임베디드 소프트웨어를 위한 기능 중심 평가 모델

(A Functionality-based Evaluation Model for Embedded Software)

최 현 미 † 성 아 영 †† 최 병 주 ††† 김 재 웅 ††††
(Hyunmi Choi) (Ahyoung Sung) (Byoungju Choi) (Jaewoong Kim)

요 약 임베디드 소프트웨어는 대상 프로세서에 내장되어 특정 기능을 수행하기 위한 소프트웨어로, 한번 탑재되면 수정하는 것이 쉽지 않기 때문에 임베디드 소프트웨어의 기능에 대한 평가는 중요하다. 임베디드 소프트웨어는 탑재할 대상 플랫폼 및 정의된 기능 요구사항에 맞게 구성되고 맞춤하여 개발되므로 그 종류가 매우 다양하고, 임베디드 소프트웨어를 구성하는 여러 요소들이 매우 밀접하게 연결되어 있기 때문에 이를 평가한다는 것은 쉽지 않다. 본 논문에서는 이러한 임베디드 소프트웨어의 특성을 반영한 평가 모델을 제안하고 실제 임베디드 소프트웨어에 본 평가 모델을 적용한 결과를 기술한다.

키워드 : 임베디드 소프트웨어, 임베디드 소프트웨어 테스트

Abstract Embedded software is mounted on the target processor for controlling its dedicated functions. To evaluate functions of embedded software is important because it is intricate to modify embedded software once embedded. However, it is difficult to evaluate embedded software because it varies in kinds, which is customized into each target platform and functional requirements, and all the elements within are tightly coupled. In this paper, we propose the evaluation model reflecting these unique features of embedded software. We present the results of the case studies by applying the proposed model to practical embedded software.

Key words : Embedded Software, Embedded Software Test

1. 서 론

임베디드 소프트웨어란 핸드폰, 디지털 카메라, 마이크로 오븐과 같은 가전제품에서부터 자동차, 비행기와 같은 디지털 제어 시스템 및 세이프티-크리티컬(safety-critical) 시스템에 이르기까지의 매우 광범위한 전자 제품에 내장되어, 그 제품의 특정(dedicated) 기능을 제어하기 위해 수행되는 소프트웨어[1]를 의미한다.

최근 임베디드 시스템 산업은 기술 개발 및 시장 경쟁에 의해 빠르게 변화하고 있고, 마이크로프로세서(micro-processor)의 가격이 낮아지고, 시스템의 소형화 및 고성능화가 진행되며, 제품 경쟁력의 핵심이 하드웨어 생산에서 소프트웨어 기술로 이동함에 따라 제품의 가치가 소프트웨어에 의해 좌우되는 기술 집약적 고부가가치 산업으로 발전하고 있다. 임베디드 소프트웨어에 대한 관심 및 비중이 커짐에 따라, 임베디드 소프트웨어의 평가 또한 중요한 이슈로 부각되고 있지만, 임베디드 소프트웨어의 다양성과 확고한 표준의 부재로 임베디드 소프트웨어의 개발에 비해 임베디드 소프트웨어에 대한 평가는 그 진척이 미흡한 상황이다.

임베디드 소프트웨어는 대상 시스템에 탑재되어 그 제품의 특정 기능을 제어하기 위해 수행되는 소프트웨어로, 임베디드 소프트웨어가 나타내고자 하는 기능이 올바르게 동작하는지 확인하는 것은 매우 중요하다. 특히 임베디드 소프트웨어는 일반 패키지 소프트웨어에 비해 다음과 같은 특성을 지니고 있기 때문에, 임베디드 소프트웨어의 특성을 고려하여 대상 임베디드 소프트웨

· 본 논문은 한국정보통신기술협회의 벤치마킹 테스트 사업 지원으로 수행되었음
† 정 회 원 : 이화여자대학교 컴퓨터학과
hmchoi@ewhain.net
†† 학생회원 : 이화여자대학교 컴퓨터학과
aysung@ewhain.net
††† 종신회원 : 이화여자대학교 컴퓨터학과 교수
bjchoi@ewha.ac.kr
†††† 정 회 원 : 한국정보통신기술협회 소프트웨어인증센터 연구원
jwkim@tta.or.kr
논문접수 : 2004년 12월 30일
심사완료 : 2005년 10월 14일

어의 기능을 평가해야 한다.

첫째, 임베디드 소프트웨어는 어플리케이션만으로 구성된 펌웨어와 같은 간단한 형태에서부터 각종 디바이스 드라이버 및 운영체제, 미들웨어, 어플리케이션과 같이 다양한 소프트웨어 계층을 모두 포함한 복잡한 형태에 이르기까지 매우 다양한 형태로 존재한다. 따라서 이를 반영하여 임베디드 소프트웨어의 기능을 평가할 수 있어야 한다.

둘째, 임베디드 소프트웨어는 탑재할 대상(target) 플랫폼 및 정의된 기능 요구사항에 맞게 구성하고 맞춤화하여 개발된다. 예를 들어, 임베디드 운영체제를 포함하는 임베디드 소프트웨어의 경우, 특정한 용도에 맞추어 최소한의 리소스로 의도된 운영체제 기능만을 구현하여 사용하게 된다. 또한 임베디드 소프트웨어 내에는 하드웨어에 의존적인 코드도 존재하게 된다. 따라서 이러한 다양한 구성을 모두 커버할 수 있는 기준을 가지고 평가할 수 있어야 한다.

셋째, 임베디드 소프트웨어는 각 요소가 매우 밀접하게 연결되어 있다. 즉, 어플리케이션 코드가 하드웨어를 직접 제어하는 코드와 함께 존재하며, 실시간 운영체제를 사용하는 임베디드 소프트웨어의 경우 운영체제 코드와 어플리케이션 코드가 혼재된 형태로 존재하기도 한다. 이렇듯 임베디드 소프트웨어는 이질적인 요소들이 결합되어 하나의 기능을 구현하기 때문에, 임베디드 소프트웨어의 기능을 평가할 때, 어플리케이션 로직 자체의 평가 외에 추가적인 평가 관점이 필요하다.

마지막으로, 임베디드 소프트웨어는 하드웨어에 탑재되어 수행되는 소프트웨어이다. 따라서 하드웨어에 탑재되어 실행될 때에 예상하지 못한 상황이 발생할 수 있다.

본 논문에서는 위의 임베디드 소프트웨어의 특성을 반영하여 임베디드 소프트웨어의 기능을 체계적이고 객관적으로 평가할 수 있는 평가 모델을 제안하고자 한다. 또한 실제 임베디드 소프트웨어에 본 평가 모델을 적용하여 수행한 기능평가 결과를 제시한다.

본 논문의 구성은 다음과 같다. 2절에서는 관련 연구를 기술하고, 3절에서는 평가 모델의 입력이 되는 임베디드 소프트웨어 및 평가 모델의 평가 영역에 대한 정의를 기술하고, 4절에서는 제안하는 임베디드 소프트웨어 평가 모델의 전체 구조 및 구성에 대해서 기술하고, 5절에서 사례 연구를 기술하며, 마지막으로 6절에서 결론 및 향후 연구에 대해 기술한다.

2. 관련 연구

본 절에서는 관련연구로써 소프트웨어를 평가하기 위한 품질 모델과 임베디드 소프트웨어의 핵심인 임베디드 운영체제 및 관련 표준을 제시하며, 이로부터 본 논문에서 제안하는 임베디드 소프트웨어 기능 평가 모델의 근거를 기술하고자 한다.

본 논문에서는 임베디드 소프트웨어의 기능을 평가할 수 있는 평가 모델을 개발하고자 한다. 소프트웨어의 품질을 평가하기 위한 모델로써 가장 대표적으로 이용되는 것은 ISO/IEC 9126[2]이다. ISO/IEC 9126은 소프트웨어의 품질을 평가할 수 있는 측정 기준을 제시하기 위해, 품질 특성 및 부특성과 이러한 특성을 위한 내부 및 외부 메트릭에 대해서 정의하고 있다. ISO/IEC 9126의 기능성 평가는 소프트웨어의 기능을 적합성, 정확성, 상호운용성 등과 같은 관점에 따라, 소프트웨어의 내부적인 구성과는 관계없이 외부로 표출되는 결과를 기준으로 기능을 평가하기 위한 품질 관점을 제공하기 때문에, 서론에서 언급한 임베디드 소프트웨어의 특성에 대한 기능을 평가 하는 데는 어려움이 있다. 따라서 본 논문에서 제안하는 평가모델은 임베디드 소프트웨어의 기능을 구체적으로 평가할 수 있도록 함을 목적으로 한다.

임베디드 소프트웨어는 하드웨어 및 운영체제를 비롯한 여러 소프트웨어가 긴밀하게 연관된 형태로써 존재한다. 임베디드 제품의 많은 부분을 차지하는 임베디드 운영체제는 크게 임베디드 리눅스, Windows CE .NET, Windows XP Embedded, Palm OS와 같은 범용 임베디드 운영체제와 VxWorks, pSOS, QNX, VRTX, uC/OS-II와 같은 실시간 운영체제(RTOS)등 매우 다양하다. 한편 이들에 대한 표준화를 위하여 범용 임베디드 운영체제 부문의 경우, EL/IX[3], ELCPS[4], KELPS[5], CELF[6], Emblix[7] 등과 같은 임베디드 리눅스 표준 및 표준화 활동들이, RTOS 부문의 경우, ITRON[8], POSIX 1003.4[9] 등과 같은 실시간 운영체제 기반의 표준 및 표준화 활동들이 있다. 그러나 임베디드 소프트웨어를 평가하기 위해서는 이들 운영체제 자체에 대한 평가 이외에도 하드웨어를 제어하는 코드, 하드웨어와의 결합 후 실행 관점 등 고려해야할 부분이 있다.

본 임베디드 소프트웨어 평가모델은 표 1에서처럼 임베디드 운영체제 기능 평가를 위하여 임베디드 운영체제의 API와 관련된 표준인 ELCPS, KELPS, POSIX의 API그룹을 평가항목으로 포함하며, 이외에도 하드웨어를 제어하는 코드, 하드웨어와의 상호작용 등을 평가할 수 있도록 한다.

3. 임베디드 소프트웨어 구성

서론에서 기술한 임베디드 소프트웨어의 특성을 반영하여 임베디드 소프트웨어의 기능을 체계적이고 객관적으로 평가할 수 있도록 임베디드 소프트웨어를 분석하

표 1 평가 모델과 기존 임베디드 운영체제

	하드웨어 제어 부분	OS 모듈 부분	OS 몸체 부분	응용 부분
POSIX[9]		O		
ELCPS[4], KELPS[5]		O		
RTOS 문서[13]	O	O	O	
임베디드 리눅스[14]	O	O		
일반 운영체제[15]	O	O		
임베디드 시스템 설계 문서[16,17]	O	O	O	O
펌웨어 문서[18]	O		O	

고자 한다.

3.1 임베디드 소프트웨어의 구성 요소

임베디드 소프트웨어는 그림 1에서처럼 AP, OP, HP로 구성하며. 각 구성 요소는 다음과 같다.

- 어플리케이션에 의존적인 부분(Application-dependent Part, AP)

임베디드 시스템의 요구사항 자체를 구현한 어플리케이션 부분이다.

- 임베디드 운영체제에 의존적인 부분(OS-dependent Part, OP)

운영체제를 사용하는 임베디드 소프트웨어에서, 운영체제 모듈 자체 또는 운영체제의 기능을 이용하기 위한 구현 부분을 의미한다.

- 하드웨어 제어에 의존적인 부분(Hardware-dependent Part, HP)

하드웨어에 접근하여 직접적으로 이를 제어하는 구현 부분을 의미한다.

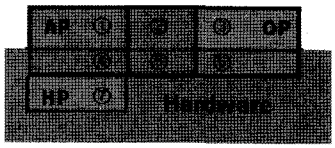


그림 1 임베디드 소프트웨어의 구성

임베디드 소프트웨어는 제품에 따라 AP, HP, OP가 다양하게 연결되어 있다. 이를 체계적으로 평가하기 위하여 임베디드 소프트웨어를 구성하는 구성요소를 다시 ①~⑦로 다음과 같이 세분화할 수 있다.

① 기능을 구현한 어플리케이션 로직 자체를 의미하는 부분이다.

② OS의 API 호출 또는 어플리케이션을 개발하기 위해 맞춤된 OS모듈 부분이다.

③ 변형되지 않은 OS모듈 자체를 의미하며, 운영체제의 몸체(Body)에 해당하는 부분이다.

④ 어플리케이션의 수행을 위해 하드웨어에 접근하는 부분이다.

⑤ ⑥을 수행하기 위한 OS의 API 호출 또는 맞춤된 OS모듈이다.

⑥ 하드웨어에 OS를 포팅(Porting)하기 위한 환경 설정(Configuration) 부분과 OS가 메모리, 프로세서, 입출력 디바이스 등의 하드웨어에 접근하는 부분이다.

⑦ ④와 같이 어플리케이션에 특화된 구현을 제외한 나머지 부분으로 하드웨어 초기화 같은 부분이 해당된다.

세분화된 구성 요소에 따른 평가 내용은 표 2와 같다. 본 평가모델에서는 어플리케이션 기능 자체에 대한 평가(①) 및 운영체제 자체의 기능 구현(③), 운영체제에서 내부적으로 하드웨어에 접근하는 부분(⑥)은 제외한다. ①의 경우, 일반 소프트웨어에서의 어플리케이션 기능 평가와 차이가 없으며, ③과 ⑥은 운영체제 자체에 대한 평가를 의미하기 때문이다. 본 모델의 평가 대상은 어플리케이션에서 사용되는 OS 모듈과 OS의 기능(②,④)을 위한 OP 평가와 어플리케이션에서 하드웨어를 제어 및 접근하는 기능(④,⑦)을 위한 HP 평가로 그 범위를 제한한다.

표 2 임베디드 소프트웨어의 구성 요소와 평가 대상

구성요소	평가 내용	평가 분야
①	어플리케이션 기능이 올바로 구현되었는가?	해당사항 없음
②	어플리케이션 기능 구현에 필요한 OS기능 이용을 올바로 구현하였는가? 만족해야 할 조건에 따라 구현하였는가?	OP 평가
③	OS가 올바로 구현되었는가?	해당사항 없음
④	어플리케이션 기능 구현을 위한 하드웨어 접근 코드를 올바로 구현하였는가? 만족해야 할 조건에 따라 구현하였는가?	HP 평가
⑤	어플리케이션 기능 구현을 위한 OS의 하드웨어 접근을 올바로 구현하였는가? 하드웨어를 추상화한 OS계층을 위해 만족해야 할 조건에 따라 구현하였는가?	OP 평가
⑥	메모리 접근과 같이 OS에서 하드웨어를 접근하는 부분이 올바로 구현되었는가? OS를 하드웨어에 포팅 할 수 있도록 환경 설정을 올바로 하였는가?	해당사항 없음
⑦	어플리케이션 기능에 특화된 구현 이외의 하드웨어 접근 코드가 올바로 구현되었는가?	HP 평가

3.2 임베디드 소프트웨어의 예

그림 2에서 보는 것과 같이 임베디드 시스템은 일반적으로 펌웨어(예. Boot loader와 같은 시스템 초기화 소프트웨어), 디바이스 드라이버(예. LCD 모니터, 키패드, USB, 시리얼 포트), 임베디드 운영체제(Embedded OS) (예. Embedded Linux, RTOS 와 같은 임베디드 운영체제), 임베디드 응용(예. Pocket PC의 게임, PDA의 스케줄 관리 프로그램)과 같은 여러 임베디드 소프트웨어가 계층(Layer)적으로 존재하는 시스템이다. 각 계층별 임베디드 소프트웨어는 3.1절에서 정의한 AP, OP, HP로 이루어지며, 대상 및 계층의 특성에 따라 AP, OP, HP의 구성 비율이 달라진다. 그림 2에서 보는 것과 같이 각 계층별 그 구성을 살펴보면 다음과 같다. 펌웨어 및 디바이스 드라이버의 경우에는 하드웨어에 의존적인 부분(④, ⑦)과 이를 구현하기 위해 운영체제의 서비스를 활용 하는 부분(⑤, ⑦)이 존재한다. Embedded OS의 경우에는 운영체제를 의미하는 것으로 운영체제 바디(Body)에 해당하는 부분(③)과 ③이 내부적으로 하드웨어에 접근하는 부분(⑥)이 존재한다. 임베디드 어플리케이션의 경우에는 임베디드 어플리케이션의 기능 자체를 구현한 부분(①)과 운영체제의 서비스를 이용하는 부분(②, ⑤), 하드웨어에 의존적인 부분(④, ⑦)이 존재한다.

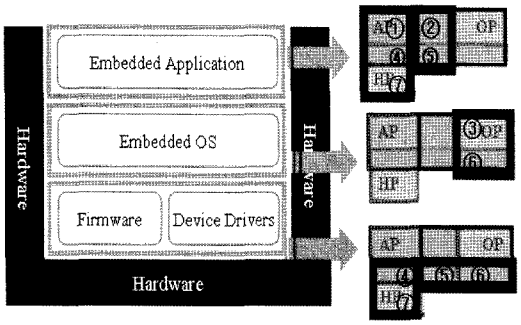


그림 2 임베디드 소프트웨어의 계층 및 계층별 구성의 예

3.3 임베디드 소프트웨어의 평가 영역

본 평가 모델은 HP 평가, OP 평가, BC(Base Consideration) 평가, IC(Interactive Consideration) 평가의 네 가지 평가 영역으로 구성하며 그 세부적인 기술은 다음과 같다.

- BC 평가

  BC 평가는 임베디드 소프트웨어가 기본적으로 만족시켜야 할 조건들을 올바르게 구현하고 있는가를 평가한다.

- HP 평가

  임베디드 소프트웨어는 하드웨어에 탑재되기 때문에 하드웨어의 초기화 및 제어하는 부분을 평가한다.

- OP 평가

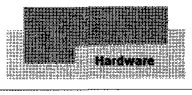
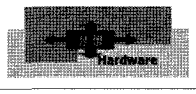
  임베디드 소프트웨어가 운영체제 기능 이용을 올바르게 구현하고 있는가를 평가한다.

- IC 평가

  임베디드 시스템은 하드웨어와 여러 소프트웨어의 조합으로 이루어져 있으므로 이러한 구성 요소가 결합되어 함께 실행되는 관점을 평가한다.

각 평가 영역의 대상 범위는 표 3과 같다. HP 평가와 OP 평가의 경우 표 2에서 언급한 바와 같이 구성요소의 일부 부분을 의미하며, BC 평가의 경우 임베디드 소프트웨어의 개별적인 구성 요소와 관계없이 전체를 대상으로 하며, IC 평가의 경우 임베디드 소프트웨어와 하드웨어가 결합하여 실행하는 관점이 평가의 대상이다. 각 평가의 수행을 위한 세부적인 사항은 4절에서 기술하도록 한다.

표 3 평가 영역

대상	HP 평가	OP 평가
범위		
대상	BC 평가	IC 평가
범위		

4. 평가 모델

평가 모델은 그림 3에서처럼 총 9개의 평가 항목으로 구성된 BC 평가, 총 24개의 평가 항목으로 구성된 HP 평가, 총 100개의 평가항목으로 구성된 OP 평가, 총 5개의 항목으로 구성된 IC 평가를 위한 평가 영역이 존재한다. 본 논문에서 제안하는 평가 모델은 그림 3에서와 같이, BC 평가, HP 평가, OP 평가, IC 평가와 같은 평가영역과 각 평가영역별로 존재하는 평가부분 및 평가 항목으로 구성한다. 평가항목은 평가를 수행할 때 고려해야 할 구체적인 항목들을 의미하며, 평가부분은 평가항목의 의미 있는 집합이다.

4.1 평가 영역

4.1.1 BC 평가

BC 평가는 임베디드 소프트웨어가 기본적으로 만족시켜야 할 조건들을 올바르게 구현하고 있는가를 평가한다[21]. 따라서 BC 평가 부분은 그림 4의 (가)와 같이 임베디드 소프트웨어 전체이다. 또한 이러한 조건들

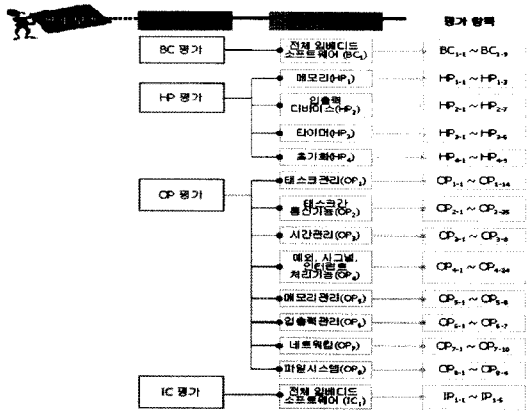


그림 3 평가 모델의 구성

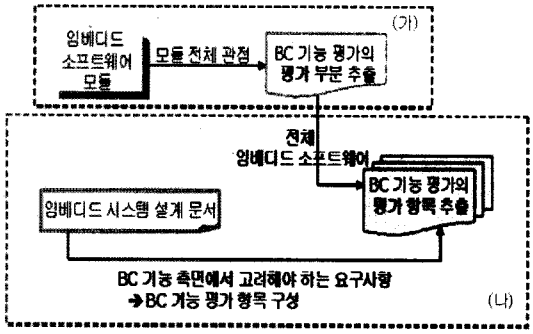


그림 4 BC 평가 부분 및 평가 항목 추출의 근거

은 임베디드 소프트웨어를 설계할 때에 지침으로써 제시되므로 BC 평가 항목은 그림 4의 (나)에서 볼 수 있듯이 임베디드 시스템 설계와 관련된 문서들을 토대로 추출한다.

임베디드 소프트웨어가 기본적으로 만족시켜야 할 조건은 다음과 같다.

- 소프트웨어로써 가져야 하는 조건인 올바른 구성(BCR1),
- 실행 오버헤드 감소(BCR2)
- 최소한의 메모리 사용(BCR3)
- 저전력 이용(BCR4)

BCR1과 BCR2의 경우 소프트웨어로써 가져야 할 조건이며, 임베디드 소프트웨어가 메모리 및 전력 공급의 제약을 가지는 것을 고려하여 BCR3과 BCR4를 추가하였다.

BC 평가 항목은 이 들 조건들을 기준으로 구성하였으며, 표 4는 BC 평가항목의 일부이다.

4.1.2 HP 평가

HP 평가는 하드웨어를 접근하는 부분의 평가이다. 그림 5에서 보는 바와 같이 임베디드 시스템을 구성하는 하드웨어는 크게 마이크로프로세서, 메모리, 입출력 디

표 4 BC 평가 항목

		평가 항목	평가 항목 그룹
전체 임베디드 소프트웨어 (BC_1)	BC_{1-1}	특정 연산을 수행하기 위해 반드시 선행되어야 하는 연산이 존재함을 조건으로 하는 경우, 이러한 순서가 엄격히 지켜졌는지에 대해 확인해야 한다.	BCR1
	BC_{1-2}	두 개 이상의 태스크가 공유할 수 있는 함수는 재진입 가능한 함수로써 구현해야 한다.	
	BC_{1-3}	기능적으로 연관돼 있거나 긴밀하게 연관된 모듈은 하나의 작업 단위 모듈로 묶어서 구현되어야 한다.	BCR2
	BC_{1-4}	기능적으로 연관이 없더라도, 언제나 동시에 수행되는 부분들은 하나의 작업 단위 모듈로 묶어서 구현되어야 한다.	
	:	:	:

바이스, 타이머가 있다[18]. 메모리는 명령어와 수행에 필요한 데이타를 저장하며, 입출력 디바이스는 사용자와의 입출력을 담당하고, 타이머(외장 또는 마이크로프로세서에 내장된 타이머)는 시스템 구동의 시간 관리를 수행한다. 마이크로프로세서는 나머지 세 개의 하드웨어와 통신하며 주어진 명령어를 수행한다[18].

한편 하드웨어 접근과 관련된 부분으로는 대상 어플리케이션이 하드웨어와 함께 작동하기 위해 기본적으로 필요한 초기화 부분이 있다. 따라서 본 모델은 그림 5의 (가)에 나타내었듯이, 타이머 부분, 입출력 디바이스 부분, 메모리 부분, 하드웨어 초기화 부분을 HP 평가 부분으로 구성하며, 각 평가 부분에서 평가하고자 하는 관점은 다음과 같다.

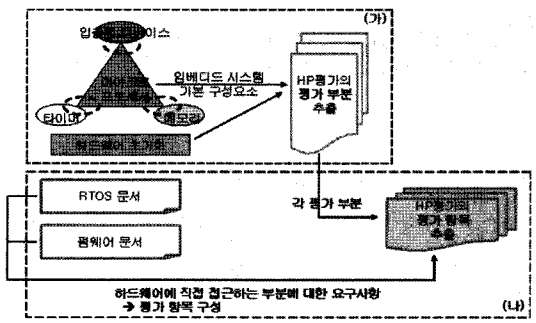


그림 5 HP의 평가 부분 추출 및 평가 항목 추출의 근거

1) 메모리

메모리 접근 부분에 대한 평가는 응용 소프트웨어를 처음 비휘발성 메모리에 로드하기 위한 설정 부분 또는 시스템이 수행되는 동안 마이크로프로세서의 레지스터

및 램(RAM)에 직접적으로 접근하는 부분에 대한 평가를 의미한다. 즉, 데이타의 공간 할당(HPR1), 메모리 접근(HPR2)이 해당한다.

2) 입출력 디바이스

디바이스와 직접 통신 및 입출력 요청을 수행하기 위한 관점들에 대한 평가를 의미한다. 본 평가에서는 특정 디바이스에 의존적이지 않은 상위레벨에서, 디바이스와 관련된 구분 형태를 분류하고 이에 따라 요구되는 평가를 수행한다. 즉, 입출력 연산 트리거(HPR3), 입출력 연산을 위한 메모리 할당(HPR4), 입출력 수행(HPR5)이 해당한다.

3) 타이머

경과 시간과 관련된 작업을 수행하거나, 일정 주기나 입력 펄스 회수에 따라 특정 이벤트를 처리해야 경우를 위해 타이머가 하드웨어로써 존재할 때에, 응용 소프트웨어 내에서 이 타이머의 기본 동작을 제어하는 부분에 대한 평가를 의미한다. 즉, 타이머 파라미터 값 세팅(HPR6), 타이머의 신호 받기(HPR7)가 해당한다.

4) 초기화

하드웨어와 결합하여 기능을 수행하는 임베디드 소프트웨어의 동작을 위해 필요한 기본적인 초기화 부분에 대한 평가를 의미한다. 즉, 하드웨어에 대한 기본적인 초기화(HPR8)가 해당한다.

HP 평가를 위해 추출한 각 평가 부분에 대하여 필요한 요구사항을 실시간 운영체제 및 펌웨어 기능을 토대로 하여 추출한 것이 HP 기능 평가 항목이며(그림 5의 (나)), 표 5에 그 일부를 나타낸다.

표 5 HP 평가 항목

HP 평가 부분	평가 항목		평가 항목의 토대
메모리 (HP_1)	HP_{1-1}	특정 작업을 위한 메모리 할당 시, 메모리 맵에 의거하여 유효한 번지 값을 할당해야 한다.	HPR1
	HP_{1-2}	특정 작업을 위해 레지스터 연산을 필요로 하는 경우, 의도한 레지스터에 올바로 접근해야 하며, 해당 레지스터에 유효한 값으로 세팅해야 한다.	HPR2
	:	:	:
입출력 디바이스 (HP_2)	HP_{2-1}	능동형 I/O 디바이스를 사용하는 경우, 인터럽트 발생을 처리하는 모듈을 포함해야 한다.	HPR3
	HP_{2-2}	수동형 I/O 디바이스를 사용하는 경우, 폴링 요청을 받아들일 수 있는 모듈을 포함해야 한다	
	HP_{2-3}	해당 주변 디바이스가 점유한 주소 공간에 대해 올바르게 구현해야 한다.	HPR4
	:	:	:
:	:	:	:

4.1.3 OP 평가

OP 평가 부분은 임베디드 시스템에 탑재하는 운영체제가 어떠한 기능을 제공하는가에 대한 평가이다. OP 평가 부분을 도출하기 위하여 그림 6의 (가)에서처럼 일반적인 운영체제, 실시간 운영체제, 임베디드 리눅스가 제공하는 기본 기능에 대하여 조사를 수행하였으며, 그 결과 OP 기능 평가 부분으로는 태스크 관리, 태스크 간 통신 관리, 시간 관리, 인터럽트·시그널·예외처리, 메모리 관리, 입출력 관리, 네트워킹, 파일 시스템으로 요약된다. 각 평가 부분에서 평가하고자 하는 관점은 다음과 같다.

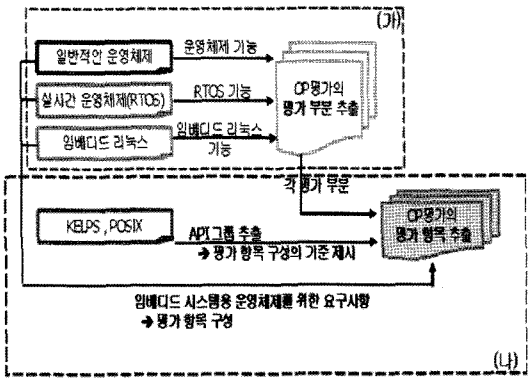


그림 6 OP 평가 부분 및 평가 항목의 근거

1) 태스크 관리

태스크 생성, 구현 및 삭제, 우선순위 부여 및 이에 따른 스케줄링과 같은 운영체제 서비스를 위한 평가 항목을 의미하며, 임베디드 시스템의 복잡한 기능을 효율적으로 처리하면서 응답성을 높이기 위해 태스크 단위로 어플리케이션을 구현하는 프로그래밍 방식이 일반화되었기 때문에 이러한 평가가 이루어져야 한다.

2) 태스크 간 통신 기능

다양한 태스크간의 자원 및 동작 동기화를 위한 각종 태스크 간 통신 메커니즘을 제공하는 운영체제 서비스를 위한 평가 항목을 의미하며, OP에서는 태스크간의 자원 및 동작 동기화를 위한 각종 태스크 간 통신 서비스 메커니즘을 사용하기 때문에, 이에 대한 평가가 이루어져야 한다.

3) 시간 관리

시간과 관련된 작업이나, 일정 주기 또는 입력 펄스 회수에 따라 특정 이벤트를 처리해야 하는 경우를 지원하는 운영체제의 시간 관련 서비스에 대한 평가를 의미한다. 임베디드 어플리케이션은 하드 타이머를 다루거나

소프트 타이머를 구현하고 다루기 위한 운영체제 시간 관리 서비스를 이용하기 때문에 이에 대한 평가가 이루어져야 한다. 이는 HP 평가에서 정의한 바와 같이, 타이머의 기본적인 동작을 제어하는데 있어서 운영체제의 사용 여부와 관계없이 개발자가 관여해야 하는 부분은 위에서 말한 타이머 관리 서비스와 구분하도록 한다.

4) 인터럽트·시그널·예외 처리 기능

예외, 시그널 및 인터럽트 처리와 관련된 운영체제의 지원 서비스에 대한 평가를 의미하며, 예외, 시그널, 인터럽트와 같은 처리 루틴을 지원 할 수 있는 서비스에 대한 평가가 이루어져야 한다.

- 인터럽트 처리 기능

  다양한 하드웨어 장치들은 특정한 이벤트가 발생했을 때, 이를 처리하기 위해 마이크로프로세서에 인터럽트를 보내게 되고, 이를 감지한 마이크로프로세서는 인터럽트 처리 루틴을 실행시키게 된다. 이런 인터럽트 처리 과정을 지원하기 위해 임베디드 어플리케이션은 운영체제의 인터럽트 처리 서비스를 사용한다.

- 예외 및 시그널 처리 기능

  소프트웨어적 인터럽트 시그널을 위한 시그널 처리루틴 및 태스크 내부적인 명령어에 의해 발생하는 예외를 위해 예외 처리 루틴을 위해 임베디드 어플리케이션은 운영체제 서비스를 사용한다.

5) 메모리 관리

태스크가 동적 또는 정적 메모리 할당 정책을 선택하고, 이에 따라 메모리를 할당 및 반환할 수 있도록 지원하는 운영체제의 메모리 관리 서비스에 대한 평가를 의미한다. 임베디드 어플리케이션은 작업을 수행하기 위해 동적 또는 정적 메모리 할당 정책을 선택하고, 이에 따라 메모리를 할당 받고, 반환하는 등의 메모리 관리와 관련된 일련의 운영체제 서비스를 이용할 수 있으며, 이러한 서비스에 대한 평가가 이루어져야 한다.

6) 입출력 관리

각 디바이스 드라이버에 대한 API를 추상화하여 일관된 인터페이스를 제공하는 운영체제의 I/O 서브시스템 서비스에 대한 평가 및 입출력 요청을 처리해주는 운영체제의 입출력 작업 서비스에 대한 평가를 의미한다. 이는 HP 평가에서 정의한 바와 같이, 운영체제의 이용과 관계없이 입출력을 위해 기본적으로 요구되는 요소에 대한 평가와는 구분하도록 한다.

7) 네트워킹

많은 임베디드 시스템이 인터 네트워킹을 기본적인 요구사항으로 가지게 됨에 따라 그 사용이 증가된, 인터 네트워킹 및 분산 컴퓨팅을 지원하기 위한 운영체제의 TCP/IP 프로토콜 스택 및 원격 프로시저 호출(RPC) 관련 서비스에 대한 평가를 의미한다.

8) 파일 시스템

파일과 디렉토리의 생성, 삭제 및 이용 그리고 시스템 내부의 별도 저장장치나 네트워크 상의 저장장치에의 저장 및 검색을 수행하는 운영체제의 파일 시스템 서비스에 대한 평가를 의미한다. 최근 임베디드 시스템의 소형 PC화로 인해 파일 시스템을 사용하는 임베디드 시스템이 증가하였기 때문에 임베디드 어플리케이션에서의 파일 시스템 운영체제 서비스를 이용에 대한 평가가 이루어져야 한다.

OP 평가 항목은 그림 6의 (나)에서 볼 수 있듯이, 위에서 기술한 8개의 OP 평가 부분에 대해, ELCPS[4], KELPS[5]. POSIX[9] 표준의 API 그룹 함수들로부터 각 평가 부분을 수행하기 위해 요구되는 사항들을 평가 항목으로 추출한다. 즉, OP 평가 항목은 운영체제의 주요 기능을 토대로 추출되었으며, 항목을 추출한 그룹 함수들의 목록은 표 6과 같다.

표 7에서는 OP평가 항목의 일부를 기술한다. OP 평가의 경우, 운영체제 기능의 특성상 소프트웨어의 기능에 기여하는 중요도 또는 임베디드 소프트웨어의 특성을 지원하기 위한 중요도가 존재하기 때문에, 각 평가 항목에 상·중·하와 같은 비중을 두어, 실제 평가 시 비중 있게 평가되어야 할 부분들을 선별할 수 있도록 한다.

4.1.4 IC 평가

그림 7의 (가)에서 보는 바와 같이, 임베디드 제품은 하드웨어와 소프트웨어로 구성되어 있으므로 이러한 구성 요소가 결합되어 함께 실행되는 관점에서 평가할 필요가 있다[19]. IC 평가의 평가 부분은 하드웨어와 소프트웨어의 동적인 실행 평가 부분이다.

소프트웨어를 하드웨어에 탑재하여 동적으로 실행되는 관점에서 고려해야 평가 항목의 기준으로는 하드웨어와 소프트웨어 사이의 시그널 전환(ICR1), 실행 시 시간 관련 연산(ICR2) 및 동적 연산에 관한 요구사항(ICR3)이 있으며, 표 8에 IC 평가항목의 일부가 있다.

4.2 평가 모델의 적용

평가 모델을 임베디드 소프트웨어 평가에 적용하는 절차는 그림 8과 같이 제품의 '구성 요소 파악', '평가선택', '평가수행'의 단계로 이루어지며, 각 단계에 대한 세부 설명은 다음과 같다.

1) 단계 1

평가 대상 임베디드 소프트웨어의 구성 요소를 파악한다. 구성 요소에 대한 정의는 3.1절에서 기술한 것과 같다.

2) 단계 2

평가 대상 임베디드 소프트웨어에 적용할 평가를 파악한다.

표 6 OP 평가 항목의 구성

OP 평가 부분	OP 평가 항목을 추출한 표준의 API 함수 그룹 [참조한 표준 번호]	번호
태스크 관리 (OP_1)	태스크관리: _JOB_CONTROL[4,5,9], _SINGLE_PROCESS[4,5,9], _SPAWN[5,9]	OPR1
	스케줄링: _SCHEDULING[5], _PRIORITY_SCHEDULING[9]	OPR2
	멀티프로세스: _MULTI_PROCESS[4,5,9], _MULTI_ADDR_SPACE[4,5,9]	OPR3
	쓰레드: _THREADS[4,5,9], _THREADS_EXT[4,5,9], _THREAD_PROCESS_SHARED[9], _THREADS_REALTIME[5,9], _THREAD_PRIORITY_SCHEDULING[9], _THEADS_REALTIME_EXT[5], _SPIN_LOCKS[9], _BARRIORES[9]	OPR4
태스크 간 통신 (OP_2)	태스크간 통신: _IPC[4,5,9]	OPR5
	세마포어: _SEMAPHORES[5,9]	OPR6
	파이프: _PIPE[4,5,9]	OPR7
시간 관리 (OP_3)	시간관리: _TIMERS[5,9], _CLOCK_SELECTION[5]	OPR8
인터럽트 · 시그널 · 예외 처리 (OP_4)	인터럽트 · 시그널 · 예외 처리: _SIGNALSI5,9], _REALTIME_SIGNALS[9], _SIGNAL_JUMP[4,5,9]	OPR9
메모리 관리 (OP_5)	메모리관리: _DYNAMIC_LINKING[4,5], _MEM_MGMT[4,5,9], _MAPPED_FILES[9], _MEMORY_PROTECTION[9], _MEM_LOCK[9], _MEM_LOCK_RANGE[9]	OPR10
	주소관리: _MULTI_ADDR_SPACE[4,5,9]	OPR11
	레지스터관리: _REG_EXP[4,5,9]	OPR12
입출력 관리 (OP_6)	입출력관리: _ASYNCHRONOUS_IO[5,9]	OPR13
	Device IO:_DEVICE_IO[4,5,9], _DEMULTI_ADDR_SPACE[4,5,9], _WIDE_CHAR_DEVICE_IO[4,5,9]	OPR14
	장치: _DEVICE_SPECIFIC[4,5,9], _DEVICE_SPECIFIC_R[4,5,9]	OPR15
네트워킹 (OP_7)	네트워킹: _NETWORKING[4,5,9]	OPR16
	Remote Procedure Call: _NETWORKING_RPC[4,5,9]	OPR17
파일 시스템 (OP_8)	파일시스템: _FD_MGMT[5,9], _SYNCHRONOZIED_IO[9], _FYNC[9], _FIFO[4,5,9], _FILE_ATTRIBUTES[4,5,9], _FILE_SYSTEM[4,5,9], _FILE_SYSTEM_EXTEN[4,5,9], _FILE_SYSTEM_R[4,5,9], _LARGE_FILE[4,5,9], _STDIO_LOCKING[4,5,9]	OPR18

표 7 OP 평가 항목

OP 평가 부분	평가 항목		우선순위			평가항목 의 도메
			상	중	하	
태스크 관리 (OP_1)	OP_{1-1}	시스템이 요구된 기능을 올바르게 수행하기 위해, 각 태스크는 의도된 우선순위를 부여받아야 한다.	√			OPR1 OPR2
	OP_{1-2}	커널의 동작을 보장하기 위해, 응용 프로그램 태스크는 시스템 태스크의 우선순위 레벨을 가져서는 안 된다.		√		
	OP_{1-3}	커널의 동작을 보장하기 위해, 응용 프로그램 태스크가 시스템 태스크의 우선순위를 변경해서는 안 된다.			√	
	OP_{1-4}	공유 메모리 접근 시, 읽기 기능을 하는 태스크가 쓰기 기능을 하는 태스크보다 높은 우선순위를 가져야 한다.			√	
	OP_{1-5}	우선순위 역전을 피하기 위해, 우선순위 계승 프로토콜, 상한 우선순위 프로토콜, 우선순위 상한 프로토콜 등의 자원 접근 제어 프로토콜을 사용해야 한다.		√		
	:	:				:
태스크 간 통신 기능 (세마포 연산) (OP_2)	OP_{2-8}	하나의 태스크에서 크리티컬 섹션을 설정하는 경우, 진입과 해제가 항상 쌍으로 존재하도록 구현해야 한다.	√			OPR5 OPR6
	OP_{2-9}	바이너리 세마포, 카운팅 세마포, 뮤텍스 가운데 지원 기능 차이 및 사용 효율을 고려하여 적절한 방식을 선택해야 한다.		√		
	OP_{2-10}	공유자원에 대한 배타적 접근과 I/O장치에 대한 올바른 동작을 보장하기 위해, 세마포를 획득하지 않은 태스크도 세마포를 반환할 수 있게 되는 바이너리 세마포와 카운팅 세마포의 경우, 세마포의 올바른 반환이 일어나야 한다.	√			
	OP_{2-11}	세마포가 공유 자원이나 크리티컬 섹션을 보호하고 있는 경우, 데이타의 일관성을 유지할 수 있도록 사용 중인 세마포는 삭제하지 않아야 한다.			√	
	:	:				:
:	:	:				:

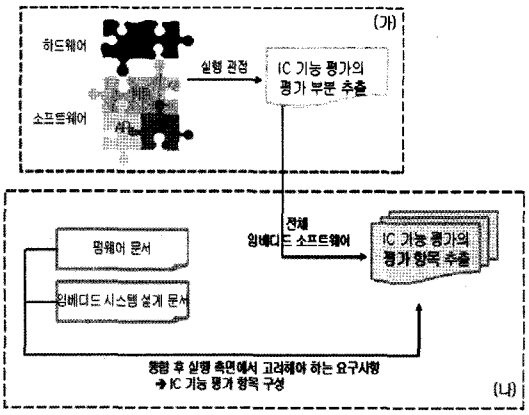


그림 7 IC의 평가 부분 추출 및 평가 항목 추출의 근거

표 8 IC 평가 항목

IC 평가 부분	평가 항목		평가 항목 명칭
전체 임베디드 소프트웨어 (IC_1)	IC_{1-1}	두 구성요소가 올바르게 구현되었다고 하여도, 실행시의 시그널 변환 관점에서 오류가 발생할 수 있기 때문에 이를 평가한다.	ICR1
	IC_{1-2}	실행에 있어서 시간 지연으로 인해 연산이 영향을 받는 부분이 있는가를 평가한다.	ICR2
	:	:	:

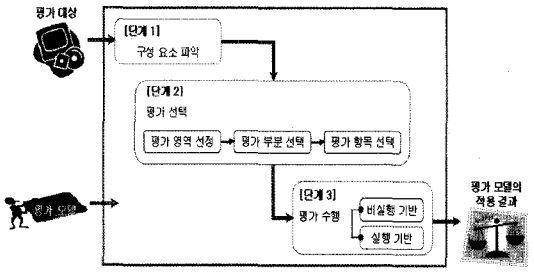


그림 8 평가 모델의 적용 절차

- 평가 영역 선정 : 대상 임베디드 소프트웨어의 구성요소에 따라 BC 평가, HP 평가, OP 평가, IC 평가로 이루어지는 평가 영역을 선정한다. 표 9는 임베디드 제품의 논리적 구성 요소 조합에 따라 가능한 기능 평가 종류에 대해 기술한다.
- 평가 부분 및 평가 항목 선택 : 평가 영역에 대하여 대상 임베디드 소프트웨어의 평가 부분 및 평가 항목을 선택한다.

3) 단계 3

평가 항목을 적용함으로써 평가를 수행한다. 이 때,

표 9 구성 조합 별 기능 평가

구성 조합	가능한 기능 평가
AP + HP	BC 평가, HP 평가, IC 평가
AP + OP + HP	BC 평가, HP 평가, OP 평가, IC 평가

평가를 수행하는 방법은 비실행 기반 평가와 실행 기반 평가의 두 가지가 있다. 비실행 기반 평가란, 개발된 임베디드 소프트웨어의 구현 코드를 검토(Review)하거나, 개발에 관련된 문서를 검토하는 것과 같이 별도의 실행 없이 평가를 수행[22]하여 평가 값을 얻는 평가 방법을 의미하며, 실행 기반 평가란, 개발된 임베디드 소프트웨어를 하드웨어에 탑재하여 직접 실행하면서 평가를 수행하여 평가 값을 얻는 평가 방법을 의미한다.

본 모델의 평가 영역 가운데 BC 평가의 경우 비실행 기반 평가를 위해서만 사용될 수 있는 평가 항목을 갖추고 있으며, IC 평가의 경우는 탑재되어 실행되는 실행 기반의 평가를 위해 사용될 수 있는 평가 항목들을 갖추고 있다. 따라서 BC 평가와 IC 평가는 각각 한 종류의 평가 방법만을 이용한다. 이에 반하여 HP 평가와 OP 평가의 경우는 실행 및 비실행 기반의 평가 방법 모두를 위해 사용될 수 있다. 이는 이 두 가지 평가에서 비실행 기반의 평가 방법으로 수행하여 평가 항목이 만족되어도, 실행하면서 예기치 않은 결과를 야기하는 경우가 있기 때문이다. 이러한 오류들은 HP와 OP 부분에 존재하면서, 실행되는 동안만 발견이 되기 때문에 실행 기반의 평가 방법까지 수행함으로써 대상 소프트웨어의 기능을 더욱 견고히 평가할 수 있게 된다. 그러므로 HP 평가와 OP 평가의 경우 평가자는 어떤 방법으로 평가를 수행할 것인지에 대해서 선택하여 평가를 수행하게 되며, 표 10은 이를 정리하여 제시한다.

표 10 평가 수행 방법

평가 영역	비실행 기반 평가	실행 기반 평가
BC 평가	√	
HP 평가	√	√
OP 평가	√	√
IC 평가		√

5. 사례 연구 및 평가모델 분석

평가 모델의 사례 연구 대상은 RTOS를 기반으로 LCD (출력 장치) 및 시리얼 통신 디바이스를 이용한 임베디드 소프트웨어로써, 어플리케이션 코드에 디바이스를 제어하는 코드까지 혼합된 형태를 갖고 있으며, 각 디바이스에 관련된 기능을 수행하기 위해 독립적인 태스크를 생성하는 임베디드 소프트웨어이다.

5.1 평가 절차

4.2절에서 제시한 평가 절차에 따라 적용한 내용은 다음과 같다.

[단계 1] 구성 요소 파악

대상 소프트웨어는 "AP + HP + OP"의 형태로 구성되며, 이러한 분석 예시의 일부는 그림 9와 같다.

```
  :
TimeTickInit() ;                                  → HP, 초기화와 관련된 HP 부분
CommInit() ;
  :
ret += OSTaskCreate((void *)commTask, ,.....) ;   → OP, 태스크 관리와 관련된 OP 부분
  :
sData = CommGetChar(0, &err) ;
If(sData == KEY_ESC) asm rst;                     → AP, 통신채널로부터 받은 데이터를 처리하는 로직을
sprintf(dispbuf, "%05u", cnt++) ;                   구현한 AP부분
DispStr(1,3,dispbuf);
  :
OSSemPend(DispSem, 0, &err) ;                     → OP, 태스크간 통신과 관련된 OP 부분
DispSel (DISP_SEL_CMD_REG) ;
DispDataWr ( 0x40 + ( id << 3 ) ) ;               → HP, 메모리와 관련된 OP 부분
DispSel (DISP_SEL_DATA_REG) ;
  :
```

그림 9 임베디드 소프트웨어의 구성 요소 분석 예

[단계 2] 평가 선택

① 평가 영역 선정 : 표 8에 따라 본 소프트웨어는 HP 평가, OP 평가, BC 평가, IC 평가를 모두 수행할 수 있으며 본 사례에서는 모든 기능 평가 영역을 선택한다.

② 평가 부분 선택 : 대상 소프트웨어를 분석하여 선택한 평가 부분은 표 10과 같다.

③ 평가 항목 선택 : 선택한 각 평가 부분에 대하여, 관심이 되는 평가 항목을 선택하였으며, 선택한 평가 항목은 평가 수행 결과와 함께 표 11에서 기술한다.

[단계 3] 평가 수행

[단계 2]에서 선택한 평가 항목에 대해서 평가를 실행한다. 본 사례에서는 표 11에서 볼 수 있듯이 BC 평가, HP 평가의 경우는 비실행 기반 평가만을 수행하고, OP 평가의 경우는 비실행 기반과 실행 기반 평가 방법 모두를 적용함으로써 더 견고하게 임베디드 소프트웨어 평가를 수행하였으며, IC 평가는 실행 기반의 평가를 수행하였다.

표 11 평가 부분 및 평가 수행 방법 선정

기능평가 영역	평가부분 선택	평가 수행 방법
BC평가	전체	비실행 기반
HP평가	메모리, 입출력 디바이스, 타이머, 초기화	비실행 기반
OP평가	태스크 관리, 태스크간 통신 기능, 예외 • 시그널 • 인터럽트 처리 기능	비실행 기반, 실행 기반
IC 평가	전체	실행 기반

5.2 평가 결과

각 기능 평가에 대한 수행 결과의 일부는 표 12와 같다. 표 12는 평가 항목에 대한 평가 결과 값과 이러한 결과 값을 가지게 된 근거를 나타낸다. 각 평가 항목을 만족한 경우 1점, 만족하지 않은 경우 0점이다. 단 OP 평가의 경우는 평가 항목을 만족한 경우, 항목에 대한 비중을 고려하여 상·중·하에 따라 결과 점수가 3,2,1점이 된다. 표 12의 총 결과에서는 각 평가영역의 평가 부분별 전체에 대하여 요구되는 결과 값의 총 합에 대한 측정된 결과 값의 비율로써 표시하며, 이를 통해 대상 임베디드 소프트웨어에서 개선이 필요한 부분을 식별할 수 있다.

표 12에 나타내었듯이 BC평가에서 2개, OP 비실행평가에서 4개, OP 실행평가에서 5개(단 4개는 OP비실행 평가 결과와 동일), IC평가에서 1개 총 8가지의 오류를 발견하였다. BC 평가에서의 오류는 대상 임베디드 소프트웨어가 올바른 구성 및 시스템 수행의 오버헤드 측면을 위해 개선해야 할 항목들을 의미하며, OP 비실행 평가에서의 오류는 응용 프로그램의 올바른 동작을 저해하는 태스크 관리 및 태스크간 통신 측면에서의 오류를 의미한다.

한편, OP 비실행 기반 평가 및 실행 기반 평가 결과의 비교로부터, 비실행 기반 평가를 수행하였을 때 발견할 수 없던 태스크 수와 관련된 오류가 실행 기반의 평가를 통해 발견될 수 있었음을 알 수 있었다. IC 평가에서의 오류는 동적 실행 시, 시간과 관련한 설정으로 인해 올바르지 못한 실행을 야기한 오류를 의미한다.

5.3 평가모델 분석

본 평가 모델은 임베디드 소프트웨어의 특성을 반영하여, 임베디드 소프트웨어의 기능을 평가하기 위해 개발하였다. 임베디드 소프트웨어는 탑재할 대상 플랫폼 및 정의된 기능 요구사항에 따라 매우 다양한 형태로 존재하기 때문에, 주로 특정 제품에 한정된 기술에 대한 연구가 대부분이다[20]. 또한 임베디드 소프트웨어에 대한 시스템 테스트를 주로 수행하는 경우, 오류가 발생하여도 오류가 발생한 모듈을 찾는 것이 어렵다는 문제점이 발생한다. 이는 임베디드 소프트웨어의 컴포넌트가 매우 밀접한 구성으로 이루어져 있고 또한 직접 실행될 때에 문제가 발생하기도 하기 때문이다. 따라서 제안하는 평가 모델은 임베디드 소프트웨어의 기존 기능 평가 방법들에 대한 단점들을 보완할 수 있다.

첫째, 도메인에 특화된 관점이 아닌 일반적인 관점으로 임베디드 소프트웨어의 구성 요소를 정의하고 이를 기반으로 하여 평가를 제시하므로 다양한 형태의 임베디드 소프트웨어를 모두 대상으로 할 수 있다. 또한 평가 영역 별 세부적인 선택 기준인 평가 부분을 제시하

표 12 평가를 위한 평가 부분 및 평가 항목 선정

BC 평가	비실행	전체 (BC_1)	BC_{1-1}	1	LCD 디바이스를 다루기 위해 제시된 DispHorBar()와 DispHorBarInit()의 우선순위가 바르게 지켜짐		0.5 (2/4)
			BC_{1-2}	0	DispStr()이 시작 태스크와 바 태스크에 의해 공유되지만, 고려되지 않았음	오류 검출	
			:	:	:		
			BC_{1-4}	0	기능별로만 모듈화가 이루어짐	오류 검출	
			:	:	:		
HP 평가	비실행	메모리 (HP_1)	HP_{1-2}	1	DispSel()함수와 이를 이용하는 부분에서 올바로 구현되어 있음		1 (2/2)
			HP_{1-3}	1	DispDefChar()에서 의도한 메모리 주소 값 설정		
		입출력 디바이스 (HP_2)	HP_{2-2}	1	외부 통신포트로부터 값을 가져오는 기능을 구현한 모듈이 있음		1 (3/3)
			HP_{2-4}	1	정의된 프로세서 명령어로 포트가 접근됨		
			:	:	:		
		:	:	:	:		:
		전 체				1 (8/8)	
OP 평가	비실행	태스크 관리 (OP_1)	OP_{1-1}	3	각 태스크를 의도한 우선순위로 구현함		0.778 (7/9)
			OP_{1-3}	1	해당 모듈 없음		
			OP_{1-9}	0	receive task를 대기상태로 만드는 OSMboxPend()를 사용하였지만 이를 구동시킬 수 있는 send task의 OSMboxPost()가 올바르게 구현되지 않음	오류 검출	
			:	:	:		
			OP_{1-12}	0	블로킹 호출을 구현하지 않았음	오류 검출	
			OP_{1-14}	1	적합한 태스크 수를 가짐		
			:	:	:		
		태스크 간 통신 기능 (OP_2)	OP_{2-1}	3	세마포를 사용하여 자원을 공유함		0.839 (26/31)
			OP_{2-2}	1	통신의 동기화를 위해 메일박스를 생성함		
			OP_{2-3}	2	세마포와 메일박스, 두 개의 오브젝트를 구현하였으나 자원 공유와 통신이라는 다른 목적을 가지므로 효율적인 구현이라고 할 수 있음		
			OP_{2-4}	1	의도한 오브젝트에 대한 연산을 수행함		
			OP_{2-5}	3	DispInit(), commInit()에서 구현		
			OP_{2-7}	2	LCD라는 자원과 통신 채널의 버퍼를 공유하기 위해 세마포를 사용		
			:	:	:		
			OP_{2-13}	0	데드락 문제 해결에 관한 구현부분 없음	오류 검출	
			:	:	:		
			OP_{2-22}	0	리턴 되는 에러상수값에 대한 처리부분 없음	오류 검출	
			:	:	:		
		:	:	:	:		:
		전 체				0.848 (39/46)	
	실행	태스크 관리 (OP_1)	:	:	:		0.667 (6/9)
			OP_{1-14}	0	비실행 기반에서는 문제가 되지 않는다고 판단되었으나, 탑재하여 실행시켰을 경우, 스택 오버 플로우로 인해 시스템이 문제를 일으킴	오류 검출	
			:	:	:		
		:	:	:	:		:
		전 체				0.826 (38/46)	
IC 평가	실행	전체 (IC_1)	IC_{1-1}	1	실행 시 문제없음		0.8 (4/5)
			IC_{1-2}	0	실행 시, 적합하지 않은 시간 딜레이로 인해 의도한 기능이 나타나지 않음	오류 검출	
			IC_{1-3}	1	실행 시 문제없음		
			:	:	:		

여, 대상 임베디드 소프트웨어에 따라 적합하게 평가를 선택하여 수행할 수 있는 체계를 구성함으로써 존재하는 대부분의 임베디드 소프트웨어를 위해 효용을 가질 수 있도록 하였다.

둘째, 임베디드 소프트웨어의 경우, 하나의 기능을 구현하는데 있어서 그 주요 기능 로직을 구현한 부분 이외에 하드웨어를 제어 하는 부분, 운영체제 서비스를 이용하는 부분들이 밀접하게 연관되어 구현된다는 특성으로 인하여, 이러한 추가적인 관점에서의 오류가 전체 기능 모듈의 오류를 발생시키기도 한다. 그러나 이러한 오류의 경우, 요구사항 명세서로부터 추출할 수 있는 정보가 아니라, 특정 기능을 구현하는데 있어서 운영체제를 사용하거나 하드웨어에 접근하기 위해 필요한 내재된 속성과 같은 정보를 의미하기 때문에 이러한 항목을 체계적으로 평가하는 데는 어려움이 있다. 이에 본 평가 모델은 OP 평가, HP 평가 영역을 제시함으로써 이러한 문제점을 보완할 수 있으며, 앞의 사례에서도 그 결과를 확인할 수 있었다. 예를 들어 표 12의 OP_{1-9} 항목인 '특정 태스크를 대기상태로 만들 때, 이 태스크를 준비상태로 만들 수 있는 인터럽트 루틴이 존재하거나, 시스템에 있는 다른 태스크가 시그널을 주도록 해야 한다'를 통하여 태스크간의 연동을 위해 receive task를 대기상태로 만드는 OSMboxPend()함수를 사용하였지만, 이를 구동시킬 수 있는 send task의 OSMboxPost()를 올바르게 구현하지 못함으로써 이를 포함한 기능 모듈에서 발생한 오류를 검출할 수 있었다.

셋째, 임베디드 소프트웨어는 하드웨어에 탑재하여 수행되는 소프트웨어이기 때문에, 임베디드 소프트웨어의 각 구성요소에서 탑재하기 이전에 문제가 없다고 판단되었지만 실행 시에 오류가 발견되기도 하며, 구성요소와 관계없이 전체적인 실행 관점에서 오류가 발견되기도 한다. 이러한 오류의 근원은 찾기가 매우 힘들다. 본 평가 모델에서는 OP 평가와 HP 평가의 실행 기반 평가 방안 및 IC 평가를 제시함으로써 이러한 문제점을 보완하였으며, 실제 사례 연구를 통하여 이를 확인할 수 있었다. 예를 들어, OP_{1-14} 항목인 '수행에 적합한 태스크 수를 가져야한다'를 통하여 RTOS의 서비스를 이용하여 태스크를 생성하고 이용할 때에 실행으로 제한이 생기는 태스크 수에 대한 오류를 검출할 수 있었으며, IC_{1-3} 항목인 '실행에 있어서 시간 지연으로 인해 연산이 영향을 받는 부분이 있는가를 평가 한다'를 통해 적합하지 않은 시간 지연 설정이 LCD 모니터에 아무런 결과도 출력하지 않았던 실행 오류를 발견할 수 있었다.

6. 결론 및 향후 연구

최근 임베디드 소프트웨어 산업 기술은 기술 개발과 시장 경쟁에 의해 빠르게 변화하고 있으며, 하드웨어의 소형화 및 고성능화가 진행됨에 따라 제품의 경쟁력이 하드웨어에서 소프트웨어 최적화 기술로 이동하고 있다. 그러나 임베디드 소프트웨어의 다양성과 확고한 표준의 부재로 임베디드 소프트웨어의 개발에 비해 임베디드 소프트웨어에 대한 평가는 그 진척이 미흡한 상황이다.

임베디드 소프트웨어는 대상 시스템에 탑재되어 그 제품의 특정 기능을 제어하기 위해 수행되는 소프트웨어이다. 이러한 임베디드 소프트웨어가 나타내고자 하는 기능이 올바르게 동작하는지 확인하는 것은 필수적이기 때문에, 임베디드 소프트웨어의 기능에 대한 평가는 매우 중요하다. 그러나 임베디드 소프트웨어는 그 특성상 탑재할 대상 플랫폼 및 정의된 기능 요구사항에 맞게 구성되고 맞춤하여 개발되므로 그 종류와 구성이 매우 다양하고, 여러 요소들이 매우 밀접하게 연결되어 있기 때문에 이를 평가한다는 것은 쉽지 않다. 본 논문에서는 임베디드 소프트웨어의 특성을 반영하여 임베디드 소프트웨어의 기능을 체계적으로 평가할 수 있는 임베디드 소프트웨어 평가 모델을 개발하였다.

본 논문에서는 다양한 계층에서 여러 형태로 존재하는 임베디드 소프트웨어를 일관 화된 관점으로 평가하기 위하여, AP, OP, HP와 같이 임베디드 소프트웨어의 구성요소를 정의하였다.

본 논문에서 임베디드 소프트웨어의 기능을 평가하기 위해 제시하고자 하는 평가는 다음과 같다. 임베디드 소프트웨어의 기능을 평가하기 위해, 임베디드 소프트웨어가 기본적으로 만족시켜야 하는 고려사항에 대한 평가인 BC 평가, 하드웨어에 의존적인 부분에 대한 평가인 HP 평가, 운영체제 서비스를 사용하는 부분에 대한 평가인 OP 평가, 그리고 마지막으로 임베디드 소프트웨어가 탑재되고 실행 될 때 고려해야 하는 평가인 IC 평가와 같은 평가 영역을 두었다. 또한 각 평가 영역 별로 평가 부분을 제시함으로써, 평가 대상 임베디드 소프트웨어에 따라 적합하게 평가항목을 적용할 수 있도록 구성하였다.

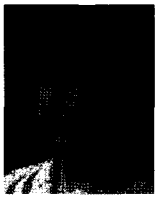
사례 연구를 수행함으로써, 평가 모델을 사용하여 임베디드 소프트웨어를 평가할 수 있는 구체적인 적용 방법을 기술하였으며 제시한 평가 모델의 타당성을 검증하였다. 즉, 제안한 평가 모델을 사용하여 임베디드 소프트웨어를 평가하였을 경우, 대상 임베디드 소프트웨어에서 발생 가능한 다양한 구현상의 오류, 시간 지연으로 인한 오류, 스택 오버플로우(overflow)와 같은 임베디드 소프트웨어에 대한 오류를 발견하였으며, 검출된 오류에 대한 보완점을 제시 할 수 있었다.

향후 제안한 평가 모델이 실질적인 임베디드 소프트웨어 테스트에 적용될 수 있도록 평가 항목별로 구체적

인 테스트 데이타 선정 기법에 대한 연구를 수행할 예정이다. 또한 본 논문에서 제안한 임베디드 소프트웨어의 기능 평가 모델 뿐 아니라, 성능, 실시간성, 신뢰도, 효율적 메모리 사용, 전력 소모, 재사용성과 같이 임베디드 소프트웨어의 비기능적 요소에 대한 평가 모델을 제안할 예정이다.

참 고 문 헌

[1] E. A. Lee, "What's Ahead for Embedded Software?," IEEE Computer, pp. 18-26, September, 2000.
[2] ISO/IEC 9126-1.2 Information Technology-Software Product Quality, 1998.
[3] EL/IX, http://sources.redhat.com/elix
[4] ELCPS, http://www.embedded-linux.org
[5] KELPS, http://www.kesic.or.kr/index.asp
[6] CELF, http://www.celinuxforum.org
[7] Emblix, http://www.emblix.org/english/etop.html
[8] ITRON, http://tron.um.u-tokyo.ac.jp/TRON/ITRON/home-e.html
[9] POSIX 1003.4, http://www.pasc.org
[10] IEEE Std 1003.1-2001, IEEE Standard for Information technology-Portable Operating System Interface(POSIX), 2001.
[11] ELC Platform Specification v1.0, 2002.
[12] KESIC Embedded Linux Platform Specification (KELPS), 2004.
[13] Jean J. Labrosse, "MicroC/OS-II, The Real-Time Kernel," CMP Books, 1999.
[14] Robert Love, "Linux Kernel Development," Developer's Library, 2003.
[15] Avi Silberschatz, Peter Galvin, Greg Gagne, "Applied Operating System Concepts," WILEY, 2001.
[16] Qung Li and Caroline Yao, "Real-Time Concepts for Embedded Systems," CMP Books, 2003.
[17] David E. Simon, "An Embedded Software Primer," Addison Wesley, 2003.
[18] Ed Sutter, "Embedded System Firmware Demystified," CMP Books, 2002.
[19] Ahyoung Sung, Byoungju Choi, "Interaction Testing in an Embedded System using Hardware Fault Injection and Program Mutation," LNCS, Springer, Vol2931, pp192-204, 2003.12.
[20] Bas Graaf, Marco Lormans, and Hans Toetenel, "Embedded Software Engineering: The State of the Practice," IEEE SOFTWARE, pp.61-69, November/December, 2003.
[21] B.Broekman, E.Notenboom, "Testing Embedded Software," Addison-Wesley, 2003.
[22] C.M.Weinberg, D.P.Freedman, "Reviews, Walkthroughs, and Inspections," IEEE Transactions on Software Engineering, 12(1): pp.68~72, 1984.



최 현 미

1999년~2003년 이화여대 컴퓨터학과 학사. 2003년~2005년 이화여대 컴퓨터학과 석사. 2005년~현재 삼성전자 기술총괄 시스템 연구소 소프트웨어 센터. 관심분야는 Software Testing Technique, Embedded Software Testing, Software Process Improvement

성 아 영

1996년~2000년 이화여대 컴퓨터학과 학사. 2000년~2002년 이화여대 컴퓨터학과 석사. 2002년~현재 이화여대 컴퓨터학과 박사과정. 관심분야는 Software Testing Technique, Embedded Software Testing, Software Reliability

최 병 주

1979년~1983년 이화여대 수학과 학사 1984년~1985년 Purdue Univ. Computer Science 학사수료. 1986년~1987년 Purdue Univ. Computer Science 석사 1987년~1990년 Purdue Univ. Computer Science 박사. 1991년~1992년 삼성종합기술원. 1992년~1995년 용인대 전산통계학과 조교수 1995년~현재 이화여대 컴퓨터학과 교수. 관심분야는 소프트웨어공학, 소프트웨어 테스트, 임베디드 소프트웨어 테스트, 소프트웨어 프로세스 개선, 소프트웨어 및 데이타 품질 측정

김 재 웅

1993년 전북대학교 전자계산학과(이학사). 1995년 전북대학교 전산통계학과(이학석사). 2001년 전북대학교 전산통계학과(이학박사). 1996년~2000년 한려대학교 정보통신공학과 교수. 2000년~2001년 ㈜케이테크 선임연구원. 2001년~현재 TTA S/W시험인증센터 선임연구원. 관심분야는 S/W품질평가, 벤치마크테스트, S/W복잡도