비기능 요구사항을 고려한 컴포넌트 추출 기법

황위용*, 강동수*, 조은애*, 송치양**, 백두권*
* 고려대학교 컴퓨터전파통신공학과
** 경북대학교 소프트웨어공학과
e-mail : *{wyhwang, 2008010372, eacho, baikdk}@korea.ac.kr, **cysong@knu.ac.kr

A Method of Component Extraction Considering NFRs

Wi-Yong Hwang*, Dong-Su Kang*, Eun-Ae Cho*, Chee-Yang Song**, Doo-Kwon Baik*
* Department of Computer and Radio Communications Engineering, Korea University
** Department of Software Engineering, Kyungpook National University

요 약

최근 시스템을 구축하는데 있어서 점점 더 많은 상용 컴포넌트가 쓰이고 있다. 컴포넌트에서 핵심 요소로 작용하는 요구사항은 기능 요구사항과 비기능 요구사항으로 나뉘며, 실질적인 컴포넌트의 재사용에 있어서 비기능적인 요소가 결정적인 기준으로 작용하고 있다. 비기능 요구사항은 해당 시스템이 지원해야 할 기능 요구사항의 제약사항 또는 품질 속성을 말하며, 소프트웨어의 품질 요구사항으로 반영된다. 결국 시스템의 품질을 보장하기 위해서는 시스템을 구성하는 컴포넌트가 가진 품질을 고려해야 한다. 따라서 본 논문에서는 시스템의 품질에 관여하는 비기능 요구사항을 분석 및 반영 하기 위해 품질 속성이나 제약사항과 같은 컴포넌트가 가져야 할 비기능 요구사항을 고려한 컴포넌트의 추출 기법을 제안한다. 비기능 요구사항의 분석은 UML 의 유스케이스에서 이루어지며 기능-비기능 요구사항의 영향관계를 고려하여 컴포넌트를 추출한다. 추출된 컴포넌트는 문서화를 통해 잘 기술된 제약사항 및 품질 요구사항에 대한 정보를 가지고 있기 때문에 보다 효과적인 컴포넌트를 이용한 개발을 가능케 한다.

1. 서론

오늘날 시장에 대한 신속한 대응을 위해 시스템 구축에 점점 더 많은 상용 컴포넌트가 쓰이고 있다. 컴포넌트가 모여 아키텍처를 이루고, 하나의 시스템을 이룬다. 이와 같은 컴포넌트를 활용할 경우 그 컴포넌트가 소프트웨어 아키텍처를 제약하기 때문에 설계 과정에도 변화가 생긴다. 일반적으로 컴포넌트를 채택하면 대체로 기능성은 확보하게 되지만, 그 과정에서 아키텍처 레벨에서의 다양한 비기능적인 고려사항이 발생하게 된다. 하지만 기존 컴포넌트는 품질과 관련한 고려가 미약하기 때문에 품질속성이나 제약사항을 정확히 파악하기 힘들며, 시스템 구축 시에 문제를 발생시킨다. 이는 결국 아키텍처의 구조가 품질 속성에 영향을 받으므로 아키텍처를 구성하는 컴포넌트 또한 품질적인 측면에서의 다양한 고려가 필요하다[1].

최근 시스템들은 그 규모가 크고 비교적 복잡하다. 이런 대형 시스템을 구축하기 위해서는 대상 시스템의 특성을 충분히 파악하여, 시스템의 기능적인 요구사항뿐만 아니라 시스템의 품질을 좌우하는 비기능적인 요구사항(NFRs)에 대해서도 신중하게 고려하여 시스템을 구축하여야 한다[5]. 요구사항 중에 기능 요구사항은 비기능 요구사항과 비교했을 때 보다 명시적인 분석이 용이하지만, 비기능 요구사항은 명시적으로 분석하고 측정하기가 힘들다는 문제점이 있다. 그러나 요구사항이 제대로 파악되지 못한다면 소프트웨어 개발 후반부로 갈수록 의도했던 품질을 위한 수정비용은 비약적으로 늘어나게 될 것이다[4]. 따라서 개발 단계부터 품질 관련한 비기능적 요구사항을 파악할 수 있는 방법이 요구되며, 기존 컴포넌트의 문제점인 품질적 측면의 비기능 요구사항을 보완하기 위한 연구가 필요하다.

본 논문에서는 시스템 개발의 전체 라이프 사이클(Software Life Cycle)에서 품질과 재사용 관점으로 비기능 요구사항을 분석하며, 분석 결과를 컴포넌트를 이용한 설계에 반영하기 위해 비기능 요구사항을 고려한 컴포넌트 추출 기법에 대한 연구를 진행하였다. 이를 통해 비기능 요구사항의 분석 및 관리의 효율성과 컴포넌트의 활용을 높이고자 한다.

* 이 연구에 참여한 연구자는 '2단계 BK21 사업' 의 지원을 받았음.

2. 관련 연구

2.1 UML 에서의 유스케이스 모델링

대표적으로 분석 및 설계 단계에서 널리 쓰이는 방

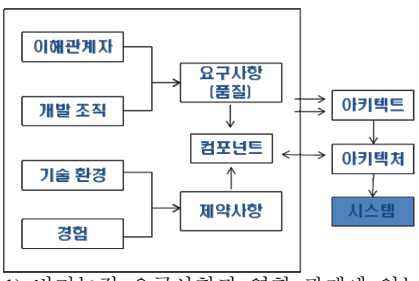
법 중 하나가 유스케이스를 이용한 분석방법 이다. 이것의 가장 큰 장점은 시스템을 분석하는데 있어서 복잡도 문제를 해결해주며, 전문가가 아닌 고객이나 사용자들의 이해가 쉽기 때문에 원활한 대화를 통하여 고객과 개발 사이의 분석 및 협의점 도출에 큰 기여를 하는 것이다. 하지만 유스케이스를 통한 요구사항 분석이 이루어질 때 기능적인 요소를 다루는 것은 수월하지만 비기능적인 요소인 품질 요구사항을 고려하는 데는 어려움이 따른다. 왜냐하면 UML 은 기능만을 다루는 표준이기 때문이다[7]. 하지만 품질이라는 것은 이해관계자의 관점에 따라 생각하는 의미가 다르기 때문에[6], 유스케이스를 이용한 분석단계에서 고려해야 이해관계자의 관점을 효율적으로 파악할 수 있다.

2.2 기능-비기능 요구사항의 통합

관련 논문 [2,8]에서는 유스케이스에서의 비기능 요구사항의 전파 규칙을 정의하고, 목표 지향의 접근(Goal-Oriented Analysis)을 통해 요구사항의 분화를 다루었다. 비기능 요구사항을 Softgoal 로 인식하여 비기능적인 측면을 고려한 요구사항의 통합을 가시적으로 나타내었다. Softgoal 은 근본적으로 요구사항을 나타내기 때문에 측정 및 평가가 필요하다. 하지만 이를 측정하는 것은 어려움이 따르며[3], 또한 비기능 요구사항을 그래프로 나타냈기 때문에 엄밀히 본다면 유스케이스에서의 가시적인 통합이라고 볼 수는 없다.

2.3 품질 요구사항과 컴포넌트

실제적으로 컴포넌트가 만들어질 때 개발 당시의 제약사항이나 품질과 관련된 속성들에 의해 영향을 받는다[6]. 결국 컴포넌트를 활용하여 시스템 구축을 할 때 아키텍처단계에서는 직접적으로 컴포넌트가 가지고 있는 제약사항 때문에 아키텍처가 영향을 받게 된다. 그림 1 에서는 비기능적 측면의 요구사항이 미치는 영향이 나타나있다. 컴포넌트와 아키텍처는 제품과 관련된 이해관계자들의 요구사항에 영향을 받는다. 여기에 추가로 개발 조직의 구성이나 목표, 기술 수준 및 경험 등도 영향을 주는 요소라 할 수 있다.



(그림 1) 비기능적 요구사항과 영향 관계에 있는 요소

시스템을 성공적으로 인도하려면 요구된 품질속성을 만족시켜야 하는 것은 필수적이다[5]. 그러므로 컴포넌트 추출 시에 비기능 요구사항을 고려할 수 있도록 비기능 요구사항의 분석과 이를 컴포넌트에 반영하는 것이 필요하다.

3. 컴포넌트 추출 기법

제안하는 비기능 요구사항을 고려한 컴포넌트 추출 기법은 독립적인 컴포넌트의 활용 증진을 위한 것이며 아래의 표 1 은 제안 기법에 대한 절차를 나타낸다.

<표 1> 제안 방법의 기본 개념 및 절차

단계	활동
비기능 요구사항 분석	Step1. 유스케이스 작성 Step2. 품질 속성 파악 및 비기능 요구사항 도출 Step3. 유스케이스와 비기능 요구사항 매핑
컴포넌트 추출	Step4. 컴포넌트 추출 Step5. 컴포넌트 명세

다음은 표 1 과 같이 비기능 요구사항을 반영하기 위한 요구사항 분석과 컴포넌트 추출을 각각의 Step 에서 상세히 기술한 것이다..

Step1. 유스케이스 작성

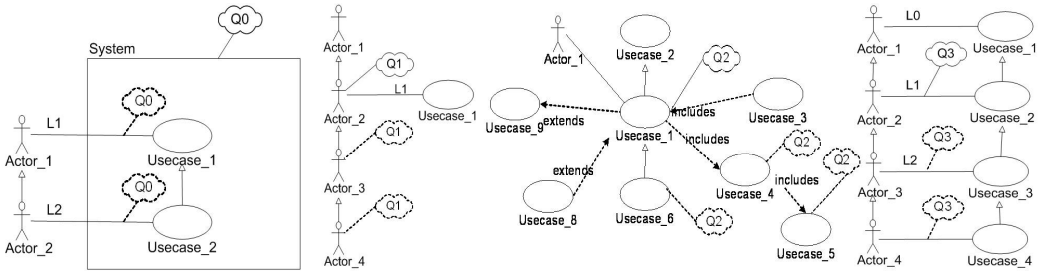
유스케이스를 작성하기 위해서는 요구사항이 도출되어 있어야 한다. 여기에서는 요구사항이 도출되어 있거나 유스케이스가 이미 작성되있다고 가정하고, 이를 토대로 시스템의 범위를 결정한 뒤 하나의 사용사례 단위의 기능성으로 유스케이스와 유스케이스 다이어그램을 작성한다. 본 단계에서의 기능 중심 유스케이스 다이어그램은 다음 절차들을 거치면서 비기능 요구사항이 반영되어 정제된다. 유스케이스를 위한 시나리오는 보통 일반 경로, 실패경로, 대안경로 등으로 구성되며, 여기에서는 품질 속성을 파악하기 위해 사용사례를 기반으로 하는 품질속성 시나리오를 추가시킨다. 구체적인 품질 요구사항과 제약사항을 추출해 낼 수 있기 때문이다[6]. 또한 유스케이스 다이어그램에서 액터(Actor)와 유스케이스(Use case) 사이의 관계는 두 개체 사이의 인터페이스와 같다. 그러므로 컴포넌트를 추출 후 인터페이스 정의 및 관련 제약사항을 도출하는 과정에서도 이용한다.

Step1. 품질 속성 파악 및 비기능적 요구사항 도출

이 단계에서는 비기능적 요구사항의 분석을 위한 몇 가지 원칙을 정의한다.

① 유스케이스 다이어그램에서 비기능적 요구사항과의 관계점을 정의하고 이를 중심으로 품질 요구사항을 고려한다.
② 유스케이스 다이어그램에서 비기능적 요구사항의 전파 규칙을 정의하고 이를 적용시켜 분석한다.
③ 소프트웨어의 비기능적 요구사항은 기능적 요구사항의 속성이거나 품질 요구사항이다.
④ ISO/IEC 9126[9]의 소프트웨어 품질 속성을 기본으로 사용한다.

소프트웨어 품질속성은 6 가지의 대표적인 속성과 각



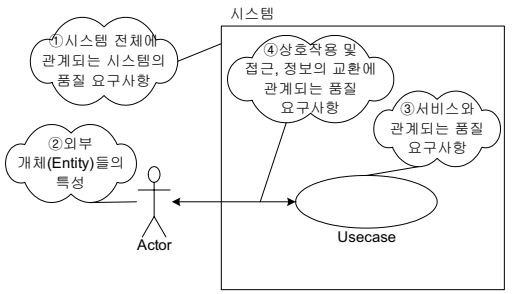
(a) System-NFR 전파 규칙 (b) Actor-NFR 전파 규칙 (c) Actor-NFR 전파 규칙 (d) Actor-NFR 전파 규칙

(그림 3) 비기능 요구사항 전파 규칙의 예

속성별로 다시 다수의 세부속성이 분류 및 정의되어 있다[9]. 품질 요구사항 식별 시 [2]에서 제안한 4 가지의 비기능 요구사항과의 관계점을 바탕으로 목표 도메인에 관련된 품질속성들과 비기능 요구사항들을 파악하고, 매핑시킨다. 유스케이스와 비기능적 요구사항과의 관계점은 아래의 그림 2 와 같으며, 매핑시킨 결과를 매핑표를 작성하여 기록한다. 매핑표에서는 비기능적 요구사항 및 비기능적 요구사항에 대한 설명과 관련 품질 속성, 속성 명세를 항목으로 작성한다.

Step3. 유스케이스와 비기능 요구사항 매핑

Step2 에서의 관계점을 바탕으로 [2]에서 제안한 비기능 요구사항 전파 규칙을 통해 유스케이스 다이어그램에서 유스케이스와 관계성이 있는 비기능 요구사항을 매핑시킨다. 매핑표의 항목은 유스케이스 및 관련된 기능 요구사항과 비기능 요구사항 그리고 각 요구사항에 대한 설명, 관련 품질 속성으로 기록한다. 이러한 전파 규칙은 그림 3 에 가시적으로 나타내었으며 규칙 정의를 다음과 같이 기술한다.



(그림 2) 유스케이스에서의 품질 요구사항과의 관계점

① System-NFR 전파 규칙: 시스템과 연관된 비기능 요구사항은 시스템과 외부 개체 사이의 인터페이스에 영향을 미친다.

② Actor-NFR 전파 규칙: 액터 i 와 연관된 비기능 요구사항은 액터 i 로부터 분화된 액터 j 에 영향을 미친다.

③ Usecase-NFR 전파 규칙: 유스케이스 i 와 연관된 비기능 요구사항은 유스케이스 i 로부터 분화된 유스케이스 j 에 영향을 미친다.

④ Actor-Usecase assocation NFR 전파 규칙: 액터와 유스케이스 사이의 연관은 개체간의 인터페이스

로 볼 수 있으며, 이와 연관되어 있는 비기능 요구사항은 인터페이스 i 로부터 분화된 인터페이스 j 에 영향을 미친다.

유스케이스와 매핑된 비기능 요구사항을 유스케이스 다이어그램에 반영하여 가시화 시키기 위해 다음의 세 가지 원칙을 정의한다.

① 하나의 비기능 요구사항이 공통적으로 여러 유스케이스나 인터페이스에 반영이 된다면 독립된 유스케이스로 분리시킨다. 또한 영향관계에 있는 유스케이스와는 연관관계나 include 관계를 부여한다.

② 서로 상충되는 비기능 요구사항이 하나의 유스케이스나 인터페이스와 영향관계에 있다면 비기능 요구사항의 우선순위를 고려한다. 요구사항간의 충돌을 피하기 위해 우선순위가 상대적으로 높은 요구사항을 선택하여 연관관계나 include 관계를 부여한다.

③ 액터의 관점에 의해 선택적으로 요구되는 비기능 요구사항은 기본 유스케이스와 extend 관계를 부여한다.

비기능 요구사항과의 관계부여는 기존 유스케이스에서 지원하는 include, extend 관계를 그대로 적용한 것이다. 유스케이스에서 관계를 나타내는 include 는 기본 유스케이스가 수행될 때 반드시 같이 수행되어야 할 관계를 나타내며, extend 는 기본 유스케이스가 수행될 때 특정한 조건에 따라 선택적으로 수행되어야 하는 관계를 말한다.

Step4. 컴포넌트 추출

Step3 에서 정립된 기능, 비기능 요구사항간의 영향관계성을 바탕으로 비기능 요구사항까지 고려한 컴포넌트를 추출 한다. 본 논문에서는 상호작용을 통해 비즈니스 요구사항을 달성하는 비즈니스 시스템을 위한 관점으로 컴포넌트를 추출한다. 이 단계에서는 컴포넌트 추출을 위해 다음과 같은 네 가지 유형으로 유스케이스를 분류하여 서브시스템을 식별하고 컴포넌트를 추출한다.

① 유저 인터페이스 요소 : 사용자 화면을 생성하거

나 사용자가 입력한 내용을 보여주거나 결과를 조회하는 기능을 말한다.

② 비즈니스 로직 요소 : 시스템의 비즈니스 로직을 수행하기 위해 다양한 연산을 포함하는 기능을 말한다.

③ 데이터 엑세스 요소 : 데이터 엑세스 요소는 파일 및 데이터베이스를 통해 READ 나 WRITE 와 같은 오퍼레이션을 사용하여 시스템이 사용하는 데이터와 정보를 조작하는 기능을 말한다.

④ 공통 요소 : 후보 컴포넌트가 식별된 상태에서 공통적으로 필요하거나 존재하는 기능을 말한다.

일반적으로 주기능은 유저 인터페이스, 비즈니스 로직이나 데이터 엑세스 요소를 포함한 유스케이스가 담당하게 되며, 이것을 포함하는 유스케이스를 키(Key) 유스케이스로 간주한다. 컴포넌트는 서브시스템을 이루는 구성요소이므로 주기능을 포함하고 있는 키 유스케이스를 중심으로 식별하여 추출한다. 그리고 매핑표는 제약사항 및 품질 요구사항과 관련된 비기능 요구사항을 매핑시킨 결과를 나타낸다.

Step5. 컴포넌트 명세

Step4 에서 식별한 컴포넌트에 대하여 관련된 비기능 요구사항 및 제약사항에 대해 명세한다. 이 단계에서는 재사용과 개발을 위한 정보를 선택하고 적절한 항목을 구성하여 작성한다. 재사용을 위한 정보로는 컴포넌트, 인터페이스, 오퍼레이션 명이나 오퍼레이션 인자 및 리턴타입 등 을 예로 들 수 있다. 개발을 위한 정보로는 컴포넌트 명, 컴포넌트 간의 관계, 인터페이스와 오퍼레이션 명, 오퍼레이션 인자타입 및 리턴타입, 인터페이스의 속성타입, 인터페이스의 외부 인터페이스와의 관계, 인터페이스의 내부에 관련된 상세한 구현 제약사항들이 있다. 상호작용과 관련된 인터페이스 및 관련 제약사항 분석은 시퀀스 다이어그램을 이용하여 상세하게 파악하는 것이 좋다. 식별된 컴포넌트는 문서화를 통해 잘 기술된 제약사항 및 품질 요구사항에 대한 정보를 가지고 있기 때문에 보다 효과적인 컴포넌트를 이용한 개발을 가능케 한다.

다음의 표 2 는 각 활동을 통해 생성된 산출물을 나타낸다. 기본적으로 산출물의 연속성 있는 관리를 위해 기능, 비기능 요구사항, 컴포넌트와 관련한 항목에 ID 를 부여하여 작성했다.

<표 2> 컴포넌트 추출 기법의 산출물 목록

단계	활동	산출물
비기능 요구사항 분석	Step1. 기능 요구사항으로부터 유스케이스 작성	유스케이스 모델
	Step2. 품질 속성 파악 및 비기능 요구사항 명세	품질 속성-비기능 요구사항 매핑표
	Step3. 유스케이스와 비기능 요구사항 매핑	기능 요구사항-비기능 요구사항 매핑표
컴포넌트 추출	Step4. 컴포넌트 추출	컴포넌트-비기능 요구사항 매핑표
	Step5. 컴포넌트 명세	컴포넌트 명세표

4. 결론 및 향후 연구

실행 가능한 기존 컴포넌트를 사용하여 새로운 시스템을 구축할 때, 이 컴포넌트를 구현한 개발자에 의해 이루어진 상세한 설계 결정 때문에 필연적으로 제약을 받게 된다.

본 논문에서는 위와 같은 기존 컴포넌트의 문제점인 품질적 측면과 제약사항을 효과적으로 파악하기 위해 컴포넌트에 대한 비기능적 요구사항 분석과 추적성에 대한 연구를 진행하였다. 이를 위해 규칙 기반의 접근 및 시나리오를 이용하여 비기능적 요구사항을 분석하였으며, 기능, 비기능 요소간의 영향관계를 정립할 수 있었다. 최종적으로 기존에 가시화되지 않았던 비기능적 요구사항을 유스케이스 다이어그램을 통해 가시화시켰다.

이 연구를 활용하여 목표 도메인에 적합한 컴포넌트의 개발 및 선택과 활용을 증진 시킬 수 있다. 또한 컴포넌트가 가진 고유의 문제점을 극복하는데 이용될 수 있을 것으로 기대된다. 뿐만 아니라 컴포넌트에 대해 불가피한 수정이 필요한 상황에서도 유스케이스를 이용하기 때문에 분석가와 이해관계자간의 분석 및 대화 활동에 편의를 제공할 수 있을 것이다.

향후 연구로는 컴포넌트의 독립적인 재사용 증진을 위하여 비기능 요구사항 분석 및 요구사항간의 상충되는 문제를 해결하는 구체적인 기준과 가이드라인을 제시하는 연구가 필요하다. 또한 기능-비기능 요구사항을 통합한 컴포넌트 추출에 대한 검증 및 정제가 추가적인 연구로 이루어져야 한다.

참고문헌

[1] I. Sommerville, Software Engineering(8th Edition), Addison Wesley, 2004

[2] S. Supakkul, L. Chung, “Integrating FRs and NFRs : A Use Case Goal Driven Approach”, In Proceedings of the 2nd International Conference on Software Engineering Research, Management and Applications(SERA), pp.30-37, 2004.

[3] H. Kaiya and A. Osada and K. Kaijiri, “Identifying Stakeholders and Their Preferences about NFR by Comparing Use Case Diagrams of Several Existing Systems”, Prod. Of the International Conference on Requirements Engineering, pp.112-121, 2004.

[4] B. Bohem, Victor R. Basili, “Software Defect Reduction Top 10 List”, IEEE Transactions on Software Engineering, 2001.

[5] M. El-Wakil, A. El-Bastawisi, M. Boshra, “Object-Oriented Design Quality Models A Survey and Comparison“, International Conference on Informatics and Systems, 2004.

[6] L. Bass, P. Clements, R. Kazman, Software Architecture in Practice(2nd Edition), Addison Wesley, 2003

[7] Karl E. Wiegers, More About Software Requirements: Thorny Issues and practical Advice, Microsoft Press, 2006

[8] J. Mylopoulos, L. Chung, and B. A. Nixon. Representing and using nonfunctional requirements: A process oriented approach. IEEE Transactions on Software Engineering, 18(6):483–497, 1992.

[9] International Standard ISO/IEC 9126-1, “Software engineering - Product quality - Part 1: Quality model,” 2001.