논문 2013-08-24

저전력 임베디드 소프트웨어 개발을 위한 ADD 기반의 아키텍처 설계 기법

(A Technique of ADD-based Architecture Design for Low Power Embedded Software)

이 재 욱, 홍 장 의*
(Jae-Wuk Lee, Jang-Eui Hong)

Abstract : The embedded software has been developed in the forms of various versions that provides similar service based on product family. For increase usefulness of product family, software must has well-structured and reusable properties. Software architecture is important to improve adaptability in model-based development of embedded software mounted onto product family. In this paper, we proposed a technique of ADD(Attribute-Driven Design)-based software architecture design for low power software development. This technique provides a chance to consider the power consumption issue in design phase of software, and makes possible to develop low power embedded software.

Keywords : Low power, Embedded software, Software architecture, Attribute-driven design, Architecture tactics

Ⅰ. 서 론

임베디드 소프트웨어는 현대 생활의 모든 전자 제품에 내장되어 제품의 기능을 제공하는 핵심적인 요소가 되었다. 많은 중소 업체들이 다양한 플랫폼의 임베디드 소프트웨어의 개발에 사업영역을 확장하면서 많은 소프트웨어들이 쏟아져 나오고 있다. 그러나 임베디드 소프트웨어의 특성상, 개발되는 소프트웨어들은 아주 미소한 기능의 차이를 갖는 다양한 버전의 제품 군(Product Family)에 내장되는 특성이 있다. 따라서 개발되는 소프트웨어의 많은 부분이 재사용되어 새로운 버전의 소프트웨어로 개발되고 있다[1, 2].

* Corresponding Author (jehong@chungbuk.ac.kr)
Received: 13 Feb. 2013, Revised: 11 Mar. 2013, 25 Mar. 2013, Accepted: 27 Mar. 2013.
J.W. Lee, J.E. Hong: Chungbuk National University.
※ 이 논문은 2012년도 정부(교육과학기술부)의 재원으로 한국연구재단의 기초연구사업 지원을 받아 수행된 것임 (2012-0004556)

ISSN : 1975-5066

이러한 상황에서 개발되는 소프트웨어의 유용성, 재사용성, 그리고 일관된 확장성을 제공하기 위해서는 해당 소프트웨어 제품군에 대한 아키텍처(architecture) 개발이 요구된다. 소프트웨어 아키텍처는 소프트웨어 시스템을 구성하는 소프트웨어 컴포넌트와 이들 간의 관계를 정의한 것으로써, 시스템이 갖는 사용자 관점의 행위를 인식할 수 있도록 개발하는 추상화된 설계 모델이다[3-5]. 소프트웨어 아키텍처가 소프트웨어 시스템 개발에서 중요한 이유는 아키텍처 개발의 제품군의 안정적이고 확정 가능한 소프트웨어의 기본 틀을 제공하는데 있다. 이로써 추후기능의 확장이나 새로운 버전의 소프트웨어 개발에서 바탕이 되는 청사진 역할을 수행한다.

최근에 임베디드 소프트웨어가 유비쿼터스 컴퓨팅 등과 같은 모바일 환경에서의 활용도가 높아지면서 소모전력에 대한 관심도가 높아지고 있다[6, 7]. 저전력을 소모하는 임베디드 시스템 개발에서 배터리 등을 포함하는 하드웨어 소자에 대한 소모전력 절감 기술이 연구되어 왔으며, 하드웨어의 움직임을 제어하는 소프트웨어에 대한 소모전력 절감

기법에 대한 연구도 활발히 진행되고 있다[8, 9]. 이런 관점에서 임베디드 소프트웨어의 아키텍처를 개발하는 과정에서 소모전력 특성을 고려하여 안정적인 소프트웨어의 기반 구조를 마련하고자 하는 요구가 발생하고 있다[10].

기존에 수행된 대부분의 연구들은 소스 코드를 기반으로 하는 소프트웨어 소모전력을 분석하여 왔다[11, 12]. 그러나 코드 중심의 분석을 통해 성능이나 소모전력과 같은 요구사항을 만족하지 못하는 경우 이를 해결하기 위한 많은 노력이 필요하게 되며, 최악의 경우에는 대부분의 프로그램을 다시 작성해야 하는 상황이 발생할 수도 있다. 따라서 본 연구에서는 이러한 문제점을 최소화하기 위한 목적으로 소프트웨어의 전체 구조가 확정되는 아키텍처 개발과정에서 소모전력의 특성을 고려하는 방법에 대하여 제시한다. 기존의 ADD(Attribute-Driven Design) 방법[13]에서 제시하는 소프트웨어 아키텍처 설계 기법을 기반으로 소모전력 특성을 어떻게 반영할 것인가에 대하여 제시한다. 이를 통해 일차적으로 아키텍처 차원에서 소모전력을 절감하는 효과가 있을 것이다.

논문의 2장에서는 기존의 소프트웨어 소모전력 분석 기법에 대한 연구와 ADD 방법에 대하여 살펴보고, 3장에서는 소모전력 특성을 반영한 아키텍처 설계 기법을 예시와 함께 설명한다. 4장에서는 본 연구에서 제시한 아키텍처 설계의 평가 요소를 제시하고, 5장에서는 결론 및 향후 연구에 대해 기술한다.

Ⅱ. 관련 연구

소프트웨어 소모전력에 대한연구는 10년 전부터 연구되어 왔지만 대부분 코드 중심의 소모전력 분석에 대한 연구들이며[11, 12], 코드가 아닌 설계 정보를 기반으로 소모전력을 분석하기 위한 연구는 최근 2-3년 동안 수행되어 왔다[14, 15]. 설계 정보를 기반으로 소프트웨어의 소모전력을 분석한 연구는 T. K. Tan에 의해 수행되었다[16].

Tan의 연구[16]에서는 하드웨어 디바이스와 드라이버, 소프트웨어의 태스크들을 소프트웨어 아키텍처 그래프(Software Architecture Graph, SAG)를 정의하였다. SAG는 노드를 태스크로 매핑하고, 노드와 노드 간에 발생하는 신호, 데이터 교환, 제어흐름 등을 트랜지션으로 정의한 후 소모전력을 예측한다. 사전에 정의된 에너지 모델을 이용하여

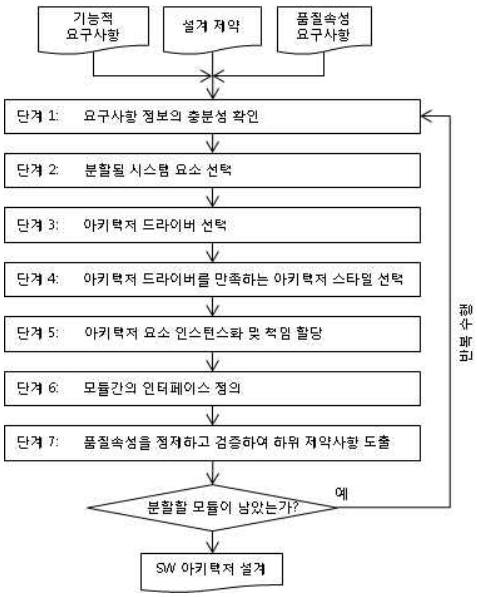

그림 1. ADD 기반의 소프트웨어 아키텍처 설계 절차[13]
Fig. 1 Design Process of ADD-based Software Architecture [13]

태스크 수행의 소모 전력량과 트랜지션의 소모 전력량을 예측한다.

Hong의 연구[17]는 UML 기반의 소프트웨어 개발에서 클래스와 시퀀스 다이어그램과 같은 설계 모델을 기반으로 하는 소모전력분석 기법에 대하여 제안하였다. 이 연구에서는 시퀀스 모델이 갖는 소프트웨어의 동작을 하위 수준의 명령어로 매핑하고 이들에 대한 소모전력을 분석한다. 이를 통해 에너지 효율적인 설계 모델을 결정하는데 도움을 주는 장점을 제공한다. 그러나 Tan의 연구나 Hong의 연구 모두 작성된 설계 모델을 입력으로 하여 소모전력을 분석하는 것에 주안점이 있으며, 본 연구에서는 소모전력 특성을 반영하는 아키텍처 모델을 개발하는 것에 주안점이 있다.

품질속성을 기반으로 아키텍처 설계를 하는 대표적인 방법은 ADD 방법[13]이 있다. ADD는 소프트웨어가 달성해야 할 품질속성을 기반으로 아키텍처를 설계하는 방법으로 그림 1과 같은 절차에 의해 수행된다. ADD를 수행하기 위해 기능적 요구사항, 설계제약, 품질 속성이 입력 산출물로 사용된다. 단계별 ADD 수행절차는 다음과 같다.

- 단계 1에서는 대상시스템에 대한 소프트웨어 요구사항의 충분성을 점검한다. 이는 충분한 요구

사항이 소프트웨어 아키텍처의 설계 결정을 내리는데 중요한 역할을 수행하기 때문에 아키텍처 설계의 초기 단계에 매우 중요한 활동이다.

- 단계 2에서는 분해할 모듈을 선택한다. 초기에는 시스템을 하나의 모듈로 가정하여 선택하며, 점진적으로 시스템 구성 요소들에 대하여 반복적인 설계 작업이 진행된다.
- 단계 3에서는 선택된 모듈에 대한 아키텍처 드라이버를 선택한다. 아키텍처 드라이버는 아키텍처가 가져야할 기능적 요구사항과 품질 속성으로 정의한다. 아키텍처 설계에서 가장 큰 영향을 미칠 수 있는 요구사항 및 품질 속성을 드라이버로 선택한다.
- 단계 4에서는 아키텍처 드라이버들을 달성할 수 있는 방안을 고려하여 아키텍처 패턴을 선택한다. 아키텍처 패턴은 요구사항과 품질 속성을 만족하는 초기 설계 모델로써, 아키텍처를 완성하기 위한 기반 모델이 된다.
- 단계 5에서는 선택한 패턴에 나타난 모듈들을 인스턴스화하고, 이들에게 기능을 할당한다.
- 단계 6에서는 모듈간의 인터페이스를 정의한다.
- 마지막 단계 7에서는 각 모듈이 갖는 제약사항의 만족여부를 점검한다.

이러한 단계를 거쳐 생성된 하위 모듈들에 대하여 더 이상의 분할이 필요하지 않을 때까지 각 단계를 반복 수행하여 소프트웨어의 아키텍처가 작성된다.

Ⅲ. 저전력 소프트웨어 아키텍처 설계

1. 고려사항

저전력 소프트웨어를 개발하기 위해서는 기본적으로 다음과 같은 사항을 고려할 수 있으며, 이러한 고려사항들은 아키텍처 설계에 잘 반영될 수 있어야 한다.

- 아키텍처 구성 모듈간의 상호 작용을 최소화함으로써, 글로벌 메모리와 버스 사용을 줄인다.
- 행위 특성이 상이한(동기, 비동기, 주기적 행위 특성) 동작을 포함하는 모듈을 분할한다.
- 병렬처리 가능한 부분을 식별하여 서로 다른 자원으로 할당한다.

이러한 고려사항들은 ADD 기반의 아키텍처 설계과정에서 반영되어야 하는데, 단계 4에서 진행하는 아키텍처에 대한 설계개념의 선정에서 소모전력 절감을 위한 설계 전략이 정의되어야 하며, 이를 기반으로 아키텍처 패턴의 인스턴스에 대한 기능할당 등에 위의 고려사항이 반영되어야 한다[18].

2. 아키텍처 설계

그림 1에 나타난 ADD 수행절차를 근간으로 소모전력 특성을 반영한 아키텍처 설계 기법을 단계적으로 제시하면 다음과 같다.

2.1 단계 1: 요구사항 정의 및 확인

본 연구에서 제시하는 저전력 소프트웨어를 위한 아키텍처 설계 기법을 설명하기 위하여 다음과 같은 대상 시스템의 요구사항을 정의한다. 대상시스템, GDO(Garage Door Opener)는 다음과 같은 기능을 수행한다.

- R1: 차고지의 문을 열고 닫는 자동화 시스템이다.
- R2: 문의 개폐는 스위치나 리모트 컨트롤러, 또는 HIS(Home Information System)에 의해 이루어질 수 있다.
- R3: 시스템의 안전성을 고려하여 차고 문을 열고 닫을 때, 장애물과의 접촉을 탐지해야 한다.
- R4: 시스템은 HIS에 의해 이상 유무 진단 등의 기능을 수행해야 한다.
- R5: 시스템은 전력소모를 최소화 하는 방향으로 개발되어야 한다.

또한 이 시스템의 비즈니스 목표와 드라이버는 제품 라인 아키텍처를 설계하는 것이며, 이때 소모전력에 대한 품질 요구사항을 반영하여야 한다.

2.2 단계 2: 분할 모듈 선택

ADD 기법에서는 먼저 대상 시스템의 요구사항과 품질 속성을 반영하기 위한 대상 모듈을 선택하게 된다. 초기 단계에서는 대상 시스템 전체가 분할 대상 모듈이 될 수 있다. 선택된 모듈은 ADD 설계 절차에 따라 요구사항에 의거한 기능을 수행하기 위한 하위 모듈로 분할되며, 단위 기능이 모듈에 할당될 때까지 반복적으로 분할 과정을 수행한다. 위의 GDO 시스템으로부터 초기에 선정되는 대상 모듈은 GDO 시스템 전체이다.

2.3 단계 3: 아키텍처 드라이버 선택

이 단계에서는 요구사항이 아키텍처에 미치는 영향에 따라 우선순위를 설정하여 핵심 아키텍처 드

라이버를 선택 한다. 예제 시스템의 아키텍처 드라이버들은 다음과 같다.

- D1: 문 개폐장치와 제어는 제품모델 기반의 제품군마다 다르다.
- D2: 제품라인에서 제품마다 프로세서가 다르다.
- D3: 문이 작동하는 동안 장애물이 발견되면 0.1초 내에 정지되거나 다시 반대로 작동한다.
- D4: 시스템이 소모하는 전력을 최소화해야 한다.

2.4 단계 4-1: 아키텍처 설계 전략

선택된 아키텍처 드라이버를 만족할 수 있는 설계 전술을 선택한다. GDO 시스템에서는 다음과 같은 세 가지 설계 전략을 선택한다.

- T1: 추상화 및 정보 은닉 : 소프트웨어에서의 모듈화는 의미적 함축과 정보 은닉을 높이기 위한 전술로 선택할 수 있다. 또한 제품라인마다 다른 개폐장치, 제어, 프로세스의 차이점(드라이버 D1)은 모듈에 대한 상세정보의 은닉을 통하여 감춰진다. 따라서 그림 2와 같이 제품라인마다 선택하는 제어나 프로세서(드라이버 D2)는 블랙박스 개념 하에 대체될 수 있도록 설계한다.
- T2: 데드라인(Deadline) 관리 : GDO 시스템의 아키텍처 드라이버 D3을 만족시키기 위해서는 시간 제약사항을 해결하기 위한 프로세스 스케줄러가 필요하다. 따라서 전반적인 시스템의 데드라인을 만족시키기 위한 방법으로 우선순위 기반의 스케쥴링 기법을 전술로 선택할 수 있다. 데드라인을 위한 아키텍처 전략은 그림 3과 같다.
- T3: 저전력 소모 : 그림 4의 아키텍처 전략은 GDO 시스템의 운영에 있어서 소모전력을 최소화, 모듈간의 결합력 감소 등과 같은 설계 원리가 아키텍처 설계에 반영될 수 있도록 한다.

2.5 단계 4-2 아키텍처 패턴 선택

아키텍처 설계 전략을 선택한 후, 이러한 전략이 잘 반영될 수 있는 아키텍처 패턴을 선정한다. 아키텍처 패턴은 소프트웨어 요구사항을 만족할 수 있는 일반화된 솔루션을 제공하는 모델로써, GDO 시스템의 경우에는 성능, 스케줄링, 운영체제 지원(Virtual Machine) 등과 같은 컴포넌트들로 구성된 아키텍처 패턴을 선택할 수 있다. 그림 5는 GDO 시스템에 대한 아키텍처 패턴을 나타낸다. 그림 5로부터 GDO 시스템의 요구사항 R1과 R4는 "Non-Performance Critical Computation" 패키지

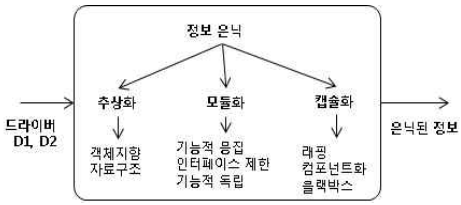

그림 2. 정보은익을 위한 아키텍처 전략
Fig. 2 Architectural Tactics for Information Hiding

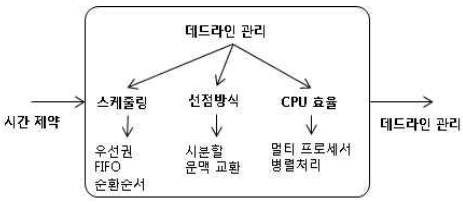

그림 3. 데드라인을 위한 아키텍처 천략
Fig. 3. Architectural Tactics for Deadline Management

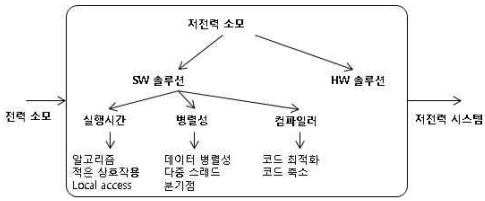

그림 4. 저전력 소모에 대한 아키텍처 전략
Fig. 4 Architectural Tactics for Low power Consumption

타입에서 구현될 수 있으며, 요구사항 R2는 서로 다른 장치와의 통신을 담당하는 "Virtual Machine" 패키지 타입으로 구현 가능하다. 또한 R3 요구사항은 성능과 관련된 "Performance-Critical Computation" 패키지로부터 구현될 수 있다. 소모전력에 대한 요구사항 R5는 품질 요구사항에 해당되며, 패턴 구성요소에 대한 제약사항으로 정의하였다.

2.6 단계 5: 인스턴스 생성 및 기능 할당

단계 5에서는 선정된 아키텍처 패턴에 대한 인스턴스 모듈을 생성하고, 각 모듈에 대하여 기능, 데이터, 정보 등을 할당하는 과정이다. 단계 4-2에서 설명한 내용을 근간으로 그림 5로부터 그림 6과 같은 인스턴스 모델을 생성할 수 있다. 그림 5에 나타난 GDO 시스템의 모듈이 복수의 기능 요구사항을

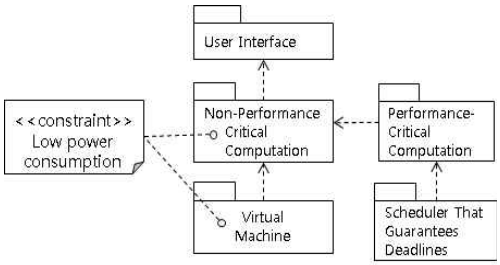

그림 5. GDO 시스템의 아키텍처 패턴
Fig. 5 Architecture Pattern for GDO System

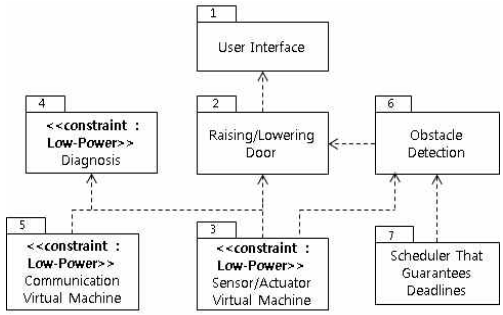

그림 6. GDO 아키텍처의 인스턴스
Fig. 6 An Instance of GDO Architecture.

표 1. GDO 시스템 기능과 인스턴스 모듈간의 매핑
Table 1. Mapping of Functionalities and Instance Modules

패턴 요소	인스턴스 모듈	기능 할당	요구사항	드라이버
User Interface	User Interface	사용자 요청 인식	R2	D1, D2
Non-Performance Critiacal Comp.	Rising/Lowering Door	차고문 개/폐 제어	R1	D3
	Diagnosis	진단 기능 수행	R4, R5	D4
Virtual Machine	Communication VM	HIS와의 통신 관리	R4, R5	D4
	Sensor/Actuator VM	모터와 센서의 제어	R2, R5	D1, D2, D4
Performance-critical Comp.	Obstacle Detection	센서로부터의 신호 체크	R3	D3
Scheduler	Scheduler	시스템의 운용 스케줄 관리	R3	D3

담당하는 경우, 이들은 별도의 인스턴스로 생성하여 단일 기능을 갖도록 표현한다. "Non-Performance Critical Computation" 패키지로부터 "Diagnosis" 모듈과 "Raising/Lowering Door" 모듈을 인스턴스화 하였다. "Virtual Machine" 패키지는 HIS와 통신을 담당하는 모듈과 센서/작동기를 제어하는 모듈로 분할하여 인스턴스화 하였다. 소모전력의 특성을 유지하기 위한 인스턴스 모듈은 다음 단계의 설계 과정에서 적용되기 위해 해당 모듈의 제한사항(constraint)으로 스테레오타입을 부여하였다. 표 1은 패턴의 구성요소와 이에 대한 인스턴스 모듈, 그리고 각 모듈에 할당된 기능을 정의한 표이다.

2.7 단계 6: 모듈 인터페이스 정의

다음 단계로 개발된 소프트웨어 아키텍처의 실체화된 모듈들에 대해 기능을 파악하고, 각 모듈의 인터페이스를 정의 한다. 모듈간의 인터페이스는 그림 6에 나타난 각 모듈 인스턴스간의 상호작용에 대하여 표현하는 UML의 상호작용 다이어그램을 이용하여 나타낼 수 있다. 그림 7은 차고 문을 열고(Door_Up), 닫고(Door_Down), 그리고 진단(Diagnosis)하기 위한 행위의 다이어그램을 보여준다. 그림 7에 나타난 것과 같이 아키텍처의 각 모듈간에 상호 작용하는 메시지를 표현하였으며, 이를 기반으로 모듈 인터페이스를 정의할 수 있다. 그림에서 보는 바와 같이 모듈 “Raising/Lowering Door"와 모듈 ”Sensor/Actuator VM"간의 데이터 상호작용이 많음을 알 수 있다.

Ⅳ. 아키텍처 뷰 개발 및 검증(단계 7)

그림 6의 GDO 시스템에 대한 아키텍처 모델에 대하여 아키텍처를 이루는 각각의 모듈이 할당된 기능의 품질 요구사항을 만족하는지 확인하기 위하여 다음과 같은 세 가지 측면의 뷰 모델을 개발한다.

- 기능적 상호작용 뷰: 아키텍처 구성 모듈간의 결합 복잡도를 표현한다.
- 병렬성 뷰: 아키텍처 모듈들을 동기화 관점에서 표현한다.

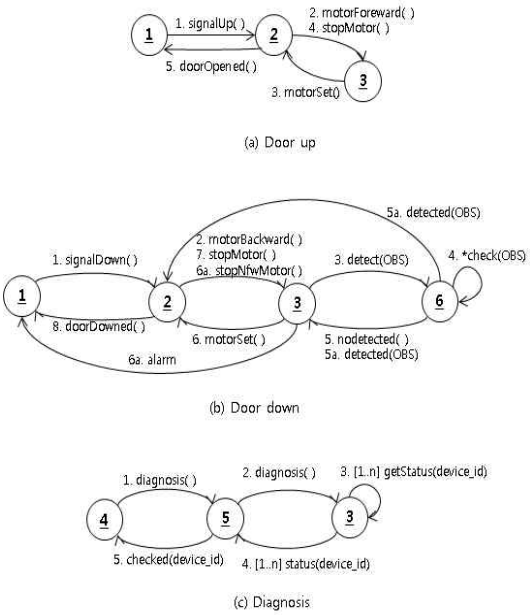

그림 7. GDO 시스템 모듈 인스턴스간 상호작용 다이어그램

Fig. 7 Communication Diagrams for Module Instance of GDO

- 배치 뷰: 아키텍처 모듈들이 물리적 구성요소로 배치되는 모습을 표현한다.

아키텍처의 뷰 모델을 작성하기 위하여 먼저 본 논문에서 고려하는 품질 속성인 소모전력 절감 특성을 반영하기 위하여 아키텍처를 구성하는 모듈간의 상호작용 매트릭스를 정의한다. 이 매트릭스를 통해 (1) 모듈간의 병렬성 결정, (2) 모듈을 물리적 구성요소로의 배치 등과 같은 정보를 추출할 수 있다. 참고로 이들은 3.1절에서 언급한 소모전력 절감의 고려사항 맥락을 같이한다. GDO 시스템에 대한 상호작용 매트릭스는 표 2와 같다.

4.1 기능적 상호작용 뷰

소프트웨어 아키텍처를 구성하는 모듈간의 상호작용을 최소화하는 것은 모듈간에 전달되는 메시지 패싱, 메소드 호출을 위한 오버헤드 등을 감소시킴으로써, 소프트웨어 시스템의 성능에 대한 요구사항 만족도를 높이고자 하는 것이다. 또한 이러한 상호작용의 축소는 결국 하드웨어 자원 사용을 감소시키며 소모전력 감소효과를 가져오게 한다. 그림 7에 나타난 GDO 시스템의 상호작용 다이어그램에 대한

표 2. 아키텍처 구성요소간의 상호작용 매트릭스

Table 2. Interaction Matrix between Architectural Components

컴포넌트	User Interface	Diagnosis	Comm. VM	Sensor VM	Rising/ Low. Door	Obs. detect.	Scheduler
User Interface	+				Sync		
Diagnosis		+	IPC	IPC			
Comm. VM		IPC	+		Sync		
Sensor VM		IPC		+	Sync		
Rising/ Low. Door	Sync		Sync	Sync	+	Sync	
Obstacle detection					Sync	+	Sync
Scheduler						Sync	+

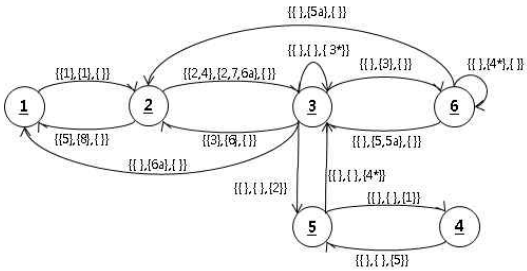

그림 8. GDO 시스템의 통합 상호작용 다이어그램

Fig. 8 Integrated Communication Diagram of GDO System

통합 다이어그램은 그림 8과 같다. 그림 8에서 나타난 각 모듈간의 상호작용 메시지는 “Door-Up”, “Door-Down”, 그리고 “Diagnosis”의 행위에 있어서 전달되는 메시지를 순차적으로 표현한 것이다. 모듈 1과 2간의 메시지는 4개, 모듈 2와 3간의 메시지는 7개, 모듈 3과 6간의 메시지는 3개 등임을 알 수 있다. 따라서 모듈간의 상호 작용을 감소시키기 위해서는 모듈 2와 3이 통합되거나, 단일 프로세서로 할당되어야 한다.

4.2 병렬성 뷰

아키텍처 구성 모듈들의 병렬 처리 특성을 살펴보는 것은 시스템의 성능과 실행 시간에 대한 품질 속성을 점검하는 것이라 할 수 있다. 상호 연결성을 갖지 않는 모듈들을 병렬 수행하게 함으로써 시스템의 실행 시간을 단축 시켜 소모 전력 절감 효과를 얻을 수 있다. 프로세서의 트랜지션 당 소모전력은 수

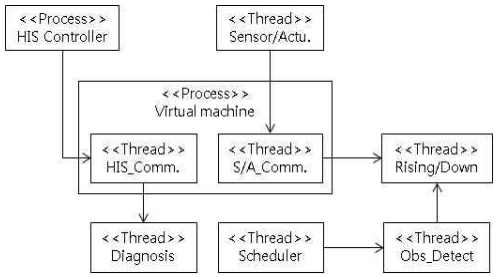

그림 9. GDO 시스템의 병렬성 뷰
Fig. 9 Concurrency View of GDO System

식 (1)과 같이 나타낼 수 있다[19].

$$P = aCV2 + LV \quad (1)$$

여기서 a는 트랜지션 상수, C는 스위치 용량, L은 누전상수, V는 공급전압 이다.

공급전압은 주파수(f)에 비례하기 때문에, 수행시간 t에 대한 소모전력은 수식 (2)와 같다.

$$P = (Kf2 + Lf)t \quad (2)$$

여기서 K는 aC, f는 주파수, t는 시간 이다.

이때 주파수를 $m > 1$배 증가시키면 수행 시간은 t/m로 줄어들기 때문에 소모전력은 다음 수식과 같다.

$$P = (K(mf)2 + Lmf)t/m = (Kfm + L)ft \quad (3)$$

또한 프로세서를 n으로 증가시키는 경우의 속도 향상을 b라 하면 소모전력은 다음 수식과 같다.

$$P = (Kf + L)ft(n/b) \quad (4)$$

수식 (3)과 (4)에 의해 $b > n/m$ 인 경우, 프로세서를 증가시키면 동적에너지 소모는 감소하고 누수전력이 커진다. 따라서 프로세서의 누수전력, 즉 대기상태의 전력소모가 적은 경우 시스템의 병렬성을 증가시키는 것은 소모전력 감소전략이 될 수 있다. 그림 9의 병렬성 뷰는 GDO 시스템이 갖는 병렬성을 나타낸다. “HIS_Comm.” 쓰레드는 “S/A_Comm.”, “Scheduler”쓰레드와, “S/A_Comm.” 쓰레드는 “Diagnosis” 쓰레드와 상호 연결성을 갖지 않음을 확인할 수 있으므로 이들 간의 병렬 수행이 가능할 수 있다.

4.3 Deployment View

네트워크로 구성된 시스템에서는 프로세서간의 메시지 전달을 수행할 때 네트워크 자원의 사용에 따른 추가적인 연산이 발생하며, 이에 따른 소모전력은 일반적으로 다음 수식과 같이 계산된다[20].

$$E = mX + c \quad (5)$$

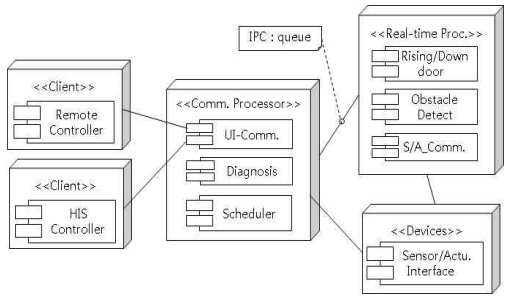

그림 10. GDO 시스템의 배치 뷰 1
Fig. 10 Deployment View 1 of GDO System

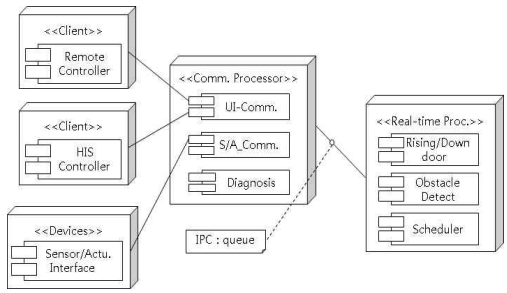

그림 11. GDO 시스템의 배치 뷰 2
Fig. 11 Deployment View 2 of GDO System

여기서 E는 에너지, m는 메시지 송/수신당 에너지 소모, X는 메시지 크기, c는 상수 이다.

그림 8의 상호작용 분석에서 “Riging/Down door" 모듈과 ”S/A_Communication" 모듈간에 메시지 전달이 빈번한 것으로 나타났다. 따라서 두 모듈을 단일 프로세서에 할당하여 상호작용을 축소시킬 수 있어야 한다. 모듈간의 상호작용을 감안한 물리적 매핑은 그림 10과 같이 나타낼 수 있다.

그림 9로부터 GDO 시스템이 갖는 병렬성을 확인하였다. 시스템 기능의 최대 병렬성은 GDO 시스템의 동작을 안전하게 수행할 수 있는 범위를 고려하여 제한할 수 있다. 병렬 수행의 제한은 그림 11의 배치 뷰와 같이 물리적인 병렬성으로 매핑할 수 있다. 이러한 배치는 병렬성을 고려하는 저전력 소모, 모듈 기능간의 응집력 등을 고려하는 방향으로 설계 결정이 이루어진 모습이다.

Ⅴ. 아키텍처 평가

아키텍처를 평가하는 것은 비기능적 요구사항이 아키텍처 설계에 잘 반영되었는지 확인하고, 아키

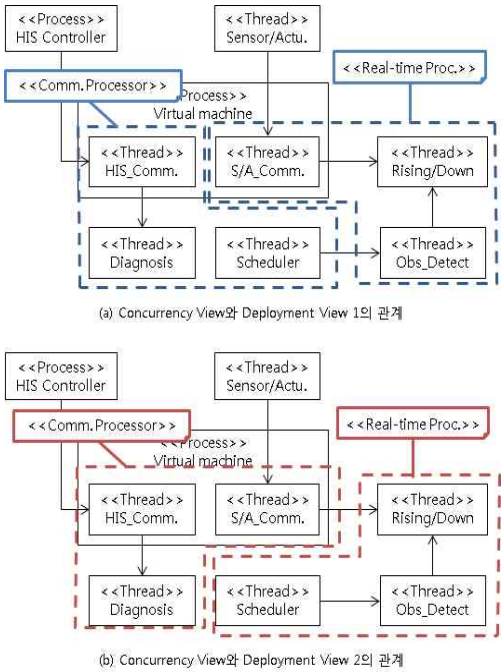

그림 12. 병렬성 관점과 배치 관점의 매핑
Fig. 12 Mapping of Concurrency View and Deployment View

텍처 설계에 결함이 존재하는지 확인하는데 목적이 있다. 또한 다수의 아키텍처 후보가 존재하는 경우 아키텍처 평가를 통해 요구사항을 가장 잘 만족시키는 후보를 선택할 수 있다.

GDO 시스템에 대한 배치 뷰는 모듈간의 상호작용을 반영한 것과 병렬성을 반영하여 설계한 두 개의 후보가 생성되었다. 따라서 본 논문에서는 두 후보를 모듈간의 상호작용 측면과 병렬성 측면에서 평가한다.

5.1 병렬성 측면의 평가

GDO 시스템의 병렬성 뷰는 상호 연결성을 갖지 않는 모듈들을 병렬 수행하게 하여 시스템의 실행시간을 단축시켜 소모전력 절감 효과를 얻게 하기 위해 설계되었다. 병렬성 뷰와 배치 뷰 후보들간의 매핑은 그림 12와 같이 나타낼 수 있다. 배치 뷰에서 다수의 모듈이 포함된 노드는 <<Comm. Processor>>와 <<Real-time Proc.>>이며, 상호연관성을 갖지 않는 모듈은 각각 한 개씩 포함되어 있다. 따라서 병렬성 측면에서 두 배치 뷰 후보 간의 차이가 크지 않음을 알 수 있다.

표 3. 시나리오에 근거한 배치 관점의 상호작용 비교
Table 3. Interaction Comparison for Deployment Views by Scenarios

시나리오	메시지 전달 횟수	
	View 1	View 2
Door up	2	5
Door down 1	2	4
Door down 2	2	5
Diagnosis	4	4

5.2 상호작용 측면의 평가

배치 뷰 후보 간의 상호작용 평가는 그림 7의 상호작용 다이어그램을 기반으로 정의한 시나리오를 이용해 수행한다. 그림 7의 다이어그램은 “Door up”, “Door down”, “Diagnosis” 세 가지 행위에 대한 메시지 전달을 나타낸다. “Door up"과 "Diagnosis" 행위는 메시지 전달 순서에 따라 각각 한 개의 시나리오가 생성되며, “Door down” 행위의 경우 {1, 2, 3, 4, 5, 6, 7, 8}의 메시지 순서를 갖는 시나리오 1과 {1, 2, 3, 4, 5a, 6a} 순서의 메시지 순서를 갖는 시나리오 2를 정의할 수 있다. 표 3은 배치 뷰에서 각각의 시나리오에 대한 노드간 메시지 전달 횟수 비교이다.

표 3에서 배치 뷰의 각 노드간 상호작용을 평가한 결과 후보 1의 메시지 전달 횟수가 후보 2에 비하여 약 50%가량 적은 것을 알 수 있다. 따라서 배치 뷰 1을 GDO 시스템의 아키텍처로 선정하는 것이 시스템 성능을 향상시키고, 소모전력을 감소시킬 수 있는 것으로 판단할 수 있다.

Ⅵ. 결론 및 향후 연구

다양한 플랫폼의 임베디드 소프트웨어들이 개발되면서 이에 대한 요구사항들 또한 증가하고 있다. 특히 배터리를 이용해 동작하는 시스템들에 대해서는 전력소비가 적은 소프트웨어에 대한 요구가 증가하고 있다. 따라서 소프트웨어 설계 단계에서부터 소모전력 특성을 반영하는 것이 중요한 전략으로 인식되고 있다.

본 논문에서는 ADD 방법론에 기반하여 소모전력 특성을 아키텍처 설계에서 반영하는 기법을 기술하였다. 이는 소프트웨어 개발의 초기 단계에서 수행되므로, 임베디드 소프트웨어에 대한 요구사항을 반영할 수 있을 것으로 판단된다. 특히 아키텍

처 후보들간의 비교 평가를 통해 시스템 성능과 저전력 소모에 대한 요구사항을 만족시킬 수 있는 아키텍처를 선택할 수 있음을 보였다.

향후 연구에서는 소프트웨어 아키텍처 기반의 전력소모량 정량화 기법에 대한 연구를 진행할 것이다.

References

[1] P. Mohagheghi, R. Conradi, "Quality, productivity and economic benefits of software reuse: a review of industrial studies," Empirical Software Engineering, Vol. 12, No. 5, pp.471-516, 2007.

[2] Y.C. Cheng, C.T. Chen, J.S. Jwo, "Exception handling: An Architecture Model and Utility Support," Proceedings of Asia-Pacific Software Engineering Conference, pp.1-8, 2005.

[3] L. Bass, P. Clements, R. Kazman. "Software Architecture in Practice," Addison Wesley, 2003.

[4] J.D. McGregor, "Software Architecture," Journal of Object Technology, Vol. 3, No. 2, pp.65-77, 2004.

[5] A.H. Eden, R. Kazman, "Architecture, Design, Implementation," Proceedings of International Conference on Software Engineering, pp.1-11, 2003.

[6] T. Arsan, E. Baskan, E. Ar, Z. Bozkus, "A Software Architecture for Inventory Management System," Lecture Noetes in Electrical Engineering, Vol. 152, pp.15-27, 2013.

[7] T. Kim, H. Kim, "A Software Architecture ofr Highly Reconfigurable Sensor Operating Systems," Journal of IEMEK, Vol. 2, No. 4, pp.242-250, 2007 (in Korean).

[8] E. Carneiro, P. Maciel, G. Callou, E. Tavares, B. Nogueira, "Mapping SysML State Machine Diagram to Time Petri net for Analysis and Verification of Embedded Real-Time Systems with Energy Constraints," Proceedings on International Conference of Advances in Electronics and Micro-electronics, pp.1-6, 2008.

[9] B. Nogueira, P. Maciel, E. Tavares, E. Andrade, R. Massa, G. Callou, R. Ferraz, "A Formal Model for Performance and Energy Evaluation of Embedded Systems," EURASIP Journal on Embedded Systems, Vol. 2011, No. 2, pp.1-13, 2011.

[10] K.H. Kim, Y. Kim, J. Kim, "Power-aware Real-time Task Scheduling in Dependable Embedded Systems," Journal of IEMEK, Vol. 3, No. 1, pp.25-29, 2008 (in Korean).

[11] E. Senn, J. Laurent, N. Julien, E. Martin, "Softexplorer: estimating and optimizing the power and energy consumption of a C program for DSP applications," EURASIP Journal on Applied Signal Progressing, Vol. 2005, No. 1, pp.25-29, 2005.

[12] A. Muttreja, A. Raghunathan, S. Ravi, N.K. Jha, "Hybrid Simulation for Energy Estimation of Embedded Software," IEEE Transaction on Computer-AIDED Design of Integrated Circuits and Systems, Vol. 26, No. 10, pp.1843-1854, 2007.

[13] R. Wojcik, F. Bachmann, L. Bass, P. Clements, P. Merson, R. Nord, B. Wood, "Attribute-Driven Design (ADD)," Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, Technical Report CMU/SEI-2006-TR-023, 2006.

[14] X. Yue, Z. Xuehai, L. Xi, G. Yuchang, "OOEM: Object-Oriented Energy Model for Embedded Software Reuse," Proceedings on IEEE International Conference of Information Reuse and integration, pp.551-558, 2003.

[15] H. Jun, L. Xuandong, Z. Guoliang, W. Chenghua, "Modeling and Analysis of Power Consumption for Component-Based Embedded Software," Proceedings on Embedded Ubiquitous Computing Workshops, pp.795-804, 2006.

[16] T.K. Tan, A. Raghunathan, N.K. Jha, "Software Architectural Transformations: A New Approach to Low Energy Embedded Software," Proceeding on Design, Automation & Test in Europe, pp.1046-1051, 2003.

[17] D.H. Kim, J.P. Kim, J.E. Hong, "A Power

Consumption Analysis Technique Using UML-Based Design Models in Embedded Software Development," Lecture Notes in Computer Science, Vol. 6543, pp.320-331, 2011.

[18] S.T. Kim, D.K. Kim, L. Lu, S.Y. Park, "Quality-driven architecture development using architecture tactics," Journal of System and Software, Vol. 82, No. 8, pp.1211-1231, 2009.

[19] R. Jejurikar, C. Pereira, R. Gupta, "Leakage Aware Dynamic Voltage Scaling for Real-Time Embedded Systems," Proceedings on the Annual Design Automation Conference, pp.275-280, 2004.

[20] S. Chinara, S.K. Rath, "Energy Efficient Mobility Adaptive Distributed Clustering Algorithm for Mobile Ad Hoc Network," Proceedings on International Conference of Advanced Computing and Communications, pp.265-272, 2008.

저 자 소 개

이 재 욱

2010년 충북대학교 컴퓨터공학부 학사.
2011년 충북대학교 컴퓨터과학과 공학석사.
현재, 충북대학교 컴퓨터과학과 박사과정.
관심분야: 소프트웨어 품질공학, 정형 기법, 저전력 소프트웨어, 소프트웨어 모델링 등.
Email: jwlee@selab.cbnu.ac.kr

홍 장 의

2001년 KAIST 전산학과 공학박사.
1990년 ~ 1995년 국방과학연구소 선임연구원.
2002년 ~ 2004년 (주)솔루션링크 기술연구소장.
현재, 충북대학교 소프트웨어학과 교수.
관심분야: 소프트웨어 품질공학, 모델기반 검증분석, 소프트웨어 아키텍처, 저전력 소프트웨어, 소프트웨어 프로세스 개선 등
Email: jehong@chungbuk.ac.kr