임베디드SW 개발을 위한 제품계열 개발방법론의 비교 및 평가

채종진○ 윤희병
국방대학교 전산정보학과 임베디드소프트웨어 연구실
jini0083○@paran.com, hbyoon37@hanmail.net

Comparison and Evaluation of Software Product Line Methodology for developing Embedded Software

Jongjin Chae○ Heebyung Yoon
Embedded Software Lab, Dept. of Computer & Information Science, Korea National Defense University

요 약

임베디드SW에 대한 개발이 활발히 수행되고 다양한 플랫폼에서 다른 임베디드SW와 통합되고 제어하는 다양한 요구사항이 발생하고 있으나 HW와의 밀접한 관계 및 재사용성의 요구를 반영하고 있지 못하므로 제품계열 개념을 적용한 접근이 필요하다. 따라서 임베디드 시스템 수명주기 및 시스템, HW, SW간의 관계를 통해 임베디드SW 개발절차를 도출하며, 제품계열 개발방법론 중에서 마르미-EM, FORM, FAST 및 KobrA를 선정하고 도출된 임베디드SW 개발절차와 비교하여 각 개발방법론의 절차상의 차이점, 강점 및 보완 사항을 분석하고 방법론별 특징을 종합한다.

1. 서 론

임베디드SW에 대한 개발이 1970년대 이후 활발히 수행되면서 단순히 독립적인 소규모의 시스템에 내장되어 운영되어오던 과거와는 달리 최근에는 고도의 정밀을 요구하는 동작제어와 함께 복잡하고 다양한 플랫폼 기반 환경에서 작동하거나 다수의 다른 임베디드SW를 통합하고 제어해야하는 여러 가지의 추가 요구사항들이 발생하고 있다. 이러한 임베디드SW는 실시간 처리 지원, 오작동이나 중지가 허용되지 않는 고도의 신뢰성, 제한된 HW자원을 효율적으로 관리할 수 있는 최적화 지원, 그리고 네트워크 및 멀티미디어 처리기능 지원 등을 특징으로 하며, 임베디드SW 개발을 위한 다양한 솔루션이나 개발도구 등 체계적인 방법이 필요하다[1]. 하지만 기존의 개발방법은 임베디드SW가 가지는 HW와의 밀접한 관계 및 여러 유사한 도메인에 대한 고수준의 재사용성의 요구를 반영하지 못하고 있다. 이에 따라 기존의 SW 공학을 통한 SW개발은 비유동적이고, 중복된 시스템의 개발을 이끌어낼 수밖에 없으므로 임베디드SW 개발시 이러한 비효율적인 요소를 제거하기 위해 제품계열(Product Line)의 개념을 적용한 접근이 필요하다. 제품계열 개발방법은 다중시스템 공학으로서 대표되는 것으로 제품개발에 대한 시간 활용의 효율성과 경제적 측면에서 중복 투자의 방지와 자산에 대한 재사용의 확대를 위해 출현하였다. 제품계열 개발방법의 구성은 공통의 특성을 가진 제품들을 도메인 영역으로 설정하고 제품계열을 형성하여 이러한 도메인에서 핵심자산(Core Asset)을 구축하는 도메인 엔지니어링(Domain Engineering)과 구축된 핵심자산을 활용하여 사용자 요구를 만족시키는 제품을 생산하는 애플리케이션 엔지니어링(Application Engineering)으로 나누어진다. 즉, 대상이 되는 제품과 관련된 공통의 핵심자산을 구축하고 활용하여 제품을 개발함으로써 경제성을 높이고 재사용성을 향상시키며 수요를 충족시키는 것이다[2]. 미국의 CMU SEI (Carnegie Mellon University Software Engineering Institute)와 독일의 IESE(Institute for Experimental Software Engineering) 연구소를 중심으로 임베디드SW 개발을 위한 개발 프레임워크로서 많은 연구가 진행 중에 있다[3][4][5].

2. 임베디드 소프트웨어 개발절차

2.1 임베디드 시스템 수명주기

임베디드 시스템 개발 프로젝트의 수명주기는 그림 1에서 보는 바와 같이 크게 6단계로 나눌 수 있으며 각 단계에 대응하는 검증 절차들을 가진다[6].

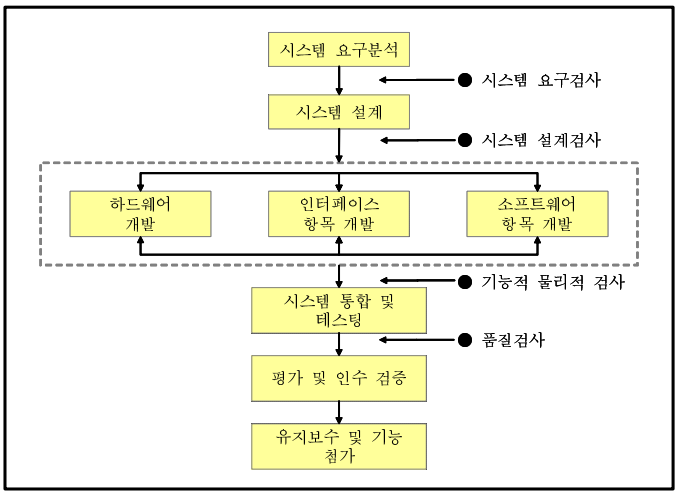

그림 1. 임베디드 시스템 개발 프로젝트 수명주기

특히 임베디드 시스템의 설계에는 기본적으로 하드웨어 요소와 소프트웨어 요소가 동시에 포함되므로 개발자들은 어떤 기능을 하드웨어로 해결해야 하고 어떤 부분을 소프트웨어로 해결해야 하는지를 최우선적으로 결정해야 한다. 임베디드 시스템 및 임베디드SW의 효율적인 개발을 위해서는 설계 단계에서 HW 설계, SW 설계 그리고 필요에 따라 인터페이스 영역 설계 단계가 HW 및 SW 동시 설계 개념으로 병행 수행되어야 한다.

시스템 설계를 통하여 HW 개발과 SW 개발로 구분하여 개발된다. SW 개발은 SW 요구조건 분석, SW 예비설계, SW 상세설계, SW 코드 구현, 테스트 그리고 통합의 개발과정을 거쳐 완성된다. 또한 HW 개발은 HW 요구조건 분석, HW 예비설계, HW 상세설계, HW 제작, 테스트 그리고 통합의 개발과정을 거쳐 완성된다. 이러한 HW와 SW의 개발과정은 독립적인 방법으로 개발되는 것이 아니라 체계공학의 시스템 설계 내용과 일치되는지를 지속적으로 분석하면서 이루어진다.

이것이 일반 정보체계 개발과정과 가장 큰 차이점이다. HW와 SW 개발이 완료되면 시스템 통합 및 테스트를 통하여 사용자 요구사항을 만족하는가를 확인하고, 운용 테스트 및 평가를 통하여 사용자에게 인계된다.

2.2 임베디드SW 개발절차

임베디드SW 특성과 2.1절에서 언급한 임베디드 시스템의 수명주기를 기반으로 임베디드SW의 개발절차 부분을 더 구체화하고 일반화하여 나타내면 그림 2와 같이 나타낼 수 있다.

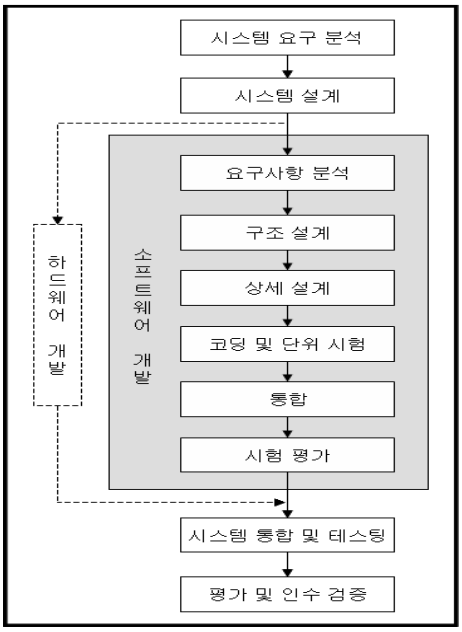

그림 2. 임베디드SW 개발절차

임베디드SW 개발절차를 보면 시스템 요구사항 분석 및 설계 이후 HW 개발과 SW 개발로 나누어 동시 진행하여 개발한 이후 시스템 통합 및 테스팅과 평가 및 인수검증을 수행한다. 여기서 HW 개발부분은 3장의 개발방법론 비교에서 HW 동시 설계의 반영 여부만 확인하므로 개발 단계를 세분화하지 않았으며, HW 개발을 제외한 나머지 부분에 대하여 그림 2의 임베디드SW 개발절차를 기준으로 각 단계에 대한 연관성을 비교 분석한다.

3. 개발방법론 비교

3.1 마르미-EM

마르미-EM은 2004년부터 2006년까지 한국전자통신연구원(ETRI)이 한국형 개발방법론인 마르미(MaRMI: Magic and Robust Methodology Integrated) 시리즈 중 임베디드 시스템용을 개발한 것으로 구성을 보면 프로세스 경로는 수명주기를 고려하여 응용 개발, 미들웨어 개발, 터미널 개발, 제품 개발로 나누어져 있고 제품계열 개념을 고려하여 자산 생성과 제품 개발로 구분하고 있다. 또한 작업 지침을 보면 프로세스 경로 구성의 단위 작업을 정의하며, 작업 수행은 영역활동 지침에 기술되어 있고, 기술적 측면의 기법서와 산출물 양식을 선택적으로 제공하고 있다. 마르미-EM 개발방법론에서는 전체 개발 프로세스 경로를 액티비티 다이어그램으로 표시하고 영역 활동은 도표를 통해 설명하여 이해하기 쉽게 하였다[7].

마르미-EM 개발방법론을 임베디드SW 개발 절차와 비교해 보면 그림 3과 같다.

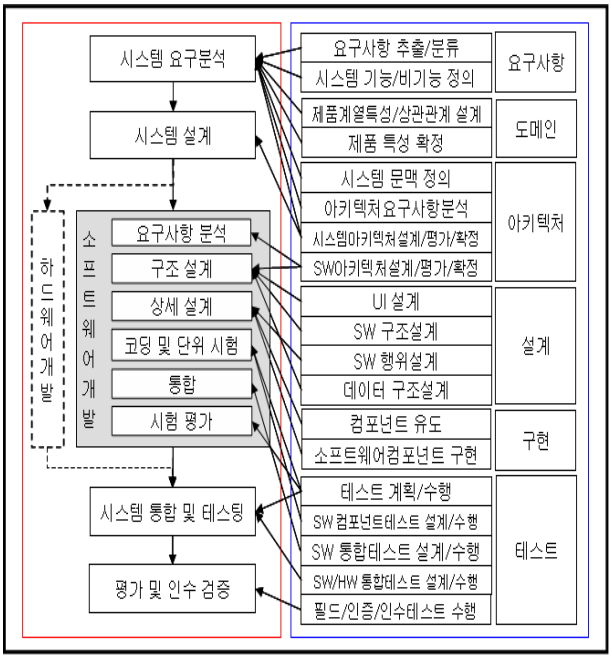

그림 3. 임베디드SW 개발절차와 마르미-EM 절차 비교

마르미-EM은 전체 개발 절차가 임베디드SW 개발절차의 전 단계를 수행하고 있으며, 각 영역 활동의 하위 과정을 세분화하고 도메인 영역 활동에서 제품 및 제품계열의 특성과 관계를 결정하고, 아키텍처 영역 중 시스템 설계와 매핑되는 "시스템 아키텍처 설계, 평가 및 확정"에서 개발 프로세스의 경로가 결정되고 있다.

또한 마르미-EM은 시스템 요구분석 단계를 세분화하여 개발대상의 특성에 따라 프로세스 경로의 결정이 용이하도록 구성되어 있고, SW 재사용을 위한 지침서를

양식과 함께 언어별(Java, C) 코딩 지침을 제공하고 있으며, 컴포넌트 개발 작업시 컴포넌트 추출, 상관관계 결정 및 코드 구현과 테스트를 수행하나 컴포넌트를 구현하는 방법에 대한 자세한 설명이 부족하다.

3.2 FORM

FORM(Feature-Oriented Reuse Method)은 1998년 포항공대에서 개발하였고, FODA(Feature-Oriented approach to Domain Analysis)를 확장하여 마케팅, 제품계획, 휘처 모델을 기반으로 한 핵심자산의 재사용과 아키텍처 설계 및 객체 지향 컴포넌트 개발을 지원하며, 방법론의 구성은 8개의 활동으로 이루어진 핵심자산을 개발하는 영역공학과 3개 활동으로 이루어진 특정 애플리케이션을 개발하는 응용 공학으로 나누어진다[8].

FORM은 사업적/공학적 측면의 분석을 통하여 개발자 및 이해관계자들에게 정보를 제공하고, 마켓 분석 결과를 기반으로 아키텍처와 컴포넌트 개발에 필요한 품질요소를 추출하며, 휘처 중심의 분석 방법을 이용하여 영역에서의 공통점과 차이점을 분석하고 있다[8][9][10]. 이러한 FORM의 개발 절차를 임베디드SW 개발절차와 비교해보면 그림 4와 같이 나타낼 수 있다.

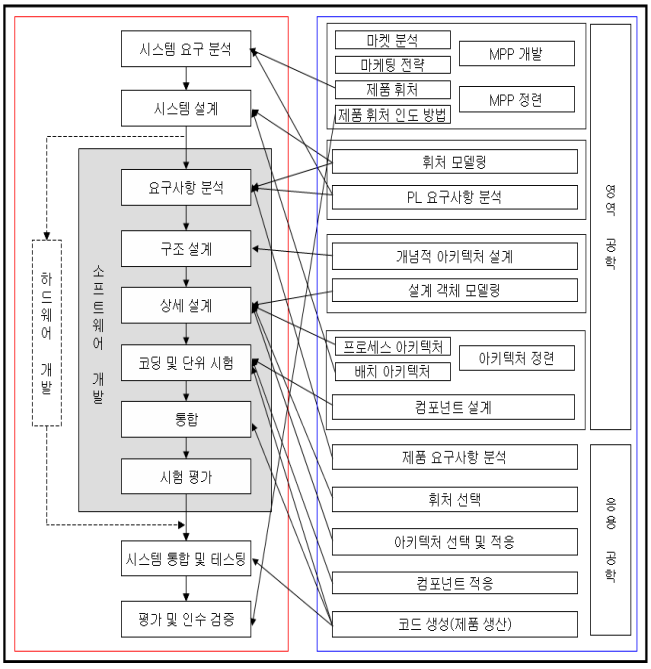

그림 4. 임베디드SW 개발절차와 FORM 절차 비교

개발 절차를 비교해 보면 제품의 사업적 측면인 마케팅 부분과 “평가 및 인수 검증 단계”에 대한 수행 과정은 임베디드SW 개발절차에 없으며, 설계 과정의 상세화는 각 단계의 정련 과정을 거쳐 이루어지고 임베디드SW 개발절차의 HW와 SW를 구분이 아키텍처 정련 과정(프로세스 아키텍처 및 배치 아키텍처)에서 수행된다.

FORM은 시장에 대한 사전 분석으로 변화에 쉽게 대응이 가능하고, 비기능적 품질요소에 영향이 큰 임베디드 시스템에 대한 품질요소를 유도하여 효율적이나, 임베디드SW 개발절차와 비교하면 HW와 SW의 동시 설계 및 개발에 대한 개념과 휘처 모델을 통해 식별된 가변 요구사항에 대해 아키텍처 단계에서의 관리방법에 대한 설명의 보완이 필요하다.

3.3 FAST

FAST(Family-oriented Abstraction, Specification and Translation)는 1992년 미국의 AT&T 연구소가 개발하였으며 제품 패밀리를 정의하고 애플리케이션 생성을 위한 환경 개발을 지원하고 있으며 PASTA(Process and Artifact State Transition Abstraction) 모델을 통해 프로세스의 일관성과 통제성을 유지한다[11][12].

FAST 개발방법론은 패밀리의 투자 가치를 식별하는 영역자격, 제품 생산을 위한 설비 및 환경에 대한 투자를 수행하는 영역공학, 빠른 제품 생산을 위한 영역공학을 활용하는 응용공학 그리고 응용공학 환경으로 구성되어 있다. 그리고 영역 자격을 통하여 제품계열의 적용여부를 판별하여 개발자의 노력을 경감시키고, 영역공학에서는 도메인 모델을 통하여 응용공학 환경과 프로세스를 명세하여 응용공학에서의 반복 작업이 감소하며, 일관성 있는 공통성의 식별을 통하여 재사용 컴포넌트의 획득을 지원하고 있고, 패밀리 변경을 통한 진화 및 유지보수를 지원한다[10][11].

임베디드SW 개발절차와 FAST의 절차를 비교하면 그림 5와 같이 나타나며, 제품계열의 도입 여부를 결정하는 영역 자격, 응용공학의 기반을 구축하는 응용공학 환경 구현, 그리고 진화 및 유지보수에 대한 측면의 패밀리 변경은 임베디드SW 개발절차의 범위를 벗어나는 부분이다. 개략적으로는 임베디드SW 개발절차에 상응하나 응용공학에서 세분화된 절차가 부족하고, 시험 및 평가에 대한 절차가 인수 및 운영 지원에서의 사용자 점검 외에는 불명확하다.

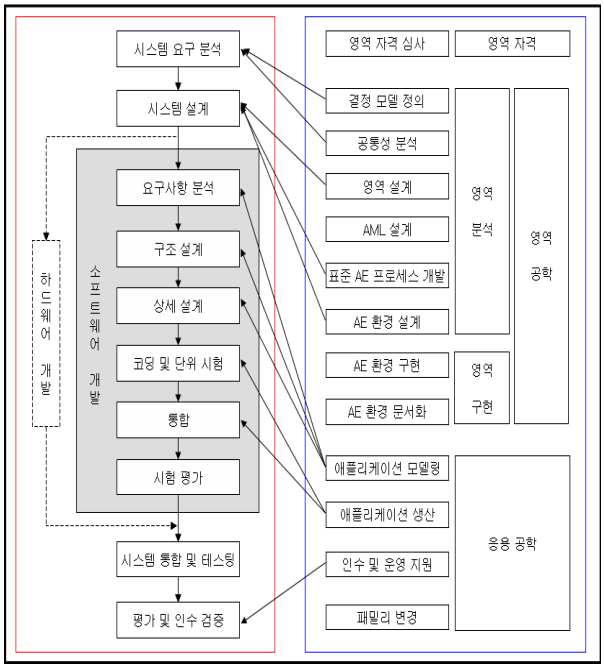

그림 5. 임베디드SW 개발절차와 FAST 절차 비교

또한 제품계열 도입을 결정이후 단계가 진행되므로 공통성 및 모델을 설계에 반영하는 과정과 가장 적절한 기반의 구축 과정은 임베디드SW 개발절차보다 세분화되어 제공되고 문서화에 대한 절차가 명시되어 있으나, 응용공학 과정에서 모델 정련시 사용자 검사가 오히려 제품 개발을 지연시키거나 지루한 반복이 발생할 수 있으며, 실제 적용할 기술적 방안이나 세부적인 지침의 보완이 요구된다.

3.4 KobrA

KobrA(Component-Based Application Development)는 1999년부터 2001년까지 IESE(Institute for Experimental Software Engineering) 연구소에서 개발한 컴포넌트 기반의 제품계열 개발방법론으로 컴포넌트 구현을 위해 UML을 사용한다[13].

KobrA의 구성은 시스템의 가변성이 표현 가능한 일반적인 자산을 생성하는 과정인 프레임워크공학과 이를 통해 생성되는 프레임워크, 프레임워크를 기반으로 컴포넌트 조립을 통해 제품을 생산하는 응용공학으로 나누어진다. 또한 KobrA는 핵심자산의 메타 데이터를 잘 정의하고 제품계열의 변경관리를 위한 진화 그래프를 사용하며, 각 단계에서의 세부 지침과 명세화, 실체화 그리고 실현의 단계에서 정련 작업을 위한 피드백으로 핵심자산을 견고화시키고 산출물을 정확히 정의한다[10][13].

임베디드SW 개발절차와 비교해 보면 그림 6과 같이 나타난다.

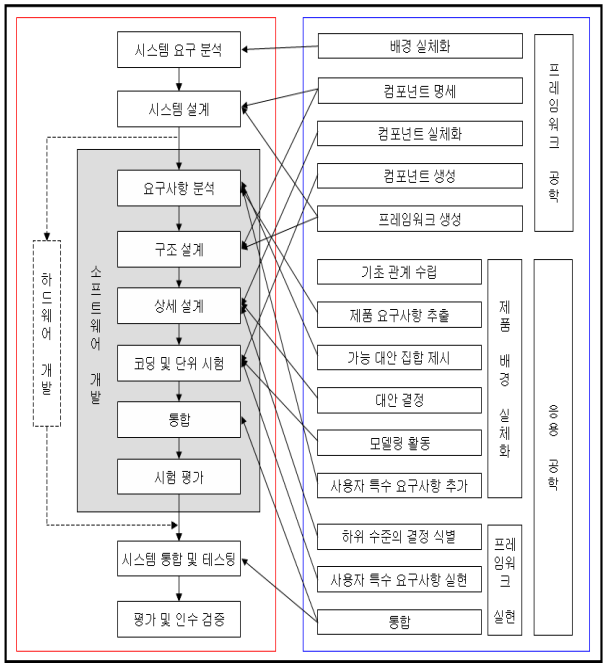

그림 6. 임베디드SW 개발절차와 KobrA 절차 비교

프레임워크 생성을 위한 프레임워크공학에서 시스템에 대한 요구분석에서 부분적인 소프트웨어의 개발까지가 이루어지므로 임베디드SW 개발절차의 상당한 과정이 진행되고, 이미 생성된 프레임워크를 활용하므로 응용공학에서는 임베디드SW 개발절차 중 SW개발에 대한 분석에서 평가까지를 수행하고 있다. 그리고 별도의 통합 단계를 두고 있으나 이것은 SW에서 시스템 통합까지의 포괄적인 통합에 대하여 제시하는 것이고, 시험이나 평가에 대한 별도의 절차가 없이 명세나 실체화 및 대안 결정 과정에 포함이 되어 있다. 또한, 고품질의 컴포넌트 기반 제품 개발에 대해 체계적인 접근을 지원하고 특정 프로그래밍 언어나 기술에서 독립적이나, 제품계열 요구사항에서 제품계열 아키텍처로 변환하는 것에 대한 세부적인 지침이나 절차는 보완이 필요하다.

3.5 개발절차 비교 내용 종합

제품계열 개발방법론인 마르미-EM, FORM, FAST 및 KobrA의 개발절차를 임베디드SW 개발절차와 비교한 결과를 다음과 같이 종합할 수 있다.

임베디드SW 개발절차와의 유사성에서는 각 단계를 모두 수행할 수 있게 세분화된 마르미-EM이 가장 높으며, FORM과 KobrA는 일부 매핑이 부족한 부분이 있고, FAST는 임베디드SW 개발절차 및 FAST 자체에 대한 매핑이 부족하여 유사성이 상대적으로 낮게 나타났다. 아키텍처에 대한 고려는 모든 개발방법론이 반영하고 있으나, 임베디드 시스템 구축시 고려해야 하는 HW에 대한 동시 설계를 마르미-EM과 FORM은 아키텍처 부분에서 반영하고 있으나, KobrA와 FAST는 불명확하거나 나타나 있지 않다.

통합에서 인수까지의 단계(통합, 시험평가, 시스템 통합 및 테스팅, 평가 및 인수검증)에 있어서는 개발방법론별로 약간씩 차이가 있다. 마르미-EM은 SW와 HW 각각에 대하여 절차를 잘 구분하고 있으나, FORM과 FAST는 통합부분이 제품생산이나 애플리케이션생산에 내포되고 평가나 테스트에 관한 사항은 불명확하다. 다만, FAST는 평가 및 인수에 대한 절차는 제시하고 있다. KobrA는 통합을 별도의 절차로 제시하고 있으나 SW와 시스템에 대한 구분이 불명확하고, 평가나 테스트에 대한 절차는 별도로 제시되어 있지 않다.

각각의 개발방법론에서 제품계열을 반영하는 단계를 보면 마르미-EM은 도메인, FORM은 MPP 개발, FAST는 영역 자격, KobrA는 배경 실체화의 부분으로 모두 초기 단계에 반영하고 있다. 개발방법론별로 고유한 특성을 살펴보면 마르미-EM은 다양한 경로의 제공과 재사용, FORM은 시장분석 및 휘처, FAST는 영역 자격, 응용 환경과 문서화 그리고 KobrA는 프레임워크 및 컴포넌트 기반 개발로 표 1과 같이 요약할 수 있다.

표 1. 개발절차 비교 결과

	절차 유사성	HW 설계	아키텍처	통합/테스트 단계 구분	제품계열 반영
마르미-EM	높음	반영	반영	HW/SW 구분	도메인
FORM	보통	반영	반영	불명확	MPP 개발
FAST	보통	불명확	반영	불명확	영역 자격
KobrA	낮음	불명확	반영	불명확	배경 실체화

4. 결 론

임베디드SW 개발절차와 제품계열 개발방법론의 비교를 위해 국내에서 개발된 마르미-EM과 FORM, 국외에서 개발된 FAST와 KobrA를 선정하고 비교하였다. 비교 결과 임베디드SW 개발절차에 대해서 마르미-EM이 적합한 것으로 분석되었으며, 나머지는 HW를 고려한 통합 및 테스트 부분에 있어서 세분화가 필요한 것으로 분석되었다. 제품계열이나 아키텍처는 대상 개발절차 모두 잘 반영한 것으로 분석되었으며, 각각의 방법론이 가지는 고유한 특징 및 절차를 비교할 수 있었다. 이러한 비교를 통하여 제품계열 개발방법론을 임베디드SW 개발시 적용할 경우 절차상의 차이점과 이에 대해 보완이 필요한 사항 및 강점을 살펴볼 수 있었다.

참고문헌

[1] Ivica Crnkovic, "Component-based Software Engineering for Embedded Systems," ICSE'05, pp. 712-713, 2005.

[2] 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.

[3] Paul Clements, Linda Northrop, *Software Product Lines practices and patterns*, Addison Wesley, 2002.

[4] CMU SEI Homepage http://www.sei.cmu.edu/

[5] IESE Homepage http://www.iese.fraunhofer.de/fhg/iese/

[6] Software Technology Support Center, *Guidelines for Successful Acquisition and Management of Software -Intensive Systems(GSAM),* Department of the Air Force, Ver. 3.0, 2000.

[7] ETRI, Magic and Robust Methodology Integrated (MaRMI)-Ⅳ, EMMA Ver 1.0, 2006.

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

[9] Kyo Chul Kang, Jaejoon Lee, Byungkil Kim, Sajoong Kim, "Software Product Line Engineering for Productivity Improvement in Embedded System Development," Communications of the Korea Information Science Society, Vol. 22, No.6, pp. 25-35, 2004.

[10] Matinlassi M., "Comparison of Software Product Line Design Methods," Proceedings of 26th ICSE, pp. 127-136, 2004.

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

[12] David Weiss, "Family-Oriented Abstraction, Specification, and Translation: The FAST Process," AT&T Bell Laboratories, 1995.

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