나타나 있다. 이 그림에서 가운데 삼각형 모양의 적색 형태가 점진적 평가를 나타내며 사각형 모양의 박스는 6개의 단계를 의미한다.

6.3.3 활동 및 작업 설계

가. 요구분석 단계

요구분석 단계에서는 국방 CBD 방법론의 3개 활동인 요구사항 정의, 아키텍처 정의, 요구사항 분석과 12

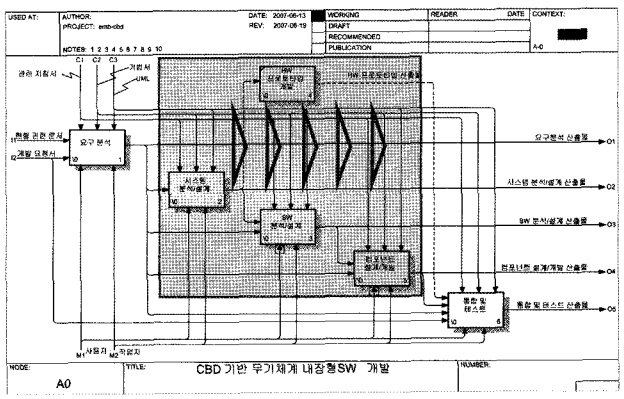

그림 4 단계 설계 구성도

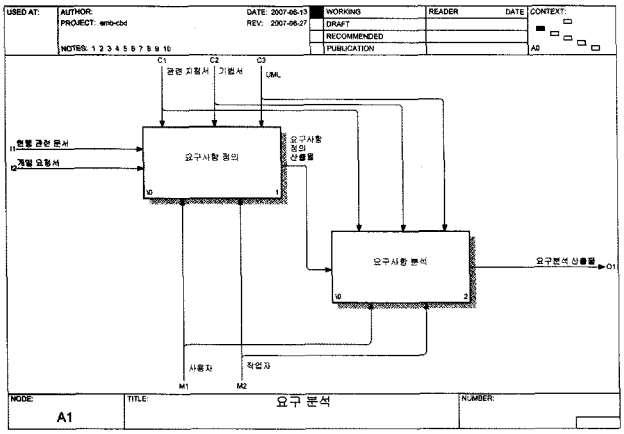

그림 5 요구분석 단계 활동 구성도

개 작업을 참조하며, DESS에서는 사용자 요구공학의 6개 작업을 참조한다. 그리고 마르미-EM에서는 전략 기획 및 요구사항 영역활동을 8개 작업을 참조하여 설계한다. 이와 같은 요구분석 단계에서의 활동 구성도가 그림 5에 도시되어 있다.

그림 5에서 요구분석 단계는 요구사항 정의와 요구사항 분석의 2가지 활동으로 구성되며, 요구사항 정의는 상위요구사항 정의, 도메인 모델링, 요구사항 명세의 3가지 작업, 그리고 요구사항 분석은 유스케이스 모델링, 클래스 모델링, UI 정의, 테스트 케이스 정의의 4가지 작업으로 구성된다.

나. 시스템 분석/설계 단계

시스템 분석/설계 단계에서는 임베디드 시스템 개발 시의 특징을 반영하여 하드웨어에 의한 구현 기능과 소프트웨어에 의한 구현 기능을 명시하고 동시 설계를 고려한다. 또한 시스템 아키텍처 수립을 위한 활동 및 작업을 반영하고 비기능적 요소의 정의를 통해 품질 기준을 수립할 수 있는 방안을 제시한다.

반영된 국내·외 개발방법론은 국방 CBD 방법론의 요구사항 정의와 아키텍처 정의의 3개 작업, DESS의 시스템 요구공학의 3개 작업, 마르미-EM의 요구사항 및 아키텍처의 7개 작업을 참조하며, 총 2개 활동, 7개 작업으로 구성된다. 이와 같은 시스템 분석/설계 단계의 활동 구성도가 그림 6에 나타나 있다.

시스템 분석/설계 단계는 시스템 요구사항 분석과 시스템 아키텍처 설계의 2개 활동으로 구성된다. 시스템 요구사항 분석은 시스템 요구사항 추출, 시스템 요구사항 분류, 시스템 기능/비기능 정의, HW/SW 기능/비기능 정의, 테스트 케이스 정의의 5개 작업으로 구성되며, 시스템 아키텍처 설계는 시스템 문맥정의, 시스템 아키텍처 요구사항 분석, 시스템 아키텍처 정의의 3개 작업으로 구성한다.

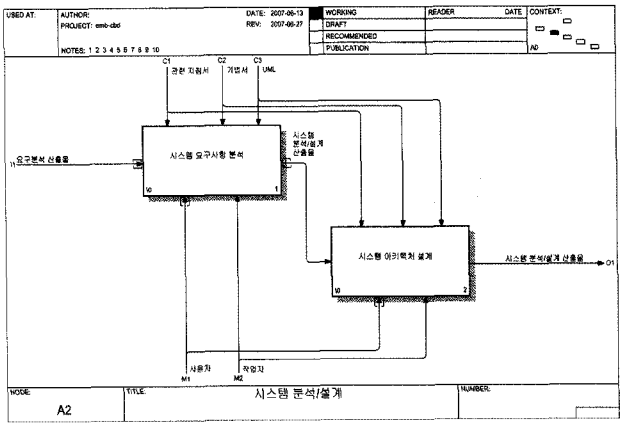

그림 6 시스템 분석/설계 단계 활동 구성도

다. SW 분석/설계 단계

SW 분석/설계 단계에서는 소프트웨어 개발 프로세스 절차[16]와 임베디드 시스템 수명주기를 반영하며 소프트웨어의 재사용 개념도 또한 반영한다. 반영된 국내·외 개발방법으로 국방 CBD 방법론의 아키텍처 정의 1개 작업, DESS의 소프트웨어 요구공학 3개 작업, 마르미-EM의 아키텍처 1개 작업을 참조하며, 총 2개 활동, 6개 작업으로 구성된다. 이와 같은 SW 분석/설계 단계의 활동 구성도가 그림 7에 도시되어 있다.

SW 분석/설계 단계는 SW 요구사항 분석과 SW 아키텍처 설계의 2개 활동으로 구성된다. SW 요구사항 분석은 SW 요구사항 추출, SW 요구사항 분류, SW 요구사항 명세, SW 테스트 케이스 정의의 4개 작업으로 구성되며, SW 아키텍처 설계는 SW 아키텍처 정의, SW 아키텍처 설계, SW 상세설계의 3개 작업으로 구성된다.

라. 컴포넌트 설계/구현 단계

컴포넌트 설계/구현 단계에서는 소프트웨어 개발 프로세스 절차 중 SW 상세 설계와 SW 코딩 및 단위 테스트 부분을 반영하며, 재사용성 향상을 위한 컴포넌트 개발(CD) 절차를 반영한다. 또한 재사용이 고려된 컴포넌트 획득방법도 반영한다.

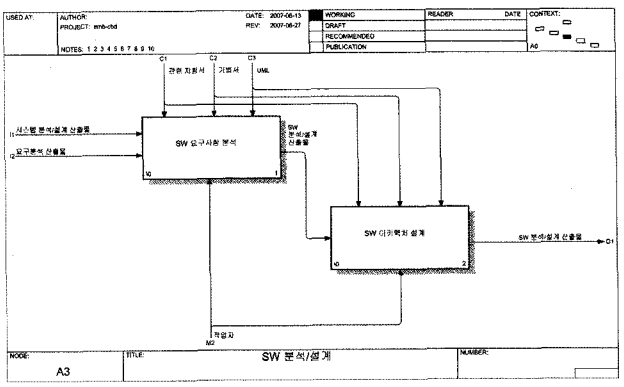

그림 7 SW 분석/설계 단계 활동 구성도

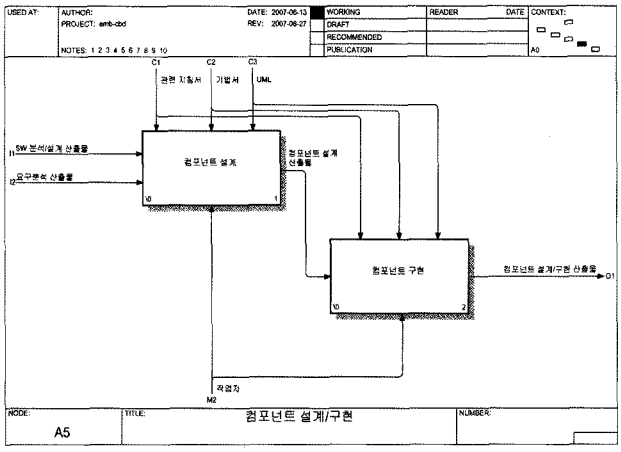

그림 8 컴포넌트 설계/구현 단계 활동 구성도

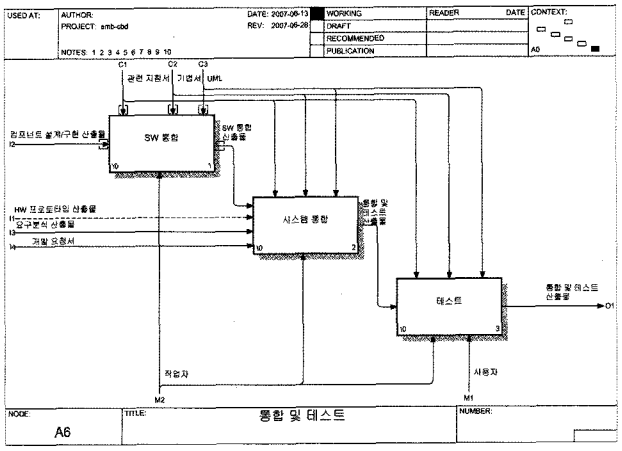

그림 9 통합 및 테스트 단계 활동 구성도

반영된 국내·외 개발방법론은 국방 CBD 방법론의 개략설계, 상세설계, 테스트 준비 및 구현의 4개 활동과 7개 작업, DESS의 명세 및 설계, 구현, 초기화 및 V & V의 6개 작업, 마르미-EM의 도메인, 설계, 구현, 테스트 부분의 7개 작업을 참조하여 설계하며, 총 2개 활동, 6개 작업으로 구성된다. 이와 같은 컴포넌트 설계/구현 단계의 구성도가 그림 8에 도시되어 있다.

컴포넌트 설계/구현 단계는 컴포넌트 설계와 컴포넌트 구현의 2개 활동으로 구성된다. 컴포넌트 설계는 컴포넌트 요구분석, 컴포넌트 아키텍처 정의, 컴포넌트 획득방법 식별의 3개 작업으로 구성되며, 컴포넌트 구현은 공통 요구사항 명세, 기반 요구사항 명세, 컴포넌트 구현, 컴포넌트 테스트의 4개 작업으로 구성된다.

마. 통합 및 테스트 단계

통합 및 테스트 단계에서는 시스템, 하드웨어 및 소프트웨어 통합을 반영하며 임베디드 소프트웨어의 특성을 반영한 테스트 작업 수행을 위해 HW/SW 통합 테스트, 필드 테스트를 지원한다. 또한 제품의 공신력을 보증하기 위해 공인된 인증기관으로부터 인증 테스트를 받기 위한 준비절차도 지원한다. 리그레션 테스트는 유지보수와 관련된 내용으로 범위에서 제외한다.

반영된 국내외 개발방법론은 국방 CBD 방법론의 테스트 준비, 통합 테스트, 시스템 테스트, 인수 지원의 4개 활동과 6개 작업, DESS의 소프트웨어 통합, 시스템 통합, 배포의 5개 작업, 마르미-EM의 테스트 부분의 10개 작업을 참조하여 설계하며 총 3개 활동 8개 작업으로 구성한다. 이와 같은 통합 및 테스트 단계의 구성도가 그림 9에 도시되어 있다.

통합 및 테스트 단계는 SW 통합, 시스템 통합, 테스트의 3개 활동으로 구성된다. SW 통합은 SW 통합 테스트 계획 수립과 SW 통합 및 테스트의 2개 작업으로 구성되며, 시스템 통합은 통합 테스트 계획 수립, SW/HW 통합 및 테스트, 시스템 통합 및 테스트의 3개 작업으로 구성되고, 테스트 활동은 인수 테스트, 필드 테스트, 인증 테스트 준비의 3개 작업으로 구성된다.

6. 결 론

본 논문에서는 소프트웨어 재사용을 고려하고 무기체계 간 상호운용성을 확보하기 위해 기존의 '국방 CBD 방법론'을 기반으로 무기체계 임베디드 소프트웨어에 적용할 수 있는 '무기체계 임베디드 소프트웨어 개발 프로세스'를 제안하였다.

이를 위해 국내·외 CBD 기반 개발 프로세스와 제품계열 기반 개발 프로세스에 대한 비교 분석 등을 실시하였으며, 5단계의 무기체계 임베디드 소프트웨어 특징 도출절차를 통해 실시간성, 테스트 어려움성, 고 신뢰성, 목적 한정성, 개발의 어려움성, 하드웨어 통합성 등의 6개 특징을 제시하였다.

마지막으로 본 논문의 핵심분야인 CBD 기반 무기체계 임베디드 소프트웨어 개발 프로세스를 제안을 위해 국방 CBD 방법론, DESS, 마르미-EM을 참조하여 총 6단계, 11개 활동, 37개의 작업으로 구성된 개발 프로세스를 기본 설계, 단계 설계, 활동 및 작업 설계 순으로 제시하였다.

제시한 국방 CBD 기반 임베디드 소프트웨어 개발 프로세스는 UML을 기반으로 하며 각 작업마다 설명서와 산출물 양식을 제공하여 개발방법론에 익숙치 않은 개발자도 쉽게 방법론을 적용할 수 있도록 하였다. 또한 임베디드 소프트웨어 생명주기 특징인 하드웨어/소프트웨어 기능 결정 및 동시 설계 기능을 반영하였고 임베디드 시스템의 위험요소 최소화를 위한 프로토타이핑 기법과 점진적 개발을 적용하였다.

참고문헌

[1] 국방부, 「국방 CBD 방법론」, 버전 1.1, 2005. 2.

[2] 배두환 외 4, "CBD 방법론 비교 분석," 한국정보처리학회 논문집, 제10권 제3호, pp. 30-39, 2003. 5.

[3] 한국전자통신연구원, 「컴포넌트 기반 시스템 개발 방법론 마르미-III」, 버전 4.0, 2003. 5.

[4] I. Jacobson, G. Booch, J Runbaugh, The Unified Software Development Process, Addison Wesley, 1998.12.

[5] Desmond D'Souza & Alan Wills, Objects, Components and Frameworks with UML - The Catalysis Approach, Addison Wesley, 1998.07.

[6] Stefan Van Baelen, Joris Gorinsek and Amderw Wills, The DESS Methodology, Information Technology for European Advancement, 2001.12.

[7] Klaus Schmid, Martin Verlage, "The Economic Impact of Product Line Adoption and Evolution," IEEE Software, Vol. 19, No. 4, pp. 50-57, July/August, 2002.

[8] 한국전자통신연구원, 『제품계열 기반 시스템 개발 방법론 마르미-EM』, EMMA Ver 1.0, 2006. 05.

[9] Maarit Harsu, "FAST Product Line Architecture Process," Software System Laboratory Tampere University of Technology, 2000.

[10] Linda M. Northrop, "SEI's Software Product Line Tenets," IEEE Software, Vol. 19, No. 4, pp. 32-40, July/August, 2002.

[11] Jan Bosch, Peter Molin, "Software Architecture Design: Evaluation and Transformation," IEEE Conference and Workshop on Engineering of Computer-Based System, 1999.

[12] Kyo C. Kang, "FORM : A Feature-Oriented Reuse Method with Domain-Specific Reference Architectures," Annals of Software Engineering, Vol. 5, pp. 143-168, 1998.

[13] Colin Atkinson, Joachim Bayer, Dirk Muthing, "Component Based Product Line Development: The KobrA Approach," Proceeding of the First Software Product Line Conference, 2000.

[14] 국방부, 「무기/비무기체계 임베디드 소프트웨어 개발 관리지침」, 2001. 12.

[15] Software Technology Support Center, Guidelines for Successful Acquisition and Management of Software-Intensive Systems, Ver. 3.0, DoAF, May 2000.

[16] 방위사업청, 「소프트웨어 개발 프로세스」, 2006. 1.

윤 희 병

1983 해군사관학교(이학사)
1986 연세대학교 (공학사)
1991 미국 해군대학원 전산공학(석사)
1998 미국 조지아공대 전산공학(박사)
2002~현재 국방대 전산정보학과 교수
2004~현재 아주대학교 정보통신대학원 외래교수
2005~현재 국방소프트웨어 산학연 협회 기획이사
2006~현재 서강대학교 정보통신대학원 외래교수
관심분야 : 임베디드 소프트웨어, 소프트웨어 테스팅, NCES, 전술데이터링크
E-mail : hbyoon37@hanmail.net

오 행 록

1987 인하대학교 전산학과(이학사)
1989 인하대학교 전산학 전공(이학석사)
1989~현재 국방과학연구소 책임연구원
관심분야 : 데이터베이스, 상호운용성, 전술데이터링크
E-mail : haengrok@hanmail.net

조 병 인

1984 동국대학교 전산학과 학사
1986 한국외국어대학교 석사
2000 미국일리노이즈공대 박사
1986~현재 국방과학연구소 책임연구원
관심분야 : 상호운용성, 인공지능
E-mail : chobyun@hotmail.net