임베디드 소프트웨어의 Maintainability 향상을 위한 Architectural Perspective 정의

김종필*, 홍장의**
충북대학교 전자계산학과
e-mail:*kimjp@selab.chungbuk.ac.kr, **jehong@chungbuk.ac.kr

Architectural Perspective to Improve Maintainability of Embedded software

Jong-Phil Kim*, Jang-Eui Hong**
Dept of Computer Science, Chungbuk National University

요 약

Architectural Perspective라 함은 요구되는 품질의 특성이 시스템에 잘 반영되도록 하기 위한 활동, 전략 및 가이드라인을 정의한 것으로써, 다양한 관점에 바라보는 시스템에 대한 뷰(View)에 품질 속성을 반영하도록 하기 위한 것이다. 본 연구에서는 임베디드 소프트웨어의 유지보수 측면에서의 품질 향상을 위한 Maintainability Perspective를 제시한다. 제시하고자 하는 Perspective는 고장(failure)으로 인한 결함의 탐지성에 주안점을 두었으며, 이에 대한 아키텍처 패턴을 정의하였다. 정의된 패턴은 소프트웨어 아키텍처 개발에 적용하도록 함으로써, 임베디드 소프트웨어의 유지보수에 대한 품질 속성을 향상시키도록 하였다.

1. 서론

소프트웨어 아키텍처는 대상 시스템의 설계를 위한 매우 중요한 활동으로 시스템의 구조와 시스템을 구성하는 요소 및 이들 간의 관계, 그리고 외부에서 가시적인 행동 특성을 표현하는 모델이다[7].

소프트웨어의 개발은 단지 대상 시스템에 대한 한 측면의 이해만으로 개발될 수 있는 것이 아니라 정적인 측면과 동적인 측면, 또는 사용자 측면과 개발자 측면 등의 다양한 관점에서 이해되어야 한다. 소프트웨어 아키텍처도 이러한 다양한 관점(view)에서 대상 시스템을 바라보게 되며, 이러한 뷰들은 적절한 심벌 및 표현 언어를 이용하여 기술하게 된다[9].

이러한 뷰들이 갖는 대상 소프트웨어에 대한 기능적 표현이외에 시스템이 만족되어야 하는 품질 요소 예를 들면, 성능, 보안, 가용성, 접근성 등의 비 기능적인 품질 요소들이 반영되어야 한다. 이러한 품질 요소의 반영을 위해 제안된 것이 Architectural Perspective이며, 이들은 아키텍처 모델링에 있어서 Quality에 대한 속성을 충분히 반영하도록 가이드하고 있다[7,10].

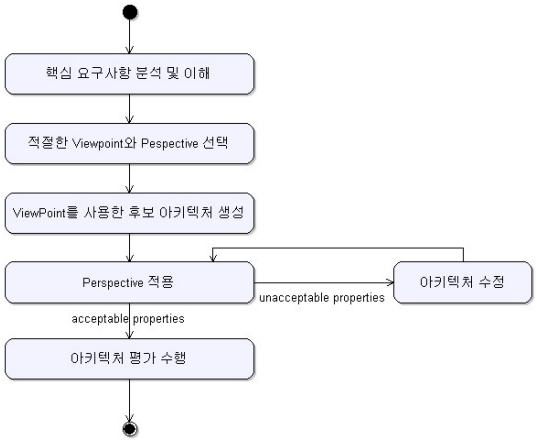
그러나 임베디드 소프트웨어의 경우는 품질 요소에 대한 특성이 일반적인 정보처리 시스템과는 다르게 정의될 수 있다. 본 연구에서는 임베디드 소프트웨어의 유지보수성(Maintainability)에 대한 Perspective를 정의하고, 이들의 활용을 통한 소프트웨어 품질의 개선 방안에 대하여 제시한다. 임베디드 소프트웨어는 일반적으로 하드웨어 플랫폼에 완전히 내장되기 때문에 유지보수가 용이하지 못하며, 특히 국방 시스템이나 항공기 탑재 소프트웨어의 경우는 소프트웨어의 오동작으로 인하여 많은 문제를 유발할 수 있기 때문에 운영 단계에서의 유지보수가 중요하다[6].

2. Architectural Perspective

2.1 Architectural Perspective의 정의

Architectural Perspective는 여러 가지의 아키텍처 뷰를 통해 고려해야 하는 품질 속성들을 시스템에 충분히 반영할 수 있도록 보장하기 위한 활동, 가이드라인, 전략들의 집합으로 정의한다. 이러한 Perspective를 소프트웨어 개발 과정에 적용함으로써 소프트웨어 아키텍처에 비 기능적 요구사항들을 보다 쉽고 효율적으로 반영할 수 있다. Perspective의 적용 절차는 (그림 1)과 같이 표현된다. 즉, 시스템에 대한 비 기능적인 핵심요구사항을 이해하고, 이러한 요구사항을 만족시킬 수 있는 Perspective를 선택한다. 선택된 Perspective는 기능적 요구사항을 기준으로 개발된 소프트웨어 아키텍처에 적용되며, 이의 결과로 대상 시스템의 품질 속성을 향상시키

는 융통성 있는 소프트웨어 아키텍처가 개발된다[7].



(그림 1) Perspective 적용 절차

2.2 Architectural Perspective의 구성

Architectural Perspective는 다양한 품질 속성들을 정의함에 있어서 체계적이고 적용의 용이성을 위하여 표준화된 정의 구조를 사용한다. 이 구조에 포함되어 있는 정보들은 다음과 같다[7,10].

- 적용성: Perspective가 적용되었을 때 그 것이 어느 뷰에 영향을 미치는 지를 설명한다.
- 관심사: Perspective가 초점을 맞추는 품질 속성에 대해 정의한다.
- 활동: 특정 뷰에 Perspective를 적용하기 위한 단계들을 설명한다.
- 아키텍처 전략: 특정 품질 속성을 달성할 수 있도록 도와주기 위해 사용할 수 있는 증명된 방식이다. 각 Perspective는 품질 속성을 달성하기 위한 중요한 전략들을 식별하고 기술한다.
- 문제와 함정: 일어날 수 있는 문제들을 설명하고 그 문제들을 어떻게 인지하고 피할 수 있는 지에 대한 가이드를 제공한다.
- 체크리스트: 중요한 관심사, 적절한 전략, 공통적인 문제들을 예외 없이 식별할 수 있도록 질의목록을 제공한다.

3. 임베디드 소프트웨어의 유지보수

3.1 임베디드 소프트웨어의 유지보수 활동

유지보수성은 소프트웨어 시스템이나 컴포넌트들이 오류 수정, 성능 개선, 변화된 환경 적응 등에 대하여 얼마나 용이한 가를 나타내는 정도를 말한다. 임베디드 소프트웨어의 경우 유지보수 활동은 다음과 같은 4가지의 형태로 구분할 수 있다[8].

- Corrective maintenance
- Adaptive maintenance
- Perfective maintenance
- Preventive maintenance

위와 같은 유지보수 활동 중에서 Corrective maintenance는 결함으로 인하여 고장(failure)이 발생한 경우에 이루어지는 활동으로써, 결함의 원인과 위치를 파악하여 수정하게 된다. 임베디드 소프트웨어의 경우 이와 같은 활동을 수행하기 위한 방법은 크게 두 가지 형태로 구분할 수 있다.

첫 번째는 호스트 머신 상에서 테스트 데이터를 통해 소프트웨어를 시험하는 것이다. 호스트 머신 상에서 테스트를 통과한 소프트웨어는 하드웨어 플랫폼에 탑재된다. 두 번째 방법은 하드웨어 플랫폼에 내장된 상태에서 결함을 찾아내고 결함을 포함하는 모듈을 대체하는 방법이다. 전자의 방법은 비교적 임베디드 소프트웨어가 기능적으로 단순하고 작은 규모의 경우에 적합하나, 시스템의 기능이 복잡해지고 다양한 하드웨어 인터페이스를 포함하는 경우에는 결함을 찾는 방법이 용이하지 않다. 따라서 항공기 및 위성 제어 소프트웨어, 고성능 통신 단말기와 같은 경우는 후자의 방법을 통해 결함을 찾아내는 것이 유용하다고 할 수 있다[3,4].

3.2 임베디드 소프트웨어의 유지보수 품질 속성

소프트웨어의 유지보수성에 대한 품질 속성은 모듈성(Modularity), 가독성(readability), 복잡도(Complexity), 회복성(Recoverability), 표준화(Standardization) 등이 있으나[5], 본 연구에서는 이러한 품질 특성이외에 고려될 수 있는 결함의 탐지성(Detectability)에 대하여 연구의 주안점을 두었다. 결함의 탐지성이라 함은 임베디드 소프트웨어에 내재된 Failure의 원인을 어떻게 하면 잘 찾아낼 수 있는 가에 관한 문제이다.

복잡한 대형의 임베디드 시스템의 경우에 있어서 결함의 탐지는 단순한 테스트 데이터를 이용하여 호스트 머신 상에서 수행될 수 있는 것은 아니다. 따라서 소프트웨어가 내장되는 하드웨어 플랫폼의 특성 및 환경을 고려하는 방법으로 결함의 탐지가 이루어져야 보다 안전한 유지보수 작업이 이루어질 수 있다[6]. 다시 말해서 내장된 임베디드 소프트웨어를 동작시키고, 이 과정에서 발생하는 이벤트 및 상태 변화를 모니터링 하여 결함을 찾아내는 것이다. 이를 위해서는 내장된 소프트웨어가 어떻게 동작하는 지를 모니터링하기 위한 로그(log)를 확보해야 하며, 소프트웨어적인 측면에서 이를 지원할 수 있도록 개발되어야 한다.

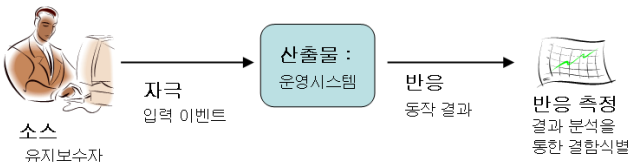
본 연구에서는 이러한 대형의 임베디드 시스템에서 결함의 탐지를 위한 전략을 중심으로 Architectural Perspective를 정의한다. 특히 앞의 2.2절에서 정의한 Perspective 구성요소 중에서 관심사, 아키텍처 전략, 적용성, 체크리스트 등에 대하여 정의한다.

4. Detectability Perspective

임베디드 시스템의 유지보수성 향상을 위한 결함 탐지성에 대한 Architectural Perspective에 대한 정의는 다음과 같다.

4.1 관심사(Concerns)

본 연구에서 고려하는 Architectural Perspective의 관심사는 탐지성이라는 품질 속성이지만 이에 대한 체계적이고 상세한 정의가 필요하다. 이를 위하여 품질 속성을 표현하는 수단인 "품질속성 시나리오(Quality Attribute Scenario)"을 이용하여 관심사를 정의한다[1]. (그림 2)는 시스템이 배포된 이후 고장이 발생했을 경우 유지보수자에 의해 결점을 탐지할 수 있어야한다는 시나리오 예제를 보여주고 있다.

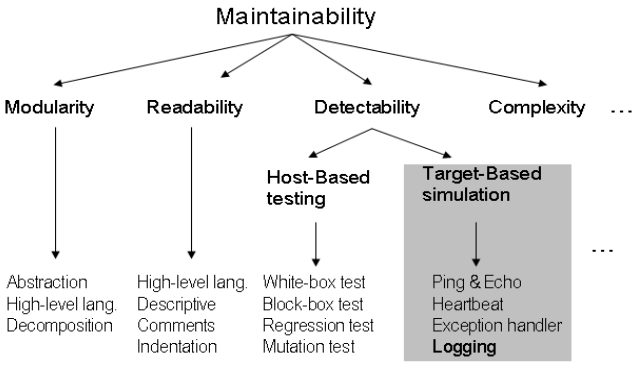


(그림 2) 탐지성에 대한 품질속성 시나리오 예제

(그림 2)에서 보여주는 것처럼 결함탐지에 대한 품질속성 시나리오는 유지보수자에 의해 입력되는 자극(입력 이벤트)에 의해 시스템이 동작하고 동작 결과를 기반으로 결함을 식별하는 활동이다.

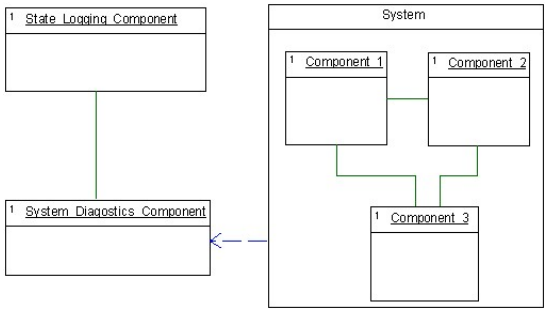
4.2 아키텍처 전략

임베디드 시스템에 대한 결함의 탐지성을 향상시키기 위하여 어떠한 기술적인 전략(tactic)을 선택할 것이며, 이를 지원하는 설계 패턴이나 아키텍처 패턴이 무엇인가를 정의한다. 결함 탐지성에 대한 아키텍처 전략을 정의하면 다음 (그림 3)과 같다.



(그림 3) 결함 탐지성에 대한 아키텍처 전략

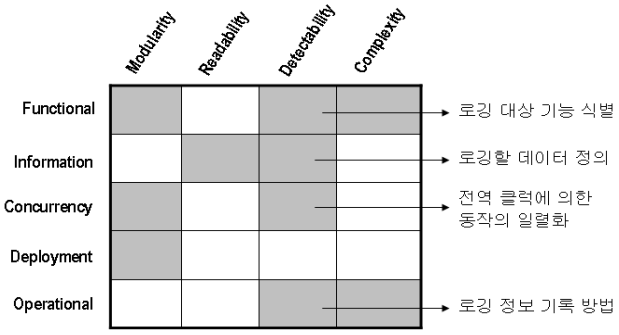
(그림 3)에서 정의한 아키텍처 전략 중에서 본 연구의 관심은 "Logging" 전략이다. 로깅(logging)은 대규모의 임베디드 시스템에서 기능 수행과 함께 매우 손쉽게 수행할 수 있는 전략으로써, 시스템 동작이나 성능에 크게 문제를 유발하지 않으며, 또한 개발 시스템의 아키텍처에 포함시킬 수 있을 정도로 부담이 적은 전략이다. 이와 같은 전략에 대하여 정의할 수 있는 아키텍처 패턴을 정의하면 (그림 4)와 같다.



(그림 4) logging 전략을 위한 아키텍처 패턴

4.3 Perspective의 적용

앞서 정의된 전략들을 임베디드 시스템 개발에 적용하기 위해서는 어떠한 아키텍처 뷰에 영향을 주는 가가 결정되어야 한다. 일반적으로 아키텍처에 대한 뷰는 Functional view, Information view, Concurrency view, Deployment view, Operational view 등으로 구분된다. 이러한 뷰들과 아키텍처 전략들 간의 관계를 정의함으로써, Perspective에 대한 적용성을 정의할 수 있다. (그림 5)는 이 관계를 나타내고 있다.



(그림 5) 각 View와 Detectability와의 관계

(그림 5)에서처럼 결함 탐지성은 시스템의 Functional, Information, Concurrency, Operational view 등에 적용될 수 있으며, 이러한 뷰들의 아키텍처링에서 탐지성을 제공하는 요소들이 반영되어야 할 것이다.

Architectural Perspective를 정의하기 위한 요소로써의 체크리스트는 적용된 예제 모델이 충분히 탐지성에 대한 품질 속성을 반영하였는가를 결정하기 위한 용도로 사용될 것이다.

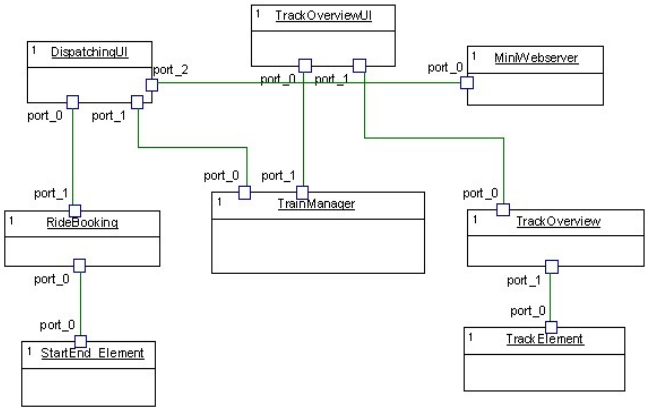
5. 예제 시스템의 적용

5.1 예제 시스템 정의

본 논문에서 적용하고자 하는 시스템은 자동화된 기

차역 운영 시스템이다. 시스템은 크게 각 레일과 기차를 제어하는 컨트롤 부분과 승객들에게 온라인 서비스를 제공할 수 있는 웹서버로 나뉜다. 트랙 상황을 감시하고 온라인으로 예약 서비스를 제공한다. 또한, 여러 외부 하드웨어 인터페이스와의 상호작용을 통해 시스템이 운영된다.

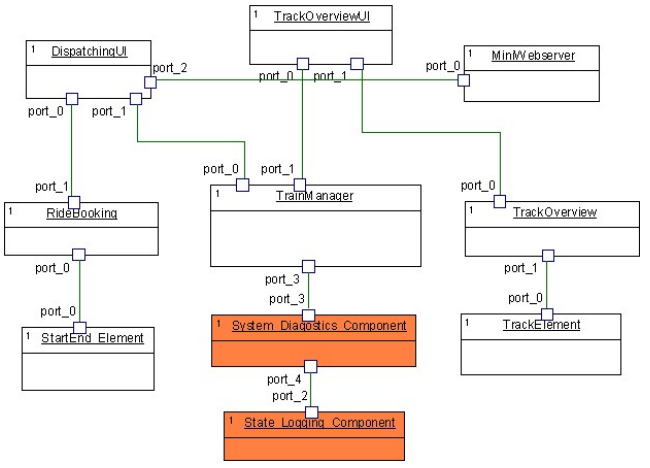
자동화된 기차역 운영 시스템의 상위수준 아키텍처는 (그림 6)과 같다. 외부 하드웨어들을 관리하는 TrainManager 컴포넌트, UI 컴포넌트 등으로 구성되어 있는 것을 볼 수 있다.



(그림 6) 기차역 운영 예제 시스템의 상위 수준 아키텍처

5.2 예제 시스템의 Perspective 적용

앞의 4장에서 정의된 Detectability Perspective에 대하여 Functional View에 적용된 형태는 (그림 7)과 같다. 시스템에 Logging 패턴을 적용하여 추가된 2개의 컴포넌트를 볼 수 있다. Diagnostics 컴포넌트가 시스템 상태를 모니터링하여 로깅 컴포넌트로 전송을 하면 유지보수자는 로깅 정보 분석을 통해 결함 탐지를 할 수 있다.



(그림 7) Detectability Perspective가 적용된 아키텍처

5.3 체크리스트를 통한 아키텍처 점검

체크 리스트는 전략의 적용이 올바르게 적용되어 아키텍처링이 이루어졌는가를 확인하기 위한 것이다. 대표적인 체크 항목들은 다음과 같으며, 이들은 일반적인 아키텍처 평가 방법인 ATAM[2] 방법과 유사하기 때문에 구체적인 사항은 생략하기로 한다.

- 시스템 구성을 충분히 고려하여 적합한 전략을 적용하였는가?
- Perspective를 적용함으로써 Ripple Effect가 일어났는가?
- 적용된 패턴이 표준에 맞게 정의 되었는가?
- 로그 분석을 통해 결함 탐지성이 향상 되었는가?

6. 결론 및 향후 연구

임베디드 시스템이 대형화 되고, 복잡해지고 있다. 특히 유비쿼터스 환경으로의 진화는 다양한 임베디드 소프트웨어 기술을 컨버전스하고 있으며, 이로 인하여 고장으로 인한 시스템 결함을 탐지하는 것은 매우 중요한 활동이라고 할 수 있다.

본 연구에서는 이러한 중요성에 따라 결함 탐지성에 대한 Architectural Perspective를 정의하고, 이를 예제 시스템에 적용하여 아키텍처링 하였다. 이렇게 함으로써, 복잡한 임베디드 소프트웨어 결함 탐지가 가능해지는 소프트웨어 개발이 가능해지며, 결함 탐지를 위한 노력이 감소될 것이다.

참고문헌

[1] Muhammad Ali Babar, "Scenarios, Quality Attributes, and Patterns: Capturing and Using their Synergistic Relationships for Product Line Architectures", APSEC, pp. 574-578, 2004

[2] Paul Clements, et al., "Evaluating Software Architectures", Addison Wesley, 2002

[3] A. Dasgupta, et al., "Perspectives to Understand Risks in the Electronic Industry", IEEE, Vol. 20, No4, 1997

[4] Bas Graaf, "Maintainability through Architecture Development", LNCS, Vol. 3047, pp. 206-211, 2004

[5] K. Hashim, et al., "A Software Maintainability attributes model", Malaysian Journal of Computer Science, Vol. 9, No. 2, pp. 92-97, December 1996

[6] Christer Norstrom, et al, "Increasing maintainability in complex industrial real-time systems by employing a non-intrusive method", ASTEC, 2003

[7] Nick Rozanski and Eoin Woods, "Software Systems Architecture", Addison Wesley, 2005

[8] E. Burton Swanson, "IS 'Maintainability':Should It Reduce the Maintenance Effort?", The DATA BASE for Advances in Information Systems, Vol. 30, No.1, 1999

[9] E. Woods. "Experiences Using Viewpoint for Information Systems Architecture: An Industrial Experience Report.", LNCS, Vol. 3047, pp. 182-193, 2004

[10] Eoin Woods, et al., "Using Architectural Perspectives", WICSA '05, Volume 00, pp. 25-35, 2005