임베디드 시스템에서 프로세스 기반 소프트웨어를 쓰레드 기반으로 전환시 재사용성 측정

Measuring Software Reusability when converting process –based software to thread-based RTOS software on Embedded System

*경보현, 최혁승, 박정형
삼성전자 디지털 미디어 연구소
*Bo-Hyun Kyong, Hyok-Seung Choi, Jeong-Hyung Park
Digital Media R&D Center Samsung Electronics
*kyonggo@samsung.com

Abstract

정보 가전 분야에 있어서 급속한 기술 발전으로 인해 하루가 다르게 새로운 기능이 추가됨에 따라 임베디드 시스템 소프트웨어의 크기 및 복잡도 또한 함께 증가하고 있고 이를 개발하고 유지보수 하는데 있어서도 막대한 비용과 노력이 요구된다. 이를 해결하기 위한 방안으로 소프트웨어의 재사용성을 높이기 위한 노력이 이어지고 있다.본 논문에서는 프로세스 방식 기반의 OS 를 적용한 소프트웨어를 스레드 방식의 OS 기반의 소프트웨어로 전환하여 임베디드 시스템을 구현 하였으며 소프트웨어 방법론으로는 스레드 방식의 OS 기반 소프트웨어에 수정된 DARTS(Design Approach to Real- Time Systems) 방법론을 적용하여 시스템을 구현하였다. 이 구현된 시스템에서 재사용성을 측정하여 표로 제시하며 그 측정 결과를 분석하였다. 그리고 스레드 방식의 OS 기반 소프트웨어에 수정된 DARTS 방법론을 적용하여 코드만 재사용이 아니라 설계방법도 재사용이 가능함을 CE 제품에 보이고자 한다.

Key words : DARTS, OS, Reusability

1. 서 론

지난 40 년동안 SW 재사용의 경향이 처음에는 함수(function)수준의 재사용이 위주였으나 그 다음은 객체 그 이후에는 객체보다 더 큰 개념인 컴포넌트 그리고 이런 객체와 컴포넌트를 다수 포함하는 framework 단위로 재사용 trend 로 진행되어 왔다. 그리고 최근 내장형 시스템(Embedded System) 분야 있어서도 다양한 기능의 추가로 인해 소프트웨어의 크기와 복잡성이 증가하고 있으며 이에 따라 임베디드 시스템 제품의 소프트웨어에 대해서도 재사용에 대한 관심이 높아지고 있으며 실제 그 효과를 측정하고자 하는 노력도 이어지고 있다. 그리고 제품의 사양이 크지 않은 폭에서 계속 변경되고 동일한 제품 군에서 여러 가지 버전의 소프트웨어가 양산되는 특성들이 존재하며 또 제품 개발 시 이전 제품에 적용된 OS 를 또 다른 OS 로 바꾸어 제품의 성능을 향상시키고자 하는 요구도 제시되고 있다. 이와 같은 요구로 인하여 OS 전환시의 재사용성을 측정하는 기준도 필요하며 이 분야에 대해서도 학계에서 지속적으로 연구하고 있다. 그리고 많은 제품의 소프트웨어가 이미 프로세스 방식의 OS 기반으로 구현되어 있는 임베디드 시스템이 존재하며 이미 구현된 프로세스 기반 제품들을 스레드방식으로 변환하는 요구들도 나타나고 있다. 이와 같은 요구들에 따라 OS 를 전환하였을 때의 재사용성을 측정해 보고자 한다. 본 문서에서는 스레드 기반 OS 로 VxWorks 를 사용하여 CE 제품의 소프트웨어 부분에 대해서 수정된 DARTS 기법을 적용해 보고자 한다.[1]

본 논문 2 장에서는 임베디드 시스템의 전체적인 구조와 DARTS(Design Approach for Real-Time and concurrent System)[2]방법론에 대하여 간략히 설명을 하고, 3 장에서는 VxWorks 기반의 소프트웨어 구조에 DARTS 방법론을 어떻게 적용하였는가에 대하여 설명과 구현된 임베디드 소프트웨어에 대해 재사용성을 측정하기 위한 기준을 제시한다. 그리고 4 장에서는 그 측정 결과를 표로 제시 하도록 하겠다.

2. 프로세스 기반의 Software 및 DARTS

그림 1 은 렌즈로부터 아날로그 영상을 입력 받아서 Video decoder 를 통하여 디지털 신호인 Y/C 신호를 출력하는 시스템의 블록 다이어그램(Block diagram)이다[3]. 디지털 영상신호는 디스플레이 단으로 출력되며 동시에 영상 압축기의 입력신호로도 사용이 된다. 이렇게 압축된 영상신호를 외부 저장매체에 저장할 수 있으며 동시에 외부 저장매체에 저장된 디지털 이미지 파일을 읽어와 디스플레이 단으로 출력이 되며 구동이 된다.

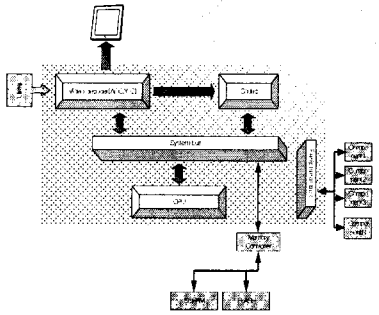

그림 1 Example of System Block Diagram

2.1. DARTS(Design Approach for Real-Time and concurrent System)

DARTS 에서는 표 1 에서 볼 수 있듯이 크게 네 단계로 나눠져 있으며 각 단계는 산출물이 있으며 순서대로 작성되어야 한다. 1)System context diagram 2)State transition diagram 3)Data flow diagram 4)Task structuring criteria 5)Task architecture diagram 6)Timing Analysis 7)Software architecture diagram 8)Structure chart 와 같은 순서로 되어 있다. [2]

1. Develop system specification	1) System context diagram 2) State transition diagram * SW structure diagram 3) Data (/control) flow diagram
2. Structure tasks	4) Task structuring criteria 5) Task architecture diagram
3. Define task interface	6) Timing Analysis
4. Design each task	7) Software Architecture Diagram 8) Structure charts

표 1 DARTS phase

임베디드 시스템 개발 시 위에 나열된 DARTS 의 모든 단계를 반드시 적용해야 되는 것이 아니라 개발 특성에 따라 생략이 가능한 단계도 존재한다.

3. 스레드 기반 Software

3 장에서는 수정된 DARTS 방법론에 대하여 각 단계별로 구현된 그림과 함께 설명을 하고 마지막으로 제품에 적용된 소프트웨어에 대한 재사용성 측정방법을 제시하도록 한다.

3.1. Context Diagram

아래 그림은 DARTS 방법론을 적용하였을 경우 소프트웨어를 기준으로 입출력 상태를 도시한 것이다.

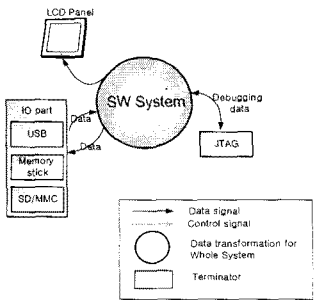

그림 2 Context Diagram

3.2. State Transition Diagram

아래 그림은 시스템의 상태 전이도를 표현하였다. 기본 메뉴 모드에서 상태에 따라 모드가 바뀌는 것를 알 수 있다. 본 제품에 대해서는 기본 메뉴의 상태변화만 표시하도록 한다.

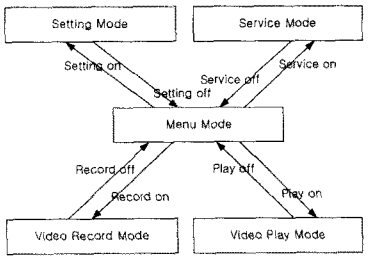

그림 3 State Transition Diagram

3.3. Primary Task Architecture Diagram/Task Architecture Diagram

본 논문의 구현 방법이 uCLinux 기반의 CE 제품를 VxWorks 기반의 CE 제품으로 전환 하였을때 재사용성을 측정하는 것이고 DARTS 에서 제안하는 Data Flow Diagram 과 Task Structuring Criteria 단계는 이미 도출된 상태이고, CE 제품 개발 특성상 DARTS 개발 단계에서 생략한다.
그림 4 는 앞서 도출된 태스크들 간의 인터페이스를 표현한 것이다. 여기서 태스크 간 인터페이스 방식을 큐를 이용하였다.

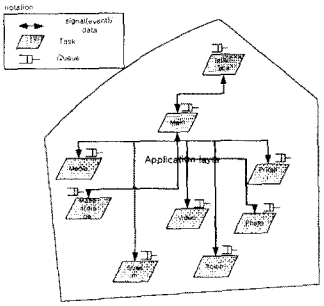

그림 4 Task Architecture Diagram

3.4. Software Architecture Diagram

다음 그림은 태스크내 데이터 저장 방식으로서 정보은닉(Information hiding) 기법이 태스크 내에 존재하는 방식을 사용하고 있다. 그리고 태스크간 통신은 메시지 커뮤니케이션 인터페이스(Message Communication Interface)로서 Loosely coupled(FIFO) 방식을 사용하고 있다.

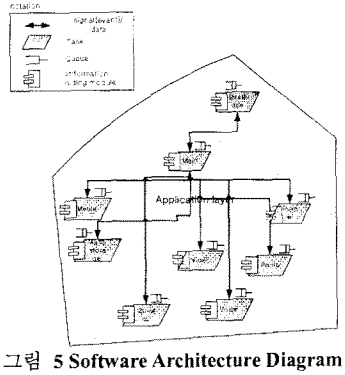

그림 5 Software Architecture Diagram

3.5. Structure Charts

Structure chart 단계는 앞 DARTS 단계에서 구해진 각 태스크 별 태스크 내부 설계를 하는 단계를 말한다. 본 논문에서는 태스크내 설계는 논문의 성격상 생략하기로 한다.

3.6. Reusability Measure

본 논문에서 구현된 uCLinux 기반 시스템에서 VxWorks 기반 시스템으로 전환할 경우 재사용성을 측정하기 위하여 두 가지 측정 기준(Basis)을 제안한다. 첫 번째로는 전체 모듈(Module)단위 재사용성을 측정하는 방법이고, 두 번째로는 각 레이어(Layer)단위의 재사용성을 측정하는 방법이다.

첫번째 방법인 전체 모듈단위 재사용성 측정이다.

$$MRM = \frac{\sum_{i=0}^{N} VMC}{\sum_{i=0}^{N} LMC} \times 100$$

MRM = Module Reusability Measure

VMC = VxWorks Module Count

LMC = Linux Module Count

uCLinux 기반의 CE 제품과 VxWorks 기반의 CE 제품은 어플리케이션, 커널, 디바이스 드라이버 크게 세 개의 레이어(Layer) 구조로 나누어진다. 본 논문에서는 재사용성 측정을 각 레이어(Layer) 단위로 구분하여 측정하고자 한다. 아래 식은 어플리케이션 레이어(Layer)의 재사용성 측정이다.

$$AMRM = \frac{\sum_{i=0}^{N} AVMC}{\sum_{i=0}^{N} ALMC} \times 100$$

AMRM = Application Module Reusability Measure

AVMC = Application VxWorks Module Count

ALMC = Application Linux Module Count

커널 레이어(Layer)의 경우 uCLinux 와 VxWorks 커널은 본 논문에서의 재사용성 측정의 생략하도록한다. 그리고 마지막으로 디바이스 드라이버 레이어(Layer)의 재사용성 측정이다.

$$DMRM = \frac{\sum_{i=0}^{N} DVMC}{\sum_{i=0}^{N} DLMC} \times 100$$

DMRM = Device driver Module Reusability Measure

DVMC = Device driver VxWorks Module Count

DLMC = Device driver Linux Module Count

4. 결과

표 1 은 3 장에서 제시한 재사용성 기준(Reusability Measure)를 이용하여 측정한 값을 표로 표현하였다. 수치가 높을수록 재사용성이 크다는 것을 의미한다.

	Substantial Reusability	intuition. Reusability	Ported function
MRM	0.775	0.959	402
AMRM	0.926	0.988	78
DMRM	0.569	0.92	324
BSP	0.00	0%	0

표 2 Reusability Measure(위 표의 수치는 모듈의 수)

5. 결론 및 향후연구

3.6 에서 언급한 식을 이용하여 재사용성을 측정하였다. 총 4 개의 항목에 대하여 측정하였는데 이중 BSP(Board Support Package)는 재사용률이 0%으로 VxWorks 기반의 제품으로 전환시 모두 새로 구현해야 한다는 의미이다. 위 표에 따라 디바이스 드라이버 레이어(Layer), 어플리케이션 레이어(Layer)순으로 재사용률 높아지는 것을 보였다. 어플리케이션 레이어(Layer)의 경우 POSIX 표준을 따라 구현하였기 때문에 재사용률의 높아지는 것을 보였다. 그러나 이 수치만으로 제품의 재사용률을 평가하는 것은 무리가 있겠으나 단순비교는 가능할 것이다. 그리고 DARTS 를 적용했다고 해서 코드의 재사용성이 높아지는 것은 아니다. DARTS 라는 설계방법을 이용하게 되므로 꼭 코드만 재사용하는 것이 아니라 설계방법, 아키텍쳐등도 재사용이 가능하게 되므로 본 CE 제품에 DARTS 방법론을 적용하게 된 것이다.

향후 좀더 재사용성이 우수한 소프트웨어 구조 및 이를 입증할 수 있는 측정 기준과 그 측정 방법에 대한 고찰이 이루어져야 할 것이다.

참고문헌

[1] VxWorks 5.5 Progrmmer's Guide, WindRiver Systems, Inc , 2002

[2] Hassan Gomaa, Software design methods for concurrent and real-time systems, Addison-Wesley, 1993

[3] Digital Video Camcorder Processor Data Sheet ver 1.05, Samsung Inc, 2004