KIPS Trans. Softw. and Data Eng.
Vol.10, No.6 pp.201~210
ISSN: 2287-5905 (Print), ISSN: 2734-0503 (Online)

https://doi.org/10.3745/KTSDE.2021.10.6.201

Operational Concept for the Software Product Line Framework of Navigation Software

Samjoon Park[†] · Sungkyu Noh[††] · Dohyung Kim[†††] · Sunju Lee[††††] · ByungSu Park[††††] · Inseop Lee[††††]

ABSTRACT

Navigation Software for the various weapon systems has common functionalities which give the possibility of common use among them. SPL(Software Product Line) framework of the navigation software for weapon system refers to developing a standardized navigation software platform from common functionalities of navigation software, managing the standardized navigation software platform, and developing weapon system navigation software such as navigation software for missile, UAV(Unmanned Air Vehicle), submarine, and etc. from the standardized navigation software platform. In this paper, we propose SPL based navigation software development process, Integrated Development Environment and operational concept of SPL framework. The operational concept will be defined by specifying the role of every stake holders and their activity scenario. The Operational concept would be referenced to implement SPL for other domain through using with detail implementation guide.

Keywords : Inertial Navigation System(INS) Software, Software Product Line(SPL), Platform, IDE, Operational Concept

항법소프트웨어 Software Product Line 프레임워크 운영개념

박 삼 준[†] · 노 성 규[††] · 김 도 형[†††] · 이 순 주[††††] · 박 병 수[††††] · 이 인 섭[††††]

요 약

무기체계에 탑재되는 항법소프트웨어의 기능요소들은 다양한 무기체계에 공통적으로 활용될 수 있는 가능성이 높은 것으로 식별되었다. 무기체계 항법소프트웨어 SPL 프레임워크는 다양한 무기체계에 적용되는 항법소프트웨어가 갖는 공통의 기능들을 표준화된 플랫폼으로 개발 및 관리하고, 이를 이용하여 유도무기, 무인기, 잠수함 등의 체계별 항법소프트웨어를 개발하는 방법론을 말한다. 본 논문에서는 SPL 기반 항법소프트웨어 개발 프로세스와 통합개발환경을 제시하고, SPL 프레임워크의 운영개념에 대해 설명한다. 프로세스에 관여하게 되는 참여자의 역할을 정의하고 각 역할자별 활동 시나리오를 도출함으로써 SPL 프레임워크의 운영개념을 정의한다. 제시한 운영개념은 여타 도메인에 SPL을 실현하기 위한 구체적인 지침 마련에 활용될 수 있을 것이다.

키워드 : 항법소프트웨어, 소프트웨어제품라인, 플랫폼, 통합개발환경, 운영개념

1. 서 론

무기체계의 항법시스템(Navigation System)은 동체의 위치, 속도 및 자세를 결정하는 장치를 말한다. 항법소프트웨어는 항법시스템에 탑재되어 목표 지점까지의 항법에 필요한 기능들을 처리하게 된다. 국내의 경우, 이러한 항법 기능은 다양한 유도, 지상, 해양, 항공 등의 무기체계에 필수요소로 현재는 각 체계별로 세부기능을 개발하여 사용 중이다.

항법에 필요한 기능을 위해 항법소프트웨어는 동체의 현재 자세를 찾는 과정인 정렬 기능, 자세, 속도, 위치계산 및 오차 보정을 위한 순수항법 기능 등을 갖는다. 또한 특정 체계의 경우 외부의 비 관성 항법센서(Global Positioning System, 영상센서 등)를 활용한 복합항법 기능 등의 연산을 수행한다.

항법소프트웨어 개발 및 적용 현황과 기술 등에 대한 도메인 분석 결과, 무기체계에 탑재되는 항법소프트웨어의 기능요소들은 다양한 무기체계에 공통적으로 활용될 수 있는 가능성이 높은 것으로 식별되었다. 그런데 현재 무기체계 소프트웨어의 개발은 단일 소프트웨어 개발 프로세스를 기반으로 이루어지고 있다. 소프트웨어를 개발하는 과정에서 생산성 향상을 위한 코드 단위의 재사용이나 기술적인 공통 모듈, 알고리즘의 재사용은 부분적으로 이루어져 왔다. 이러한 방식

※ 이 논문은 방위사업청 핵심SW과제 연구비에 의하여 연구되었음.
† 비 회 원 : 국방과학연구소 수석연구원
†† 정 회 원 : 국방과학연구소 선임연구원
††† 비 회 원 : 국방과학연구소 책임연구원
†††† 비 회 원 : 국방과학연구소 선임연구원
Manuscript Received : January 28, 2021
First Revision : March 10, 2021
Second Revision : March 26, 2021
Accepted : March 31, 2021
* Corresponding Author : Sungkyu Noh(snoh@add.re.kr)

으로는 유사 분야에서 재사용 가능한 조직 차원의 자산을 확보하고 재활용하는 수준으로 이루어지기는 어려우며, 환경이나 기반 기술 등의 다양성을 고려하는 데 한계가 있다[1].

본 논문에서는 아키텍처 수준의 표준화를 통해 정의된 플랫폼을 기반으로 다양한 형태의 소프트웨어 개발을 가능하게 하는 항법소프트웨어 SPL 프레임워크와 그 운영개념을 정의한다. 항법소프트웨어 SPL 프레임워크는 수행 프로세스와 환경을 기술함으로써 정의된다. 이때 프로세스는 도메인 공학 프로세스와 애플리케이션 공학 프로세스로 구성된다. 각 프로세스에서 관여하게 되는 참여자의 역할을 정의하고 그 들의 활동 시나리오를 도출함으로써 SPL 프레임워크의 운영개념을 정의한다.

먼저 국내외 SPL 적용현황을 살펴보고, 다음으로 개발프로세스 및 통합개발환경을 포함하는 항법소프트웨어 SPL 프레임워크를 정의한다. 다음으로 그 프로세스 및 환경에서 이루어지게 되는 운영개념을 설명한다.

2. 현황 분석

그동안 SPL 개념을 적용하는데 필요한 기본적인 여러 기술들이 연구되어 왔다. 공통 활용의 기반이 되는 플랫폼을 개발하기 위한 제품라인 범위설정 기술, 가변성 모델링 기술, 가변성 표현 기술, 가변성 구현기술 등이 제시되고 있다[2,3,4,5].

또한 SPL 개념을 효과적으로 구현하는데 필요한 가변성 모델링 기능과 외부도구와의 연동을 통해 분석부터 구현까지 필요한 일관된 기능 등을 제공하는 통합개발환경도 다양한 기능을 포함하여 개발되고 있다[6,7].

ISO/IEC JTC1/SC7은 2006년부터 소프트웨어 플랫폼 아키텍처 기술 관련 국제표준화를 진행해 왔다. 현재 소프트웨어 플랫폼 아키텍처 기술 참조모델(ISO/IEC 26550), 소프트웨어 플랫폼 아키텍처를 위한 요구공학(ISO/IEC 26551), 소프트웨어 플랫폼 아키텍처 기술을 위한 기술관리(ISO/IEC 26555) 등에 대한 국제표준이 제정되어 있다. 그러나 기존의 소프트웨어 플랫폼 공학을 지원하는 방법론과 도구들은 체계적으로 정립된 표준 기술이 없고, SPL 기반의 소프트웨어 생명주기 전 과정을 지원하지 못한다[2,8].

CMU(Carnegie Mellon University) SEI(software Engineering Institute)에서 개발된 SEI SPL 프레임워크는 프러덕트라인 개념과 적용지침 등을 포함하고 있다. 프로덕트라인의 개념과 프로덕트 개발 전에 고려해야 할 필수 행위 및 조직이 마스터해야 하는 실제 부분을 제시하고, 기술적, 관리적 프로세스 및 지침을 제공하고 있다. 하지만 프로세스가 선언적 수준의 가이드 형태로 제시되어 현장에서 SPL 구현을 위한 실질적 프레임워크로 적용하기에 한계가 있다[9].

국방과학연구소에서는 2018년 무기체계 소프트웨어 분야에 SPL 패러다임을 적용하기 위해 수행중인 “SPL 기반의 무기체계 소프트웨어 개발 프레임워크 구축” 결과를 소개하였다. 무기체계 분야 SPL 적용을 위한 프로세스, 가변성(variability) 등을 지원하기 위한 주요 요소기술 및 이를 지원할 수 있는 국내외 도구 현황, 과제를 통해 개발 중인 무기체계 SPL 통합개발

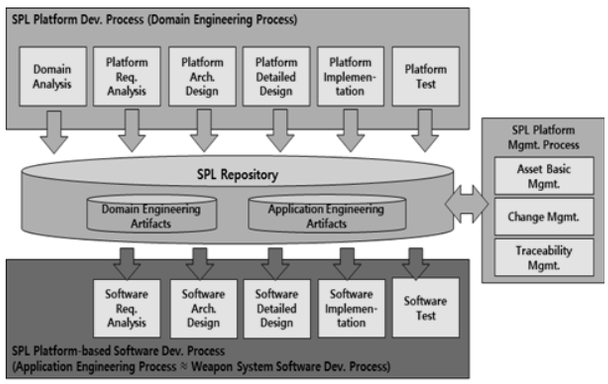

Fig. 1. SPL Process for Navigation Software

도구 및 SPL 플랫폼 시범개발 내용에 대해 소개하였다[10].

본 논문에서는 기존 연구[10]에서 확보한 기본 프레임워크를 바탕으로 항법소프트웨어 개발에 적용하기 위한 맞춤형(tailoring) SPL 프레임워크와 그 운용개념을 다룬다. 여기서 제시하는 운영개념을 참조하여 해당 도메인별로 누가, 언제, 어떤 활동을 해야 하는지를 명확히 정의할 수 있게 되므로 SPL 실현을 위한 구체적인 지침으로 활용될 수 있을 것이다.

3. SPL 기반 항법소프트웨어 개발 프로세스

SPL 기반 항법소프트웨어 개발 프로세스는 Fig. 1에 나타난 바와 같이 크게 도메인 공학 프로세스, 애플리케이션 공학 프로세스, 자산 관리 프로세스로 나뉜다. 도메인 공학 프로세스는 애플리케이션 공학 프로세스 과정에서 사용될 재사용 가능한 플랫폼 자산을 만드는 활동들로 구성되어 있고, 애플리케이션 공학 프로세스는 플랫폼 자산을 활용하여 다양한 애플리케이션을 개발하는 활동들로 구성되어 있다. 자산 관리 프로세스는 도메인 공학 프로세스나 애플리케이션 공학 프로세스 도중에 활용될 자산의 변경 및 추적성을 관리하는 활동들로 구성되어 있다.

3.1 도메인 공학 프로세스

도메인 공학 프로세스는 도메인 분석, 플랫폼 요구분석, 플랫폼 아키텍처 설계, 플랫폼 상세설계, 플랫폼 구현, 플랫폼 시험, 플랫폼 일관성 분석 활동 등으로 세분화 된다.

도메인 분석은 재사용 가능한 플랫폼 자산의 개발 및 적용 범위인 제품라인 범위를 설정하고, 이 범위에 속하는 소프트웨어 제품들 사이의 공통점과 차이점을 분석하는 피처모델링 활동 등으로 구성된다. 여기서 피처란 특정 도메인의 요구사항을 만족시키는데 필요한 다양한 소프트웨어 기능요소를 말한다.

플랫폼 요구분석은 플랫폼 요구사항을 도출하고, 도출된 플랫폼 요구사항을 정의하고, 마지막으로 플랫폼 요구사항의 가변성을 모델링하는 활동 등으로 구성된다.

플랫폼 아키텍처 설계는 플랫폼 아키텍처를 구성하는 구성요소를 도출하고, 도출된 플랫폼 아키텍처 구성요소들을 바탕으로 플랫폼 아키텍처 모델을 개발하고, 플랫폼 아키텍처

모델의 가변성을 모델링하는 활동 등으로 구성된다.

플랫폼 상세설계는 플랫폼 설계모듈 식별, 플랫폼 설계모듈 모델링, 플랫폼 설계모듈 가변성 모델링으로 구성된다.

플랫폼 구현은 가변성 구현기법 선정과 설계모듈의 가변성 구현으로 구성된다. 가변성을 구현하는 방법은 다양한 방법이 존재하며 각 방법마다 장단점이 존재한다. 가변성 구현기법 선정은 가변성을 구현하는 기법 및 프로그래밍 언어를 결정하는 활동을 말한다. 구현코드 내에 가변성을 포함시킬 수도 있고, 구현코드의 가변성을 독립적인 가변성 모델로 표현하고 관리할 수도 있다.

플랫폼 시험은 플랫폼 시험계획서 작성, 플랫폼 단위시험 수행, 플랫폼 테스트케이스 명세서 작성, 플랫폼 테스트케이스 가변성 모델링, 플랫폼 시험절차서 작성, 플랫폼 통합시험 전략 선정, 플랫폼 통합시험 수행, 플랫폼 시험 결과보고서 작성 등으로 구성된다.

3.2 애플리케이션 공학 프로세스

애플리케이션 공학 프로세스는 각 무기체계별 항법소프트웨어 개발자가 SPL 기반의 항법소프트웨어 개발을 위한 프로세스를 말한다. 이 프로세스는 플랫폼 기반 애플리케이션 인스턴스화, 애플리케이션 특화 요구분석, 애플리케이션 아키텍처 설계, 애플리케이션 상세설계, 애플리케이션 구현, 애플리케이션 시험 활동 등으로 세분화 된다.

플랫폼 기반 애플리케이션 인스턴스화는 피처선택에 따라 플랫폼 자산으로부터 항법소프트웨어 산출물을 도출하는 활동들을 포함한다. 피처선택은 피처모델로부터 특정 무기체계 항법소프트웨어를 위한 피처를 선택한 결과인 피처 컨피규레이션을 도출하는 활동이다.

플랫폼 기반 애플리케이션 요구사항 도출은 피처 컨피규레이션과 플랫폼 요구사항 가변성 모델을 바탕으로 가변점과 가변요소를 결정하고, 결정된 가변점과 가변요소를 기반으로 플랫폼 요구사항 명세서로부터 항법소프트웨어 요구사항 명세서를 도출하는 활동이다.

플랫폼 기반 애플리케이션 아키텍처 도출은 피처 컨피규레이션과 플랫폼 아키텍처 가변성 모델을 바탕으로 가변점과 가변요소를 결정하고, 결정된 가변점과 가변요소를 기반으로 플랫폼 아키텍처 모델로부터 항법소프트웨어 아키텍처 모델을 도출하는 활동이다.

플랫폼 기반 애플리케이션 설계모듈 도출은 피처 컨피규레이션과 설계 모듈 가변성 모델을 바탕으로 가변점과 가변요소를 결정하고, 결정된 가변점과 가변요소를 기반으로 플랫폼 설계 모델로부터 항법소프트웨어 설계 모델을 도출하는 활동이다.

플랫폼 기반 애플리케이션 구현코드 도출은 피처 컨피규레이션과 플랫폼 구현코드의 가변점과 가변요소를 결정하여 항법소프트웨어의 구현코드를 도출하는 활동이다.

플랫폼 기반 애플리케이션 테스트케이스 도출은 피처 컨피규레이션과 플랫폼 테스트케이스의 가변점과 가변요소를 결정하여 항법소프트웨어의 테스트케이스 명세서를 도출하는 활동을 말한다.

4. SPL 통합개발환경

SPL 통합개발환경은 기본적으로 소프트웨어 제품라인 공학(Software Product Line Engineering, SPLE)에서 정의하고 있는 도메인 공학(Domain Engineering)과 애플리케이션 공학(Application Engineering) 개발 프로세스 범위에서 운용된다. 도메인 공학 프로세스와 애플리케이션 공학 프로세스는 방위사업청에서 제시하고 있는 무기체계 소프트웨어 개발 프로세스[11]와 각각 호환된다.

Fig. 2에 나타난 바와 같이 SPL 통합개발환경은 SPLE 개발 프로세스를 지원하기 위해서 도메인 엔지니어링 도구, 어

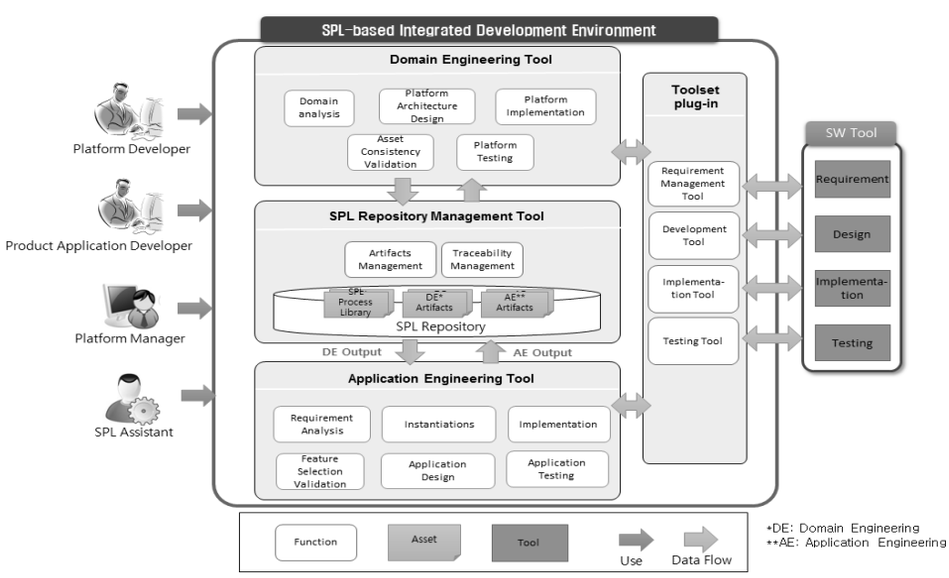

Fig. 2. SPL IDE(Integrated Development Environment)[10]

Table 1. SPL IDE Tools

Category	Function
Domain Engineering Tool	· Domain analysis, Platform architecture design, Platform implementation, Platform testing, Consistency validation · Plug-in Interface with Software Development
SPL Repository and Management. Tool	· Artifacts and SPL process Library Management · Variation Information and Point Management, Feature relation with feature · Traceability Information · Reuse Monitoring and Evolution Management
Application Engineering Tool	· Feature Configuration Validation · Application insanitation · Requirement Analysis, Design, Implementation, Testing
Plug-in Tool	· Software Development Tool · Variation Point and Variants Specification

Table 2. External Development Tool

Area	Function and Role	COTS Tool
Require-ment Manage-ment	· Platform Requirement Management · Variation Point · Requirement Specification Management based on Instances	MS EXCEL
Design	· Platform Design · Variation Point View · Design management based on design instances	IBM Rhap-sody
Implem-entation	· Platform Coding · Variation Coding · Source Code Build · Application Coding and Build	Visual Studio
Testing	· Platform Test Cases Development · Test Case Management based on Test Case Instances · Application Testing	LDRA

Table 3. Role of SPL Stakeholder

Category	Role	Assignment
Platform Developer	· Develop SPL platform for specific domain	Development Task Force
Product Application Developer	· Develop application software using SPL platform	Software Developer
Platform Manager	· Platform change management, reuse monitoring, evolution management · Assist application software development and application management	Development Task Force
SPL Assistant	· SPL technical support · SPL IDE management · Training and guide tools and SPL process · External tool interface development and management	SPLide Management Team

플리케이션 엔지니어링 도구, SPL 레파지토리 관리도구, 툴 셀 플러그인 등 모두 4개의 하위 도구를 제공하며, 무기체계 소프트웨어 개발에 사용되는 소프트웨어 개발도구와 연동 또는 연계하여 운용된다.

외부 소프트웨어개발도구는 일반적인 무기체계 소프트웨어개발에 사용되는 소프트웨어 엔지니어링 도구를 의미한다. SPL 통합개발환경과 SPL 통합개발환경 플러그-인을 통해서 연계되는 외부 소프트웨어개발도구는 요구사항관리 도구, 설계 도구, 구현도구, 테스팅 도구 등 모두 네 가지 유형이 존재한다.

항법소프트웨어 SPL 프레임워크에서는 요구분석도구로 Microsoft EXCEL, 모델링 및 설계도구로 IBM Rapsody, 구현도구로 Visual Studio, 소프트웨어 테스팅 도구로 LDRA를 연동하여 사용한다. 각 소프트웨어개발도구가 SPL 통합개발환경과 상호작용을 하면서 수행하는 역할은 Table 1과 같다.

SPL 통합개발환경 플러그-인을 통해서 연계되는 외부 소프트웨어 개발도구가 SPL 통합개발환경과 상호작용을 하면서 수행하는 역할은 Table 2와 같다.

5. 항법소프트웨어 SPL 프레임워크 운용개념

5.1 참여자

Table 3에 나타난 바와 같이 SPL 통합개발환경과 관련되어 정의된 참여자(stakeholder)는 플랫폼 개발자, 애플리케이션 개발자, 플랫폼 관리자, SPL 어시스턴트 등 모두 넷 이다. 무기체계 SW 개발자 외의 참여자는 SPLE 활동과 관련되어 추가로 정의된 관계자들이다. 항법소프트웨어 SPL 운용개념은 이러한 4가지 역할의 참여자 활동을 기준으로 정의된다.

5.2 참여자 별 운용 시나리오

항법소프트웨어 SPL 플랫폼 개발은 하나의 공용 저장소를 이용하여 소프트웨어를 개발하는 방식으로 진행된다. 전체적인 운용시나리오는 Fig. 3에 나타난 바와 같으며 다음 절차로 진행된다.

① 플랫폼 개발자는 담당하고 있는 활동에 적합한 SPL 통합개발환경의 도구(도메인공학도구, SPL 레파지토리 관리도구, 외부도구Plug-in)와 소프트웨어 개발도구를 이용하여 핵심 자산을 만드는 작업을 한다.

② 플랫폼 개발자는 정해진 동기화 규칙에 따라서 SPL 자산저장소에 보관되어 있는 자산과 자신의 자산을 동기화 시킨다.

③ 플랫폼 관리자는 SPL 자산저장소에 보관된 자산이 주어진 기준을 만족시키면 베이스라인을 확정하고 개발자들에게 알린다.

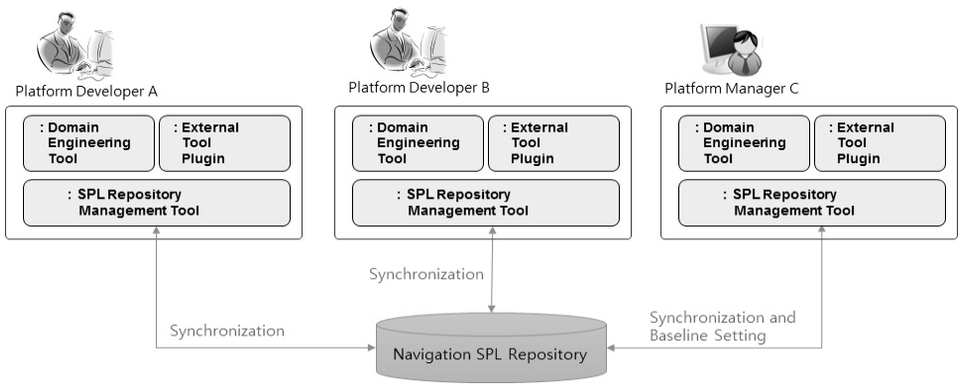

Fig. 3. Platform Development Scenario

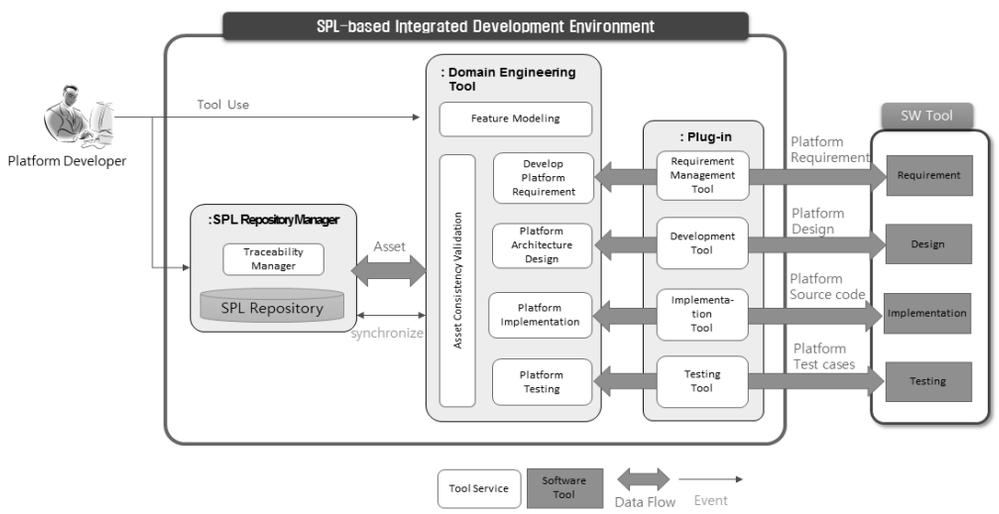

Fig. 4. Operation Concept for SPL IDE by Platform Developer

④ 플랫폼 개발자는 SPL 자산저장소의 베이스라인자산과 자신의 자산을 동기화 시킨다.
⑤ SPL 어시스턴트는 전체적인 과정에서 각 개발자 및 관리자를 기술적으로 지원하며, 통합개발환경을 관리한다. 개발 도구와 프로세스에 대한 교육과 외부 연동기능의 확장 등을 담당한다.
⑥ 애플리케이션 개발자는 SPL 자산저장소에서 관리되는 핵심자산을 활용하여 무기체계별 항법소프트웨어를 개발한다.

1) 플랫폼 개발자

Fig. 4에 나타난 바와 같이 플랫폼 개발자는 SPL 통합개발환경을 이용하여 항법소프트웨어 SPL 플랫폼을 개발한다. 통합개발환경에서 피처모델을 개발하고 플랫폼 요구사항을 도출하고 설계 및 구현, 시험 등의 일련의 개발활동을 진행한다.

a) 피처모델 개발

플랫폼 개발자는 기 적용중인 체계와 향후 적용 예상되는 체계를 기준으로 항법소프트웨어 피처와 가변성을 식별한다.
① 도메인 분석을 통해서 항법소프트웨어의 피처를 도출한다.
② 도출된 피처를 평가할 수 있는 기준을 정립한다.
③ 정립된 기준으로 피처를 평가함으로써 플랫폼 대상 피처를 정의한다.
④ 식별된 피처 모델을 통합개발환경에 입력하여 관리한다.
⑤ Fig. 5에서와 같이 피처모델 변경 시 로컬에서 일단 변경 한 후 자신의 피처모델과 SPL 저장소의 피처모델과 동기화한다.

b) SPL 플랫폼 요구사항 도출

플랫폼 개발자는 기존의 요구사항 문서에 대한 분석을 바탕으로 항법소프트웨어 플랫폼 요구사항 문서를 작성한다.
① 선정된 피처 단위로 최종(leaf) 노드를 기준으로 요구사항을 정의한다.
② 필요시 상위노드의 피처에 대해서도 요구사항을 정의할 수 있다.
③ 피처모델과의 추적성을 위해 요구사항 항목별로 가변성을 정의한다.
④ 요구사항이 변경될 경우 적절한 통제과정을 거쳐 확정한다.

c) SPL 플랫폼 설계

플랫폼 개발자는 피처모델과 요구사항을 바탕으로 플랫폼

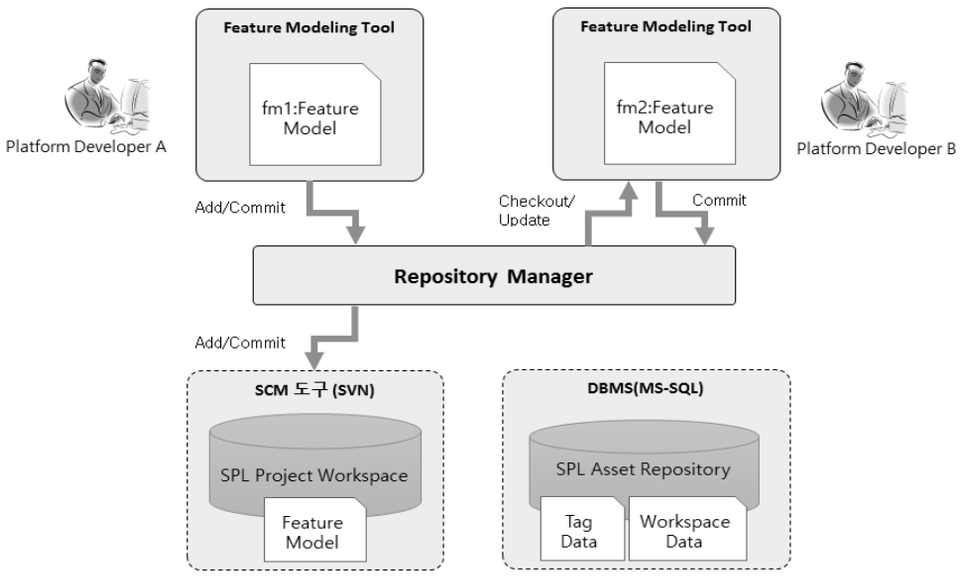

Fig. 5. Feature Model Development Scenario using SPL IDE

에 대한 설계(아키텍처, 상세설계)를 진행한다. 모델링은 외부도구인 IBM Rhapsody를 활용한다.

① 각 피처노드를 대상으로 가변성을 고려한 논리적 구성요소를 식별하여 클래스 다이어그램을 활용하여 논리뷰(logical view)를 정의한다.
② 컴포넌트 다이어그램을 활용하여 RTOS(Real Time Operating System)에서 실행되는 상황을 반영한 실행관점의 실행뷰(execution view)를 정의한다.
③ 배치(deployment) 다이어그램을 활용하여 체계에 탑재되어 구동되는 상황을 반영한 배치뷰(deployment view)를 정의한다.
④ 논리적 컴포넌트와 대응되는 물리적 컴포넌트를 식별한다.
⑤ 각 컴포넌트 내부의 변수와 함수를 식별한다.
⑥ 컴포넌트 간의 외부 인터페이스를 위해 퍼블릭 변수와 퍼블릭 함수를 식별한다. 구조체와 구조체의 변수값을 사용할 수 있는 전역에서 호출 가능한 형태의 퍼블릭 함수를 정의한다.
⑦ 시퀀스 다이어그램(sequence diagram)을 이용하여 컴포넌트간의 인터페이스를 정의한다.
⑧ 액티비티 다어어그램(activity diagram)을 이용하여 함수를 정의한다. 내부 함수의 경우 플로우차트 형태의 다이어그램으로 함수 흐름을 표현하고 세부 알고리즘에 대한 설명을 상세설계서에 기술한다.
⑨ 통합개발환경에서 설계모델과 요구사항, 피처모델과의 추적성을 입력하고 관리한다.

d) SPL 플랫폼 구현

플랫폼 개발자는 설계를 바탕으로 항법소프트웨어 SPL 플랫폼의 특정 컴포넌트를 구현한다. 구현도구로는 Visual Studio IDE를 이용하여 개발하며 동료와의 협업 개발을 위해서 Git을 이용한다.

① 플랫폼 개발자는 Visual Studio 프로젝트를 생성하고 컴포넌트 구현을 위한 소스코드를 작성한다.
② 설계를 통해 개발하는 컴포넌트의 내부에는 구현해야 하는 가변성이 있는 것을 확인하고 파라미터 기법과 전처리기 기법 등을 이용하여 소스코드를 작성한다.
③ 가변성을 소스코드에 올바르게 구현했는지 확인하기 위해서 도메인공학 도구를 이용하여 구현된 소스코드의 가변점을 추출하고 피처정보를 연결하여 가변성 정보를 생성한다.
④ 플랫폼 개발자는 플랫폼 컴포넌트의 단위 테스트를 통해서 기능적으로 문제가 없는 것을 확인하고, Git에 커밋을 시킨다.
⑤ 일관성 검사는 통합단계에서 수행한다.

e) SPL 플랫폼 시험

SPL 플랫폼 개발자는 자신이 구현한 컴포넌트에 대해서 단위시험을 진행한다. 단위시험을 위해서 정적 검사 도구와 CppUnit을 사용한다.

① 구현한 코드가 기능적으로 문제가 없는지 확인하기 위해서 CppUnit을 이용하여 테스트케이스를 만들고 진행한다.
② 가변성이 포함되어 있는 소스코드 부분에 대해서는 적용된 가변성 구현기술에 맞게 가변점 컨피규레이션 스텁(stub)을 만들어서 다른 테스트케이스와 함께 기능시험을 진행한다.
③ 정적검사도구의 제어흐름분석 결과 작성한 소스코드의 조건 커버리지가 통과기준에 미치지 못한 것을 확인하고 테스트케이스를 추가로 개발하고 테스트를 진행한다.
④ 이후 회귀테스트를 위해서 컴포넌트의 테스트 슈트를 도메인공학 도구를 통해서 SPL 자산저장소에 등록한다.

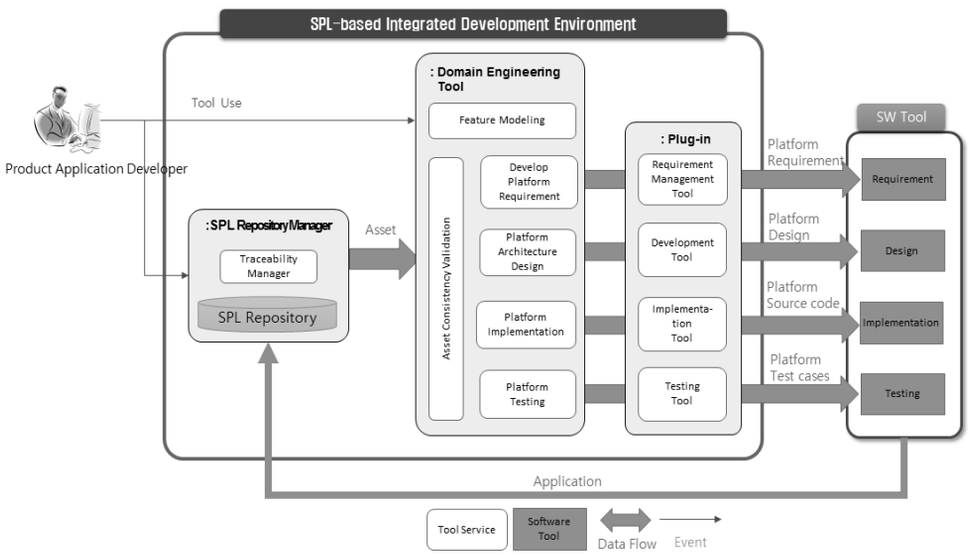

Fig. 6. Operation Concept for SPL IDE by Product Application Developer

2) 애플리케이션 개발자

Fig. 6에 나타난 바와 같이 애플리케이션 개발자는 항법소프트웨어 SPL 플랫폼을 기반으로 각 무기체계별 항법소프트웨어를 개발하게 된다. 이를 위해 체계의 기능에 필요한 피처 선택과 선택에 따른 가변성 영향을 분석하고, 필요한 산출물을 생성시킨 후 일련의 항법소프트웨어 개발활동을 진행하게 된다.

a) 피처 선택

애플리케이션 개발자는 새로운 항법소프트웨어 개발 프로젝트를 진행하기 위해서 먼저 피처를 선택한다.

① SPL 레파지토리 관리도구에 접속하여 SPL 자산저장소에 보관되어 있는 피처 컨피규레이션을 확인한다.
② 대상 체계에 필요한 피처와 피처 어트리뷰트 값을 결정하여 선택한다.
③ 이때 기 구성되어 있는 유사한 피처 컨피규레이션을 찾아서 수정해서 사용할 수도 있다.
④ SPL 레파지토리 관리도구에서 선택한 피처 컨피규레이션의 복사본을 만든다. 그리고 애플리케이션 공학 도구를 이용하여 복사본을 편집하여 피처 선택과 어트리뷰트 값을 수정한다.
⑤ 애플리케이션 공학 도구를 이용하여 선택한 피처와 설정한 피처 어트리뷰트 값에 대한 적합성 검사를 수행한다.
⑥ SPL 레파지토리 관리도구를 이용하여 새롭게 만든 피처 컨피규레이션을 자산으로 등록한다.

b) 피처 컨피규레이션 영향 분석

애플리케이션 개발자는 자산을 이용하여 무기체계 항법소프트웨어 개발의 기초로 사용할 산출물을 내려 받기 이전에 등록한 피처 컨피규레이션에 의해서 선택되는 가변요소가 무엇인지 확인한다.

① SPL 레파지토리 관리도구에서 제공하는 피처 컨피규레이션 목록에서 자산에 적용하기로 한 피처 컨피규레이션 파일을 선택한다.
② 피처 컨피규레이션 파일의 영향 분석 범위를 요구사항과 소스코드로 지정하고 '적용'을 실행시킨다.
③ SPL 레파지토리 관리도구가 보여주는 피처 컨피규레이션 목록에서 특정 피처를 선택하면 해당 피처와 관계를 가지고 있는 요구사항의 모든 가변점(파일단위 또는 파일안의 내용단위)과 각 가변점에서 선택될 가변요소에 대한 요약정보를 확인 할 수 있다. 소스코드 역시 모든 가변점(파일단위 또는 파일안의 내용)과 각 가변점에서 선택될 가변요소에 대한 요약정보를 확인 할 수 있다.

c) 애플리케이션 산출물 생성

애플리케이션 개발자는 피처 컨피규레이션에 대응되는 실제 항법소프트웨어 산출물을 생성시킨다.

① SPL 레파지토리 관리도구에서 제공하는 피처 컨피규레이션 목록에서 자산에 적용하기로 한 피처 컨피규레이션 파일을 선택한다.
② 선택한 피처 컨피규레이션에 대해서 '적용후 생성' 기능을 실행시킨다.
③ SPL 레파지토리 관리도구는 애플리케이션 공학 도구로부터 피처 컨피규레이션에 대응하는 산출물이 생성되도록 처리한다.
④ 애플리케이션 공학 도구는 피처 컨피규레이션을 적용하여 실제 산출물을 생성하고 그 결과를 SPL 레파지토리 관리도구로 전달하는 기능이 이루어 진다.
⑤ SPL 레파지토리에는 피처 컨피규레이션 적용으로 생성된 산출물들과 자산들 사이를 추적할 수 있는 관계(자산의 버전정보, 피처모델의 버전정보 등) 정보가 구축된다.
⑥ SPL 레파지토리 관리도구가 제공하는 작업공간 뷰어 상에 새롭게 추가한 항법소프트웨어 프로젝트가 신규로 등록되고 프로젝트 내부에 생성된 산출물이 위치하게 된다.

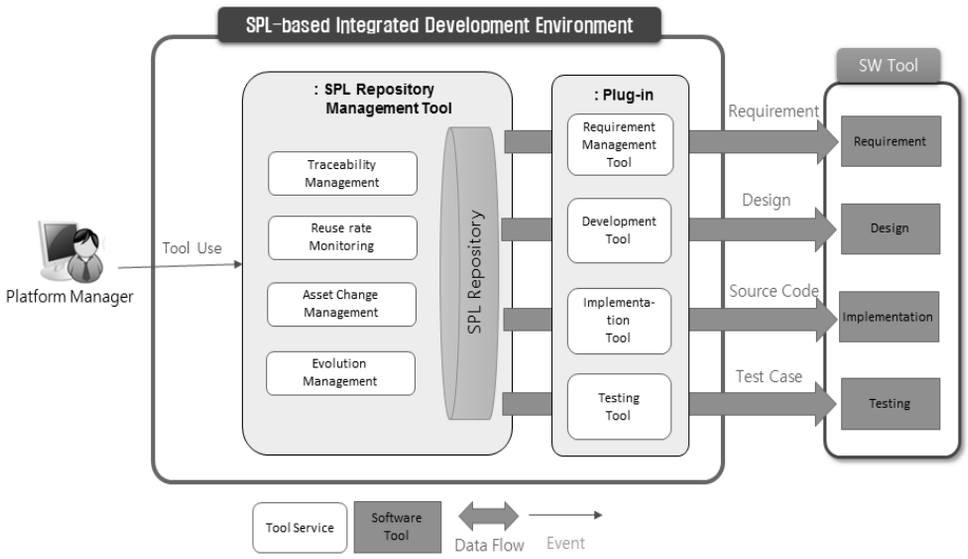

Fig. 7. Operation Concept for SPL IDE by Product Application Developer

3) 플랫폼 관리자

Fig. 7에 나타난 바와 같이 플랫폼 관리자는 제품라인 정보를 추적하고 플랫폼 개발자들이 SPL 자산저장소에 등록한 피처모델의 베이스라인 또는 릴리즈를 생성시키는 역할을 한다. 또한 자산 개발에 사용되고 있는 피처모델에 불가피한 변경이 필요할 경우, 피처모델의 변경을 진행할 수 있으며 필요에 따라 기존 SPL 플랫폼을 동결시키고, 차세대 SPL 플랫폼으로 진화시키기로 결정할 경우에는 새로운 개발 환경을 설정하게 된다.

a) 제품라인 정보 추적

플랫폼 관리자는 현재까지 얼마나 많은 무기체계 항법소프트웨어가 제품화(Production) 되었는지 확인할 수 있다.

① SPL 레파지토리 관리도구가 제공하는 작업공간 탐색기를 통해서 일단 피처모델의 릴리즈 상태를 확인할 수 있다.

② 가장 최근에 릴리즈된 피처모델을 선택하고 애플리케이션 추적 기능을 실행하면 SPL 레파지토리 관리도구는 현재 선택된 피처모델의 버전으로 만들어진 피처 컨피규레이션 목록, 피처모델과 관계가 설정된 자산들의 목록, 각 자산들의 버전정보 등을 보여준다. 그리고 피처모델과 관계가 설정된 애플리케이션 프로젝트의 목록을 보여준다.

③ SPL 레파지토리 관리도구가 보여주는 피처모델-피처 컨피규레이션, 피처모델-자산목록(버전), 피처 컨피규레이션-애플리케이션 프로젝트 관계를 확인하고, 해당 시점까지 몇 개의 애플리케이션이 만들어졌는지 확인할 수 있다.

b) 피처모델의 릴리즈

플랫폼 관리자는 플랫폼 개발자가 SPL 자산저장소에 등록한 피처모델의 베이스라인 또는 릴리즈를 생성시킬 수 있다.

① SPL 레파지토리 관리도구가 제공하는 작업공간 탐색기를 통해서 SPL 자산저장소에서 작업중인 피처모델의 최신 리비전을 확인한다.

② 플랫폼 개발자가 지정한 리비전 번호를 가진 피처모델을 선택하여 릴리즈 명령을 수행하고 릴리즈 버전을 부여한다.

③ SPL 레파지토리 관리도구가 제공하는 작업공간에서 릴리즈된 피처모델의 등록 여부를 확인한다.

c) 피처모델의 변경

플랫폼 관리자는 현재 자산개발에 사용되고 있는 피처모델에 불가피한 변경이 필요할 경우에는 피처모델의 변경을 진행할 수 있다.

① SPL 레파지토리 관리도구가 제공한 작업공간 탐색기를 통해서 수정하려고 하는 대상 피처모델을 찾는다.

② 도메인공학 도구를 이용하여 피처모델을 편집하고 다시 SPL 자산저장소에 등록하기 위해서 SPL 레파지토리 관리도구를 사용한다.

③ SPL 레파지토리 관리도구 상에서 피처모델과 현재 SPL 자산저장소에 등록되어 있는 자산들의 가변정보와 일관성 검사를 수행한다. 신규로 추가된 피처의 경우에는 관계정보가 존재하지 않음을 확인할 수 있고, 삭제된 또는 이름이 변경된 피처의 경우에는 관계된 어떤 자산의 어떤 가변정보가 상실될 수 있는지를 파악할 수 있도록 그 목록과 함께 상세한 정보를 조회 할 수 있다.

④ 도구가 제공한 정보를 통해서 피처의 삭제로 인하여 영향을 받는 자산들을 확인하고 플랫폼 개발자들과 피처 변경에 따른 현재 개발 중인 자산의 영향 범위에 대해서 검토 회의를 진행하고 최종적으로 변경을 결정한다.

⑤ SPL 레파지토리 관리도구에서 리비전된 피처모델의 버

전을 자산관리 대상으로 설정하면 피처와 가변정보 관계모델을 바탕으로 일관되지 않은 자산들의 상태는 “검사실패”로 표시됨을 확인할 수 있다.

⑥ SPL 레파지토리 관리도구의 작업공간 탐색기에서 자산의 가변정보 빌드가 실패했다는 표시가 나타날 경우에는 문제가 되는 자산항목을 확인한 후 수정한다.

d) 차세대 플랫폼 결정

기술변화와 함께 SPL 플랫폼 개발 시 고려되지 못했던 신규 요구사항이 너무 많아지게 되는 경우가 발생할 수 있다. 이 경우 플랫폼 관리자는 기존의 플랫폼에 피처를 추가하여 확장하는 것 보다는 기존 플랫폼을 동결시키고 차세대 SPL 플랫폼으로 진화시키기로 결정하고 새로운 개발 환경을 설정한다.

① SPL 레파지토리 관리도구를 이용하여 SPL 자산저장소에 접근하여 차세대 SPL 플랫폼 개발을 위한 새로운 작업공간을 생성한다.

② SPL 플랫폼의 작업공간에서 더 이상 피처모델의 변경이나 각 자산의 가변정보가 변경될 수 없도록 ‘자산동결’을 실행한다. SPL 레파지토리 관리도구는 해당 작업공간의 가변정보 관리정보를 읽기전용으로 변경하여 새로운 가변정보가 추가되는 것을 방지한다.

③ 차세대 SPL 플랫폼 작업공간에 기본적으로 차세대로 이어질 수 있는(carry over) 자산을 기존 작업공간에 가져와 차세대 SPL 플랫폼 개발 준비를 하게 된다.

e) 자산의 가변정보 일관성 검증

플랫폼 관리자는 SPL 자산저장소에 저장되어 있는 자산들의 가변정보에 대한 피처 일관성을 확인할 수 있다.

① SPL 레파지토리 관리도구를 이용하여 SPL 자산저장소에서 최신 릴리즈 버전의 피처모델을 찾는다.

② 피처모델을 선택하고 가변정보 일관성 검사 메뉴를 실행한다.

③ 자산공간 탐색기에 보이는 자산 항목들의 상태표시아이콘을 확인하면서 요구사항과 소스코드를 기준으로 일관성에 문제가 있는 자산을 찾을 수 있다.

④ 문제가 있다고 표시된 자산 항목의 경우 피처모델에 없는 피처가 가변정보로 사용되고 있음을 확인 할 수 있다.

6. 결 론

무기체계에 탑재되는 항법소프트웨어의 기능요소들은 다양한 무기체계에 공통적으로 활용될 수 있는 가능성이 높은 것으로 식별되었다. 항법소프트웨어의 표준화 필요성이 소프트웨어제품라인공학을 적용하게 만든 핵심 원동력이다.

본 논문에서는 아키텍처 수준의 표준화와 재사용 가능한 항법소프트웨어 플랫폼을 근간으로 다양한 무기체계에 적용가능하며 소프트웨어의 신속한 개발과 품질 확보, 효율적인 유지보수를 도모하는 SPL 프레임워크 운영개념을 제시하였다.

본 논문에서 제시된 SPL 기반의 항법 소프트웨어 개발 프로세스 및 운용 개념을 토대로 항법소프트웨어 SPL 플랫폼을 개발(아키텍처 설계, 상세설계, 구현)하고 시범적으로 유도무기 및 무인기 체계의 항법소프트웨어의 개발이 진행될 예정이다.

References

[1] Jin-Woo Kim, Woo-Sin Lee, Hack-Joon Kim, So-Yeon Jin, and Se-Hyeon Jo, “A Study of Software Product Line Engineering application for Data Link Software,” *The Korea Society of Communication and Information*, Vol.23, No.12, pp.65-72, 2018.

[2] ISO_IEC_26550:2013(E), Software and systems engineering - Reference model for product line engineering and management.

[3] Kyo C. Kang, Sholom Cohen, James A. Hess, William Novak, and A. Spencer Peterson, “Feature-Oriented Domain Analysis (FODA) Feasibility Study,” CMU/SEI-90-TR-21, 1990.

[4] Klaus Pohl, Günter Böckle, and Frank J. van der Linden, “Software Product Line Engineering: Foundations, Principles, and Techniques,” Springer, 2005.

[5] Sven Apel, Don Batory, Christian Kastner, and Gunter Saake, “Feature-Oriented Software Product Lines: concepts and implementation,” Springer-Verlag Berlin Heidelberg 2013.

[6] C. Kastner, T. Thum, G. Saake, J. Feigenspan, and T. Leich, “FeatureIDE: A Tool Framework for Feature-Oriented Software evelopment,” *Proceedings of the International Conference on Software Engineering*, 2009, pp.611-614.

[7] J.-S. Yang and K. C. Kang, “A Tool for Workflow-based Product Line Software Development,” *Journal of Software and Data Engineering of KIPS*, Vol.2, No.6, pp.377-382, 2013.

[8] Sungwon Kang, Hwi Ahn, Taehyun Park, Pilsu Jung, Kyungmin Ko, and Jaesun Shim, “A Research on Software Platform Engineering -A Development of the Theory and Tool Chain,” K-Valley RED&B Project Final Report, 2014.

[9] Haeng-Kon Kim and Lee-Kyeong Son, “Product Line Development Process for Mobile Software based on Product Line,” *The KIPS Transactions on Computer and Communication Systems*, Vol.12-D, No.3, pp.395-408, 2005.

[10] Ockhyun Paek, Sungkyu Noh, Minkwan Choi, and Taeho Lee, “A Framework for Software Product Line Based Development of Weapon System Software,” *Communications of KIISE*, Vol.36, No.4, pp.19-27, 2018

[11] DAPA(Defence Acquisition Program Administration) Manual 2020-1, “Manual of the Development and Management for the Weapon System Software.”

박 삼 준

https://orcid.org/0000-0003-2614-3483
e-mail : samjoon@add.re.kr
2006년 KAIST 산업 및 시스템공학과(박사)
1990년 ~ 현 재 국방과학연구소
수석연구원
관심분야 : Software Product Line, Software Testing, Machine Learning

이 순 주

https://orcid.org/0000-0001-9473-335X
e-mail : sunjulee@add.re.kr
2014년 KAIST 전기 및 전자공학(석사)
2015년 ~ 현 재 국방과학연구소
선임연구원
관심분야 : 무기체계 효과분석, 무기체계 모델링, 소프트웨어 아키텍처

노 성 규

https://orcid.org/0000-0002-6790-8465
e-mail : snoh@add.re.kr
2008년 한양대학교 전자컴퓨터통신학과
(석사)
2012년 ~ 현 재 국방과학연구소
선임연구원
관심분야 : 소프트웨어공학, 인공지능, 기계학습, 컴퓨터비전

박 병 수

https://orcid.org/0000-0002-8013-6711
e-mail : byungsu_park@add.re.kr
2009년 울산대학교 전기전자제어공학과
(석사)
2010년 ~ 현 재 국방과학연구소
선임연구원
관심분야 : 관성항법, 복합항법

김 도 형

https://orcid.org/0000-0002-4336-6104
e-mail : edooroo@add.re.kr
2019년 KAIST 산업 및 시스템공학과(박사)
2005년 ~ 현 재 국방과학연구소
책임연구원
관심분야 : 국방M&S, 인공지능/기계학습, 소프트웨어 아키텍처

이 인 섭

https://orcid.org/0000-0003-3425-2399
e-mail : inseop@add.re.kr
2013년 광주과학기술원 기전공학부(석사)
2013년 ~ 현 재 국방과학연구소
선임연구원
관심분야 : 관성항법, 복합항법