사용사례를 이용한 내장형 소프트웨어의 비기능 요구사항 추출 방안

서광익° 최은만
동국대학교 컴퓨터멀티미디어공학과
{bradseo°, emchoi}@dgu.ac.kr

Non-Functional Requirements Analysis For Embedded Software By Use-Case Diagram

Kwang-Ik SEo°, Eunman Choi
Dept. of Computer Engineering Dongguk University

요 약

최근 정보통신 기술이 빠르게 발전하면서 생활 환경 안에서 내장형 소프트웨어를 탑재한 독립적인 장치들을 많이 볼 수 있다. 이러한 내장형 장치들은 하드웨어의 특성으로 인해 비기능적인 요구사항이 중요하다. 본 논문에서는 내장형 시스템의 비기능적 요구사항을 기능적 요구사항 중심으로 추출하는 방법을 제안한다.

1. 서 론

정보통신 기술의 발전으로 인해 생활 주변에 있는 많은 장치들이 소프트웨어를 내장하고 있는 것을 볼 수 있다. 현재 내장형 시스템은 일반 PC에 비해 100배 정도 더 큰 시장을 형성하고 있으며[1], 10년 안에는 그 규모가 기하급수적으로 증가할 것으로 전망한다[2]. 이러한 내장형 시스템의 기능과 요구사항들은 다양한 사용자와 제조자 또는 시장 상황 등에 의해 영향을 받는다. 따라서 시스템이 제공해야 하는 기능적 요구사항과 비기능적 요구사항들을 명확하게 표현하는 것이 필요하다. 물리적인 하드웨어와 밀접한 관계가 있는 내장형 소프트웨어의 특성상 비기능적 요구사항은 매우 중요하다. 그럼에도 불구하고 내장형 소프트웨어에 대한 비기능적 요구사항을 표현할 수 있는 지침서나 표본은 현재 부족한 상태이다[2]. 따라서 사람들마다 비기능적인 요구사항의 표현 방법이 상이하게 되어 명세서간의 관계성과 재사용성이 낮아지는 문제점 그리고 설계자와 개발자 간의 의사소통에 대한 어려움이 발생할 수 있다. 이러한 문제점을 해결하기 위해 본 논문에서는 비기능적 요구사항의 명세가 필요한 내장형 소프트웨어의 특징을 찾아보고, 비기능적 요구사항을 추출하는 방법에 대해 연구하였다.

2장에서는 내장형 소프트웨어가 지니는 비기능적 특성을 서술했고, 3장에서는 비기능 요구사항을 추출하는 방법에 대한 기존 연구 문헌들을 살펴보고, 4장에서는 본 논문에서 비기능 요구사항을 명세하기 위한 접근 방법의 관점에 대해 기술한다. 그리고 5장에서는 기능적 관점의 사용사례를 중심으로 비기능적인 사용사례를 정의하고, 이를 적용하여 소프트웨어의 클래스 개념도를 작성하는 절차와 방법을 서술하였다. 그리고 6장에서 본 논문에 대한 결론을 정리하였다.

2. 내장형 소프트웨어의 비기능적 특성

내장형 소프트웨어의 비기능적 특성 중에서는 시간의 제약성과 동기화, 영속성(liveness), 인터페이스의 적절성, 실시간, 이기종과의 통신능력, 반응 속도, 신뢰성 등 다양한 비기능적 특성들이 있다[3][4]. 하지만 내장형 제품은 시장 출하시간이 중요하므로 신속한 제품 생산 및 시장 경쟁 체제 또한 신중하게 고려해야 한다. 만약 특정 제품에 대해 품질 특성들을 모두 살피고 이들 간의 우선순위를 결정한 후 이를 바탕으로 비기능 요소를 찾아 시험한다면 많은 시간과 자원이 필요할 것이다. 따라서 신속한 시장 출하 시간을 만족하면서 동시에 제품의 품질을 높이기 위한 비기능적 요구사항을 추출하는데 많은 시간을 절약할 수 있는 방안을 모색해야 한다. 이에 본 논문에서는 기능 위주의 요구사항을 중심으로 비기능적인 내장형 소프트웨어의 요구사항을 추출하는 방안을 제안한다.

3. 비기능 요구사항 추출 방법

3.1 품질 기반 계층적 비기능 요구사항 표현 방법

L. Chung은 절차 중심적인 방법으로 제품의 품질에 기반을 둔 비기능적인 요구사항을 추출하고 표현하는 방법을 제안했다[7]. 비기능적 요소를 표현하기 위해서 목적(goals) 집합과 목적 간의 관계를 설정하는 링크(links) 집합으로 분류를 하고, 목적은 정점으로 관계는 간선으로 구성되는 트리 형태로 표현한다. 그리고 이들을 통해

비기능적 요소를 추출하기 위한 프레임워크를 제안했다.

3.2 UML 을 이용한 비기능 요구사항 추출

대부분 시스템을 개발 할 때 소프트웨어의 기능적인 관점과 비기능적인 관점을 분리하였다. 이러한 방법은 시장 출하 시간을 늦출뿐더러, 기능 요구사항과 비기능 요구사항의 연관성이 저하되어 소프트웨어 개발의 복잡도를 높이는 원인이 된다. 하지만 Cysneiros 와 Leite 는 비기능적 요소를 하나의 시작점에서 기능적 관점과 연계하여 찾아 내는 방법을 제안하였다[8].

3.3 기능 요구사항과 아키텍쳐의 관계를 고려한 비기능 요구사항 추출

Paech 와 Dutoit 그리고 Kerkow 와 Knethen 은 비기능적인 요소를 추출할 때 기능적 요소와 이키텍쳐 구조의 밀접한 연관성을 고려한다. 검증된 기능적 요소를 통해 신뢰도 높은 비기능적 요소를 추출하는 방법을 제안하고 있다[5]. 그림 1 은 이러한 관계를 보여주고 있다.

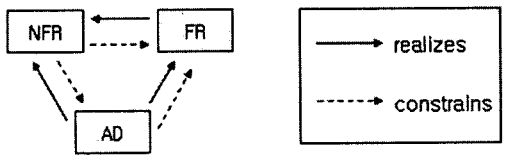

그림 1 기능, 비기능적 요소와 아키텍쳐 간의 관계

3.4 기능사항과 비기능 요구사항 간의 균형

기능적 요구사항을 분석하기 위해 사용사례를 이용한 방법이 활발하게 진행되고 있다. 하지만 사용사례를 이용한 요구분석 방법은 시스템 개발 초기 단계에서 성능이나 위험도, 또는 심각도 등을 충분하게 분석할 수 없다는 단점이 있다[9]. 비기능적인 요소를 발견할 때 마다 기능적인 요구사항 사이에서 균형을 맞춰, 개발 초기에서부터 하나의 사용사례에 대한 객체에 비기능적인 사용사례를 확장하여 매핑시키는 방법을 제안하고 있다[9]. 그림 2 는 기능 관점에서 get()메소드와 관련 있는 비기능 사용사례의 매핑을 보여주고 있는 그래프이다.

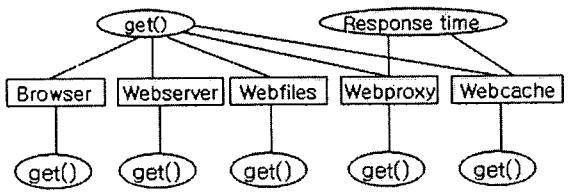

그림 2 비기능 요구사항을 나타내는 그래프

4. 비기능 요구사항을 추출하기 위한 접근방법

일반적으로 비기능 요구사항을 추출하기 위한 접근 방법은 대부분 하향식 접근 방법이다. 본 논문에서도 비기능 요구사항을 시스템 수준에서 정의하고, 중요한 비기능에 대해서는 더욱 자세히 정의한다. 즉 전체적 품질 사항을 고려한 후 사용자의 요구사항을 모형화 하는 단계에서 중요한 기능 또는 제약 조건에 관해 구체적으로 명세하기 위한 비기능 사용사례를 이용한다.

5. 내장형 소프트웨어의 비기능적 요구사항 추출 절차

5.1 내장형 시스템 품질 특성의 우선순위 결정

실시간 처리와 동기화, 영속성(liveness), 인터페이스의 적절성, 실시간, 이기종과의 통신능력, 반응 속도, 높은 신뢰성과 안정성 등 내장형 시스템의 품질 특성들은 다양하다. 하지만 제한된 시간과 자원으로는 언급된 모든 특성들을 시험 할 수는 없다. 또한 개발된 제품은 그들만의 고유한 특성이 있기 때문에 각 제품의 특성에 맞도록 제품 품질의 중요도와 우선순위를 결정해야 한다.

5.2 기능 관점의 사용사례 추출

구현하려는 시스템의 기능과 사용자의 요구사항을 추출하기 위해 사용사례를 이용한다. 본 논문에서 적용한 실험 대상 시스템은 현재 그 시장이 계속적으로 확산되고 있는 셋톱박스를 사용했다. 그림 3 은 사용자가 셋톱박스를 사용할 때 가장 기본이 되는 기능들로써 위성을 검색하여 등록하고 위성으로부터 전달되는 채널을 검색하거나 검색된 채널을 조정하는 기능, 그리고 채널에 대한 녹화와 볼륨을 조절하는 기본적인 기능들을 보이고 있다.

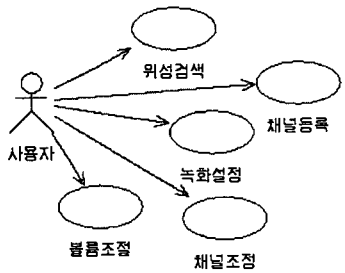

그림 3 셋톱박스의 사용사례

5.3 기능 사용사례로부터 품질 특성을 고려한 비기능 사용사례 추출

5.2 에서 추출된 기능 중심의 사용사례를 중심으로 비기능적인 사용사례를 추출한다. 사용사례에 대한 비기능 사용사례는 점선으로 된 타원형으로 나타낸다. 또한 비기능 품질 요소에 대한 의미를 나타내기 위해 스테레오타입을 적용한다.

그림 4 는 기능 사용사례와 관련하여 비기능 사용사례를 추출한 사례이다. 품질 요소 중 비기능적인 요소를 기능적 요소와 연관시켜 모델링 작업을 할 때 적용했다. 셋톱박스에서 수신 가능한 위성을 검색할 때 하나의 위성에 대한 검색 제한 시간 등을 설정할 수 있다. 또한 채널에

관해서도 위성으로부터 전달되는 채널 정보를 검색하거나 채널을 등록할 수 있는 자료구조의 크기 등에 대한 비기능적 특성을 명시할 필요가 있다. 그리고 녹화를 위한 메모리의 제한 용량이나 녹화된 정보를 관리하는 자료구조의 제한 또한 고려해야 하는 중요한 비기능적 특성이다.

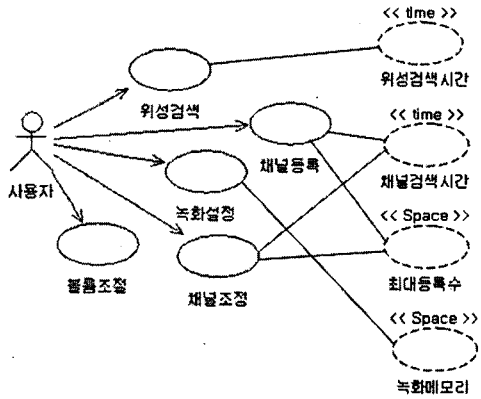

그림 4 기능/비기능 사용사례의 통합적 추출

그림 4 에서 기능 사용사례와 비기능 사용사례의 다중도는 0..n : 0..n 이 될 수 있다. 즉, 하나의 기능 사용사례는 연관 있는 비기능 사용사례가 없을 수도 있고 또는 다수의 비기능 사용사례와 연관성이 있을 수도 있다. 그 관계는 비기능 사용사례의 관점에서도 동일하다. 이러한 관계의 정의는 클래스도를 작성하는 단계 이전에 정의를 해야 한다. 만약 요구분석 단계에서 기능 요구사항과 비기능 요구사항 사이의 관계가 정의되어 있지 않다면 이를 이용한 클래스의 매핑은 더욱 복잡해진다. 왜냐하면 설계 단계에서 클래스도를 작성할 때 하나의 기능 사용사례는 하나 또는 하나 이상의 클래스와 관련이 있기 때문이다. 그 사용사례와 클래스와의 관계는 1..n : 1..n 이 된다.

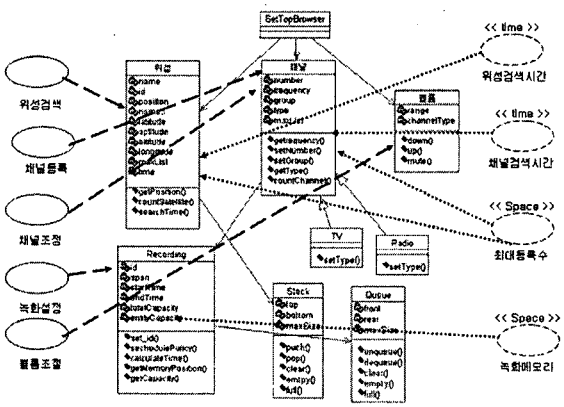

그림 5 기능/비기능 사용사례를 통한 클래스도

5.4 기능/비기능 사용사례로부터 개념도 추출

기능적 관점과 비기능적 관점에서 통합적으로 요구명세를 추출한다. 그리고 UP(Unified Process) 방법론의 관점에서 사용사례를 중심으로 기본적인 클래스의 개념도를 작성하게 된다. 5.3 에서 비기능적인 요구사항을 추출했기 때문에 비기능적인 요구사항이 적용된 균형있는 개념도 또는 기본 설계 단계에서의 개괄적인 클래스도를 작성할 수 있다. 그림 5 는 기능적 요소와 비기능적 요소가 적용된 클래스도이다.

6 결론 및 향후 과제

내장형 시스템의 소프트웨어는 하드웨어와 밀접한 관계가 있다. 따라서 시스템을 모델링 하는 동안에 기능적 요구사항뿐만 아니라 비기능적 요구사항도 깊은 관찰이 필요하다. 본 논문에서는 이러한 특징에 착안하여 기능 중심의 사용사례를 확장하여 비기능 요구사항도 표현하고 명세할 수 있는 비기능 사용사례를 제안했다. 그리고 기능 사용사례와 비기능 사용사례를 통합적으로 적용하여 클래스도를 작성하고 이들간의 관계를 간단히 보였다.

향후 연구과제로는 비기능적인 요구사항 분석 결과를 이용하여 시스템이 제공해야 하는 비기능들을 사용자 관점에서 검증하고 확인하기 위한 시험 사례를 추출할 수 있는 기법 연구가 필요 할 것이다.

참고문헌

[1] L.D.J. Eggermont, ed.: Embedded Systems Roadmap 2002, Technology Foundation, 2002, www.stw.nl/progress/Esroadmap/ESRversion1.pdf

[2] Bas Graaf, Marco Lormans, Hans Toetenel: Embedded Software Engineering: The State of the Practice, Software, IEEE, Volume:20, Issue:6, Nov.-Dec. 2003, 61-69

[3] Edward A. Lee, Embedded Software, 2001.

[4] Bard Broekman, Edwin Notenboom, " Testing Embedded Software" , Addison-Wesley, 2003

[5] P. Paech, A. H. Dutoit, D. Kerkow, A. von Knethen, " Functional requirements, non-functional requirements, and architecture should not be separated - A position paper." ,

[6] B. Nuseibeg, S. Easterbrook, " Requirements Engineering: A Roadmap" ,

[7] L. Chung. " Representing and Using Non-Functional Requirements: A Process-Oriented Approach" , Ph.D. Thesis, Dept. of Comp. Sci. of Toronto, June 1993

[8] L. M. Cysneiros, J.C.S.D.P. Leite, " Using UML to Reflect Non-Functional Requirements" ,

[9] Using trade-off analysis to uncover links between functional and non-functional requirements in use-case analysis