# 내장형 실시간 시스템의 소프트웨어 아키텍처 평가 절차

권도형°, 최윤석, 이우진, 정기원
숭실대학교 대학원 컴퓨터학과
kdh906@empal.com, bluewj@nate.com, secooling@hanafos.com, chong@comp.ssu.ac.kr

# A Software Architecture Evaluation Procedure In Realtime Embedded Systems

Dohyung Kwon°, Yunseok Choi, Woojin Lee, Kiwon Chong
Soongsil Graduate University

**요 약**

내장형 실시간 시스템은 점차 소형화, 다기능화 하여 그 복잡도가 증가하고 있다. 또한 제약사항이 일반적인 정보시스템 보다 더 까다롭다. 신뢰성 있고 안정적인 내장형 실시간 시스템을 구현하기 위해서 소프트웨어 공학의 적용이 필요하며, 특히 소프트웨어 아키텍처의 적용을 필요로 한다. 이에 본 논문에서는 내장형 실시간 시스템 개발 중 최적의 소프트웨어 아키텍처를 선정하기 위해서 필요한 소프트웨어 아키텍처 평가 절차를 제시한다. 측정기법(Measuring Techniques)과 질문기법(Questioning Techniques)을 혼합하여 분석을 수행한다. 측정기법을 위해서는 실 프로토타이핑(Real Prototyping)을 사용하며 질문기법을 위해서는 체크리스트를 사용한다. 이를 통하여 소프트웨어 개발 생명주기의 초기단계에서 미리 목표시스템의 품질을 예측하고 위험을 미리 식별할 수 있다.

## 1. 서 론

내장형 실시간 시스템은 특정 기능을 수행하기 위한 소프트웨어와 이를 구동하기 위한 하드웨어로 구성된다. 내장형 실시간 시스템의 응용범위는 점차 확대되고 있으며, 시스템 자체의 구성은 소형화, 다기능화 함에 따라 시스템의 복잡도가 증가하고 있다. 특별히 내장형 실시간 시스템은 여러 제약사항이 존재한다. 예를 들어 시간 제약사항(Temporal Constraints)을 들 수 있다. 시간 제약사항의 충족 여부는 매우 중요하고도 치명적인 요소이다[1,2]. 내장형 실시간 시스템의 복잡도가 증가하면 그 만큼 위험성이 높아지는 문제가 발생한다. 이를 해결하기 위해서는 내장형 실시간 시스템의 개발에도 소프트웨어 공학적인 접근을 필요로 하게 한다.

소프트웨어 아키텍처는 기능적인 고려사항 뿐만 아니라 비기능적인 고려사항을 다루며 설계의 기반이 된다[3]. 비기능적 제약사항이 중요한 내장형 실시간 시스템 개발에 소프트웨어 아키텍처를 적용하는 것은 매우 필요하며 적절하다. 안전한 시스템(Safety-Critical System)을 구현하기 위해서 시스템에 적용할 소프트웨어 아키텍처가 올바른 것인가를 평가하는 것 역시 중요하다. 현재 범용적인 소프트웨어 아키텍처 평가 방법이 있으나 내장형 실시간 시스템의 특징을 반영한 소프트웨어 아키텍처 평가 방법이 부재하다[3,4]. 이에 본 논문에서는 내장형 실시간 시스템의 특성을 반영하는 소프트웨어 아키텍처 평가 방법을 제안하고 그에 대한 절차를 제안한다.

## 2. 관련 연구

소프트웨어 아키텍처는 복잡한 시스템을 분할하고 구조화하여 전체 시스템의 가시성을 제공하고, 이해 관계자가 시스템의 목적, 범위, 기능 등을 명확히 이해할 수 있도록 도와준다. 그러므로 시스템의 복잡도가 증가할수록 아키텍처 기술의 적용이 필요하다.

### 2.1 평가 방법 분류

소프트웨어 아키텍처 평가 방법은 크게 두 가지로 분류 할 수 있다. 질문기법(Questioning Techniques)과 측정기법(Measuring Techniques)이다[5]. 질문기법은 해당 소프트웨어 아키텍처의 품질속성에 대해 정성적인 질문(Qualitative Questions)을 하고 이에 답을 하는 방법이다. 여기에는 체크리스트, 시나리오 등을 적용한다. 측정기법은 정량적인 측정(Quantitative Measurements)을 한다. 여기에는 메트릭스(Metrics), 시뮬레이션, 프로토타입과 경험치(Experience)들을 사용한다[5]. 질문기법은 시나리오를 이용하여 소프트웨어 아키텍처를 표현한다. 이를 바탕으로 논리적으로 소프트웨어 아키텍처를 평가한다. 이 방법만으로 품질속성을 만족시키며 위험을 식별하여 제어하기에는 불충분하다. 측정기법은 이러한 질문기법을 보안하여 양적인 분석을 제공함으로 보다 객관적인 정보를 바탕으로 한 평가를 제공할 수 있다[5].

### 2.2 소프트웨어 아키텍처 평가 방법

소프트웨어 아키텍처 평가의 목적은 소프트웨어 아키텍처 분석을 통해 해당 아키텍처가 가지는 위험을 발견하고, 이해당사자가 원하는 특정 품질속성을 보다 잘 반영하기 위함이다. 따라서 특정 품질속성에 민감한 지점(Sensitivity Point)을 발견하고 여러 품질속성 간의 Trade-off 지점을 발견하는 것이 중요하다. 대표적인 아키텍처 평가 방법으로 SAAM(Software Architecture Analysis Method)과 ATAM(Architecture Tradeoff Analysis Method)이 있

다[4]. 또한 이 방법들로부터 확장한 방법들이 존재한다[5].

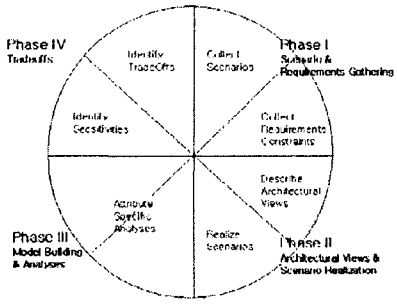


그림 1 ATAM 프로세스

그림1에서 보는 바와 같이 ATAM은 총 4개의 단계로 구성되어 있으며,각 단계는 두 개의 세부 단계로 구성된다. ATAM은 아키텍처 분석결과를 정량적인 형태로 제공하지 않으므로 처리속도 평가와 같은 객관적인 정보를 요구하는 분석의 경우 어려움이 있다. 또한 구체적인 기법을 제공하고 있지 않다.

## 3. 내장형 실시간 시스템의 소프트웨어 아키텍처 평가절차

### 3.1 전체 절차

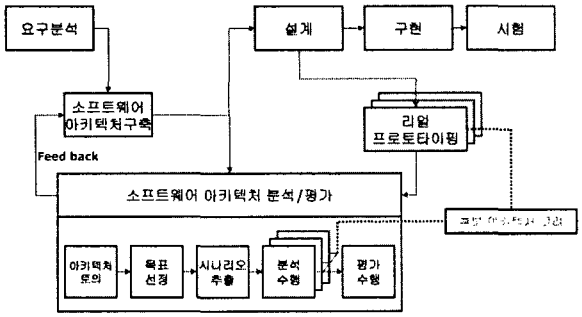


그림 2 소프트웨어 아키텍처 평가 절차

여러 소프트웨어 개발 프로세스 모델이 존재하지만 일반적으로 분석, 설계, 구현, 시험, 유지보수의 단계가 필요하다. 여기서 소프트웨어 아키텍처 단계는 설계 단계를 개략설계(Preliminary design)와 상세설계(Detail design)로 구분할 경우, 개략설계에 해당한다[6]. 그림에서 보듯이 요구분석과 설계 단계의 사이에 존재한다. 요구분석 단계의 산출물을 입력물로하여 소프트웨어 아키텍처를 작성한다. 이러한 소프트웨어 아키텍처는 여러 후보 아키텍처로 존재하거나 혹은 단일 아키텍처로만 존재 할 수도 있다. 그림2에서 보듯이 소프트웨어 아키텍처를 평가하여 그 결과를 다시 소프트웨어 아키텍처 작성에 반영한다. 단일 아키텍처일 경우에는 평가 결과를 바탕으로 수정, 보안 하여 후보 아키텍처로 존재 할 경우에는 각각의 후보 아키텍처를 모두 분석, 평가하여 가장 목표시스템의 품질 속성을 만족하는 아키텍처를 선정한다.

그림2의 리얼 프로토타이핑은 분석,평가에 영향을 주고 있다. 이것은 내장형 실시간 시스템의 어플리케이션이 자원과 밀접한 관련이 있고 또한 실시간성을 다루기 위해서는 실제적인 프로토타이핑을 통해 시스템의 동적인 특성을 평가하여 해당 아키텍처를 분석, 평가하는 것이 필요하다. 이를 통해서 목표시스템을 보다 정확하게 예측할 수 있으며 위험을 식별할 수 있다. 리얼프로토 타이핑은 요구 분석의 자료와 초기 아키텍처를 기반으로 하여 실제 구동하는 산출물이다. 프로토타입 프로세스 모델은 고객의 요구사항을 더욱 명확하게 하기 위한 목적으로 어플리케이션의 내부 구조는 고려하지 않고 사용자 인터페이스에 중점적인 프로토타이핑을 만든다. 때문에 성능이라든지 상호운용의 문제 등 품질에 관한 것은 고려하지 않는다. 하지만 본 논문에서의 프로토타이핑은 나선형 모델에서 말하는 프로토타이핑에 가깝다. 나선형 모델은 순차적 모델과 프로토타입 모델의 장점을 수용하고 위험 분석을 추가한 진화적 모델이다. 때문에 나선형 모델에서의 프로토타이핑은 시간 제약 사항이나 요구되는 성능을 측정하기위해 프로토타입을 개발하여 테스트 한다.[6] 따라서 본 논문에서 제안하는 실 프로토타이핑은 내장형 실시간 시스템의 성능을 측정하기 위해서 필수적이다. 실 프로토타이핑을 통해 스케쥴링, 우선순위 문제 등을 살펴 볼 수 있다. 또한 소프트웨어 아키텍처 상에서 공유메모리와 같은 구조가 많으면 프로세스간의 경합이 발생할 수 있는 확률이 높아진다. 여러 실시간 분석 기법을 실 프로토타이핑에 적절히 적용할 수 있다.

소프트웨어 아키텍처 분석 및 평가단계는 3.2절에서 상세히 다루도록 한다.

### 3.2 소프트웨어 아키텍처 평가 단계의 세부 활동

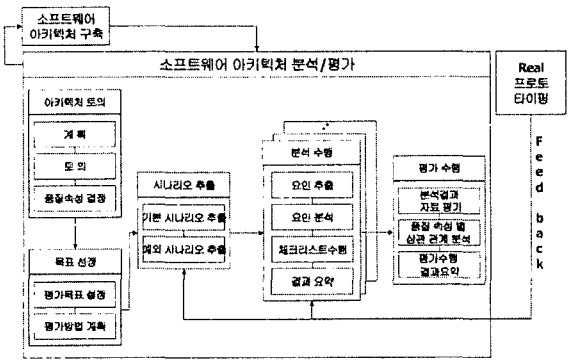


그림 3 세부 활동

그림 3은 그림2에서 소프트웨어 아키텍처 분석/평가 단계를 좀 더 세분화한 것이다. 이 단계를 아키텍처 토의, 목표 선정, 시나리오추출, 분석 수행, 평가 수행의 5개의 활동으로 구성한다.

첫째, 아키텍처 토의는 계획, 토의, 품질속성 결정 작업으로 구성한다. 소프트웨어 아키텍처 구축 단계에서 구축한 소프트웨어 아키텍처를 평가팀과 이해당사자들이 모여 토의를 한다. 이것의 목적은 목표시스템에 대한 이해 당사자들이 관심 있어 하는 품질속성을 서로 공유하며, 소프트웨어 아키텍처 구축팀은 구축한 아키텍처를 프리젠테이션하여 해당 소프트웨어 아키텍처의 이해를 높인다.

둘째, 목표 선정 활동은 평가목표 설정과 평가방법 계획 작업으로 구성한다. 아키텍처 토의 활동에서 추출한 품질속성 별 평가 목표를 설정한다. 또한 평가 방법을 계획한다. 예를 들어 성능을 평가하기 위하여 성능을 정의하고 목표 시스템에서 기대하는 성능을 정리한다. 분석/평가를 위해서 어떤 기법을 사용할 것인지를 결정하며 실 프로토타이핑을 통해 어떤 부분을 중점적으로 분석할 것인지를 결정한다.

셋째, 시나리오 추출 활동은 각 품질속성에 맞는 시나리오를 추출하는 활동이다. 시나리오는 기본 시나리오와 예외 시나리오로 나누어 추출한다. 시나리오는 요구사항명세서로부터 추출되거나 도메인 전문가에 의한 분석을 통해서 추출한다.

넷째, 분석수행 활동은 추출한 시나리오를 기반으로 하여 분석을 수행한다. 분석수행은 후보 아키텍처가 있다면 후보 아키텍처별로 수행한다. 먼저 각 품질속성에 영향을 주는 요소(factor)들을 찾아낸다. 각각의 품질 속성에 영향을 주는 요소를 정확히 식별해 내야 평가 결과가 타당할 수 있다. 요소를 식별 후 각각의 요소를 분석한다. 분석을 위해서는 질문기법(Questioning Techniques)와 측정기법(Measuring Techniques)를 혼합하여 적용한다. 이를 위해 각각의 식별 요소를 시나리오에 기반하여 적용하고 분석한 후 작성 한 체크리스트를 수행한다. 또한 내장형 실시간 시스템의 특성을 분석 작업에 반영하기 위하여 실 프로토타이핑(real prototyping)을 적용한다. 특정 컴포넌트간의 자원공유시 발생할 수 있는 우선 순위 문제, 성능 문제 등은 하드웨어와 분리하여 설명하기에는 부족한 부분이 많다. 레고와 같은 프로토타이핑 도구를 사용하여 실제와 비슷하게 모형화하여 적용하여 보는 것이 필요하다. 실 프로토타이핑은 모든 기능을 다 구현하여 사용하기 보다는 시나리오상의 제어 흐름속에서 나타나는 경우를 구현하는 방법이 더 효과적이다. 또한 실시간 분석 기법들 중의 하나인 RMA(Rate Monotonic Analysis)와 같은 스케쥴링 기법을 적용할 수 있다. 이러한 것은 시간적인 비용이 많이 들지만 내장형 실시간 시스템의 엄격한 제약사항들을 만족 시키기 위해서는 필요한 부분이다. 소프트웨어 아키텍처 평가에 실 프로토타이핑의 수행 결과를 반영하는 것은 보다 나은 구조를 구성하는데 도움이 된다. 이렇게 수행한 분석결과를 최종적으로 요약한다.

다섯째, 평가수행 활동은 각각의 품질속성별 수행한 분석 결과를 입력으로 받아서 이를 평가하고 품질 속성별 상관 관계, 후보 아키텍처들 간의 비교, 해석을 거쳐 최종적으로 아키텍처 평가를 내린다.

## 4. 결론 및 향후 연구 방향

본 논문은 내장형 실시간 시스템에 특화한 소프트웨어 아키텍처 분석 평가 절차를 제시하였다. 내장형 실시간 시스템에 소프트웨어 아키텍처의 적용과 분석/평가가 필요한 이유는 기존의 펌웨어 수준의 내장형 실시간 시스템이 아니라 여러 제약사항을 충족시켜야 할 뿐만 아니라 시스템의 복잡성이 높아져서 신뢰성 있는 시스템을 구축하기가 더욱 어려워졌기 때문이다. 소프트웨어 아키텍처 평가를 통해 충분히 품질속성을 고려하고 위험을 식별하는 것은 다른 어떤 도메인보다 내장형 실시간 시스템에 필요하다. 이를 위해 본 논문에서는 실 프로토타이핑을(real prototyping) 소프트웨어 아키텍처 분석/평가 단계에 반영하였고 전체적인 절차를 제시 하였다. 또한 반복적이며 점진적으로 소프트웨어 아키텍처를 정제해 나갈 수 있는 구조로 잠재적인 위험을 줄이고 품질속성을 만족시키는 소프트웨어 아키텍처를 구현하도록 하였다.

향후 연구 방향으로는 이러한 절차를 사용하여 실제 소프트웨어 아키텍처를 분석/평가하는 사례연구와 내장형 실시간 시스템에 특히 중요한 품질요소들을 고려하여 각각의 품질 요소에 영향을 주는 요소들을 찾아내고 이들을 분석하는 기법들을 연구해야 할 것이다.

## 5. 참고문헌

[1] Bruce Powel Douglass, "Real Time UML Third Edition", Addison Wesley,2004

[2] Bruce Powel Douglass, "Doing Hard Time: Developing Real-Time Systems with UML, Objects, Frameworks, and Patterns",1999

[3] Wolfgang A. Halang, Krzystof M. Sacha, "Real-Time Systems", World Scientific, 1992.

[4] Len Bass, Paul Clements, Rick Kazman "Software Architecture in Practice", Addison-wesley,2003.

[5] Lilianan Dobrica, Eila Niemela, "A Survey on Software Architecture Analysis Methods", IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL 28, NO.7, JULY 2002.

[6] Paul Clements, Felix Bachmann, Len Bass, David Garlan, James Ivers, Reed Little, Robert Nord, Judith Stafford, "Documenting Software Architectures Views and Beyond" Addison-wesley, 2003.

[7] Marco Castaldi, Paola Inverardi, Sharareh Afsharian, "A Case Study in Performance, Modifiability and Extensibility Analysis of a Telecommunication System Software Architecture", IEEE Int'l Symp. 2002.

[8] 강순주, "임베디드 소프트웨어 교육 워크샵", ETRI,2003

[9] 김수동, "실무자를 위한 소프트웨어 공학", 에드텍,1999