제22회 한국정보처리학회 추계학술발표대회 논문집 제11권 제2호 (2004. 11)

품질속성을 이용한 목표기반요구분석

조장희, 김귀연, 김병기
전남대학교 전산학과 소프트웨어공학연구실
veronica@natural.chonnam.ac.kr, {gykim, bgkim}@chonnam.ac.kr

Goal-based requirements Analysis using Quality Attribute

Jang Hee Cho, Gwi Yeon Kim, Byung Ki Kim
Dept. of Computer Science, Chonnam National University

요 약

소프트웨어 개발에서 요구사항 분석에 대한 관리가 품질과 생산성에 중요한 역할을 하고 있다. 기존 연구에서는 요구사항 분석에 있어 기능중심으로 문제 분석을 시도하고 , 구현 또는 시험단계에서 품질문제를 고려하고 있다.
본 논문에서는 요구사항을 추출하고 분석하는 단계에서 품질속성을 고려하는 요구사항 프로세스를 제안한다. 품질속성(quality attribute)을 개발초기인 요구사항 추출, 분석 단계에서부터 고려하여 개발하도록 함으로써 명확한 요구사항에 대한 이해와 시스템의 품질 향상을 가져다줄 수 있는 기반을 제공한다.

1. 서론

현대사회에서 과학기술의 발달은 인류에게 큰 편의를 제공하게 되었다. 특히, 소프트웨어가 과학기술에 차지하는 비율은 점차 증가하고 있다. 기존의 사무 자동화를 위하여 사용되었던 소프트웨어가 이제는 인간의 생활에 없어서는 안되는 위치까지 오게 되었다. 따라서, 소프트웨어가 실생활에 미치는 파급 효과는 매우 크다[1].
특히 소프트웨어 개발 단계에서 요구사항들의 구체적인 정의는 성공적인 소프트웨어 개발의 필수 요건이다. 설계와 구현이 잘되더라도 사용자의 요구가 반영되지 못한 소프트웨어는 실패작일 뿐이다. 이러한 변화는 요구공학이라는 새로운 개념을 연구개발(R&D)분야에 등장시켰다.
요구공학(Requirements Engineering)이란 제품 개발을 위한 요구사항 설정단계에서부터 제품의 개발과 테스트, 생산에 이르기까지 개발공정의 매단계마다 초기에 정한 개발 요구사항들은 물론 이후에 상세 요구사항들이 제품설계와 구현단계까지 제대로 지켜지고 있는지를 검증해 나가는 기법을 의미한다[2]. 그러나 문제는 이러한 과정이 실제적으로는 수행하기 어려워 성공적인 소프트웨어 개발을 위협하고 있다.
이와 같이 초기단계의 사용자 요구사항을 정확히 파악하고 이해하는 것은 성공적인 소프트웨어 개발에 중요한 척도가 된다. 본 논문에서는 요구사항을 추출하는 방법에 대한 기존의 연구를 살펴보고 효율적인 요구사항 추출에 관한 새로운 접근방법을 제시한다.
본 논문의 구성은 다음과 같다. 2장에서는 요구사항의 중요성과 요구사항의 두 가지 관점에 대해 살펴본다. 또한 본 연구와 관련 있는 기존연구들을 살펴본다. 3장에서는 기존 연구에서 제시되어 그 성능을 평가받은 목표기반(Goal-based) 요구분석방법을 기반으로 시스템의 품질을 결정짓는 품질속성(quality attribute)을 요구사항 추출, 분석단계에서부터 고려하여 개발하도록 함으로써 명확한 요구사항에 대한 이해와 시스템의 품질 향상을 가져다 줄 수 있는 새로운 접근방법을 제시한다. 마지막으로 4장에는 결론 및 향후 연구방향을 기술한다.

2. 관련연구

2.1 요구분석의 중요성

소프트웨어 요구사항은 개발에서부터 구현까지 납기, 비용, 품질에 결정적인 영향을 미치는 요소 중의 하나이다. 세계적인 소프트웨어 생산성 연구기관인 SPR에서는 소프트웨어 프로젝트가 실패하는 가장 근본적인 원인을 소프트웨어 요구공학의 미성숙과 부적절한 견적 및 계획으로 보고하고 있다.
즉, 올바른 요구사항을 조기에 추출하지 못함으로 인해 소프트웨어 개발에 있어서 오류가 늦게 발견될수록 문제해결이 더욱 어려워지며 오류수정 비용도 커진다. 어떠한 소프트웨어 개발방법론을 사용하더라도 모든 설계의 기초는 요구사항이며 이는 적절히 정의되어야 한다.
고객의 요구사항을 정확히 정의하면 할수록 다음 단계에서 발생하는 오류 또는 부적합사항은 줄어들게 된다. 요구사항의 정의 및 검증은 고객의 요구사항을 정의하는 것으로부터 출발한다. 개발의 각 단계마다 고객의 요구사항은 요구명세로 변환되고 이 명세는 역으로 고객의 요구사항과 일관성이

있어야 한다.
정확한 요구사항의 정의는 고객의 제품에 대한 만족도를 향상시키고 유지보수 및 지원비용을 절감시킨다. 또한 개발과정에서의 재작업을 축소로 인해 개발주기를 단축시키고 높은 생산성과 품질향상을 가져온다.
그러나 고객의 요구사항을 만족스럽게 기술한다는 것은 어려운 일이므로 고객의 요구사항 명세서는 오류의 근원이 될 확률이 높다. 부적절한 요구사항은 설계를 불가능하게하고, 목표가 명확히 정의되지 않음으로 인해 생산과 관리를 효율적으로 수행시킬 수 없게 한다.
사용자의 요구는 소프트웨어가 갖추어야 할 요건이다. 일반적으로 소프트웨어의 요구를 분류하기란 쉽지 않으나 크게 기능적 요구(Functional Requirements)와 비 기능적 요구(Non-Functional Requirements)로 구분해 이해할 수 있다. 기능적 요구와 비 기능적 요구에 대한 예를 간단히 살펴보면 <표 1>과 같다.

<표 1> 기능적 요구와 비기능적 요구의 예

◆ 기능적 요구 (functional requirements)	◆ 비 기능적 요구 (non-functional requirements)
- 처리 및 절차 - 입출력양식(한글,한자,색상) - 명령어의 수행결과 - 키보드 조작 - 주기적인 자료출력	- 성능 : 응답시간, 처리량 - 신뢰도 - 기밀 보안성 : 불법접근 방지 - 운용제약 - 개발비용 : 투자 한계

2.2 요구분석 방법의 종류

2.2.1 유스케이스 지향분석(Use case driven analysis)

요구사항과 유스케이스 간의 관계를 보면 유스케이스는 행위단위를 기술한 것이고 요구사항은 행위에 적용된 법칙을 기술한 것이다. 유스케이스는 하나 혹은 그 이상의 기능적 요구사항을 만족할 수 있으며, 기능적 요구사항은 하나 혹은 그 이상의 유스케이스에 의해 만족 될 수 있다.
이러한 유스케이스 지향의 분석 방법[3]에는 사용자가 그림을 통하여 요구사항을 이해할 수 있다는 장점이 있고, 이미 보편적으로 받아들여지는 UML(Unified Modeling Language)기반으로 이뤄져 있다는 점에서 큰 장점을 가지고 있다.
하지만 요구 사항 분석이 사용자 입장에서라기 보다는 개발자 입장을 고려한 분석 방법이다. 이는 유스케이스를 뽑고 나눌 때에 개발될 당시의 고려사항을 염두 하고서 진행된다는 점에서 그렇다. 유스케이스는 각각 독립적이다. 따라서, 단순한 유스케이스의 나열은 시스템 전체적인 요구사항을 이해하기 어렵다.
또한 유스케이스 간에 독립은 향후 유스케이스 위주의 개발 시 나타나는 클래스 간의 관계를 볼 때 어긋나게 된다. 즉, 독립적인 유스케이스 개발임에도 불구하고 설계나 구현단계에서 다른 유스케이스로부터 나온 클래스와 관계를 가질 경우가 발생하기 때문이다.
그러므로, 소프트웨어 시스템의 품질을 결정하는 비 기능적 요구사항은 유스케이스 지향분석 방법으로 나타내기가 쉽지 않다. 이는 유스케이스가 시스템의 액터와 상호작용을 위주로 기술되기 때문이다.

2.2.2 시나리오 기반의 요구사항분석(Scenario based requirement analysis)

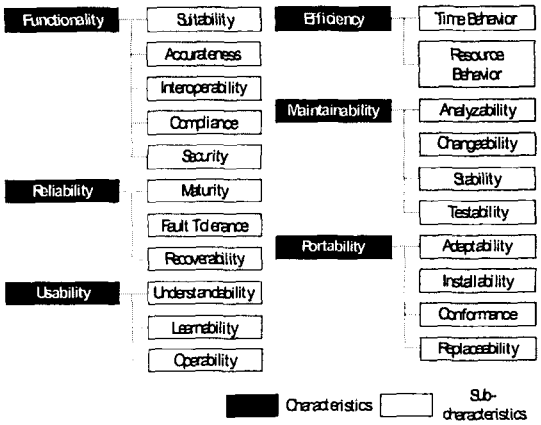
시나리오 기반의 요구사항 분석[4]는 일반 사용자가 접근하기 용이한 예나 실세계 경험을 추출하는 시나리오라는 것을 사용하여, 사용자가 자신의 시스템 요구사항을 기술하는데 편리하도록 지원하는 방법이다. 시나리오 기반의 요구사항 분석 방법은 사용자가 친숙한 시나리오를 기반으로 분석한다는 점에서 사용자 입장을 고려한 접근방법이다.
그러나 시스템을 전체적으로 볼 수 있는 부분이 약하기 때문에 시스템의 기능적인 부분을 부분적으로 알 수 있고, 시나리오 기반으로 분석하기 때문에 제약적인 요구사항만을 추출할 수 있다. 또한 제안하는 방법에서 시작이 될 수 있는 시나리오 구조 모델은 반복적으로 과정을 진행할 때 변경을 할 수 없는 문제점이 있으며, 이의 타당성에 대한 검출 방안이 존재하지 않는다.

2.2.3 목표기반의 요구사항 분석

목표기반의 요구사항 분석[5]는 목표라는 개념을 통하여 요구 사항을 분석하여 요구 사항 기술서를 만들어내는 방법이다. 목표(goal)라는 것은 이해 관계자가 개발 될 시스템에서 성취하고자 원하는 것으로 "비지니스와 조직과 시스템의 최상 위 레벨의 목표들"로 정의 한다. 목표 기반의 분석을 위한 실행단계는 목표분석(goal analysis)과 목표진화(goal evolution)로 나눌 수 있다.
목표분석은 "목표 정의(identify goals)", "목표 분류(classifying goals)", "이해관계자와 에이전트의 정의(identify agents and stakeholders)"의 세 단계로 나눠진다.
목표 정의는 초기 단계에서 목표를 정의하는 단계이다. 하지만, 사용자와 개발자들은 시스템의 목표를 잘 파악하기 어렵다. 이는 목표라는 것이 최 상위 레벨의 요구 사항이기 때문에 처음부터 목표를 추출한다는 것은 상당한 어려움을 가지게 된다. 일반적으로 목표를 정의하기 위해 "textual introductory statement", "textual process scenario", "flow chart of the existing process"와 같은 방법들을 사용한다. 목표 분류는 전 단계에서 나타난 목표들의 리스트들을 분류하는 작업이다. 이를 통해 기능적, 비 기능적인 목표들을 정리하고 배분한다. 이해관계자와 에이전트의 정의는 목표와 관련된 에이전트와 이해 관계자들을 정의한다. 목표 진화는 분석된 목표들을 이루기 위한 시나리오를 생성하고, 제약 상황 등을 기술하게 된다.
목표기반의 분석방법은 시스템의 최상위 레벨에서 요구사항을 분석할 수 있다는 점에서 시스템 전체에 대한 요구 사항을 파악할 수 있다. 또한 시스템의 기능적, 비 기능적 요구사항을 파악하므로 균형 있는 요구사항 분석이 가능하다.

2.3 품질속성 평가

품질속성은 양이나 질로 관찰하여 수치로 측정할 수 있는 시스템의 속성을 말한다. 품질속성은 이해관계자들의 관심사와 요구사항을 그대로 반영하고 아키텍처는 이해관계자들이 원하는 수준으로 품질속성을 달성해야 한다.
단순히 성능이 좋다거나 오류가 없다거나 쓰기 쉽다는 것으로 소프트웨어 시스템의 품질을 규정할 수 없다. 품질모델은 품질속성을 분류하고 정의하여 누구나 인정할 수 있는 품질 측정 기준을 정의한 것이다. 품질 모델을 도입하면 소프트웨어 시스템의 품질을 정형화(formalization)할 수 있다. 이러한 S/W품질 모델 ISO/IEC 9126을 살펴보면 (그림 1)과 같다.



(그림 1) ISO/IEC 9126

ISO/IEC 9126은 소프트웨어의 품질 속성을 기능성, 신뢰성, 사용성, 효율성, 유지보수성, 이식성으로 구분하고 있다. (그림 1)의 주요 속성을 살펴보면 <표 2>과 같다.

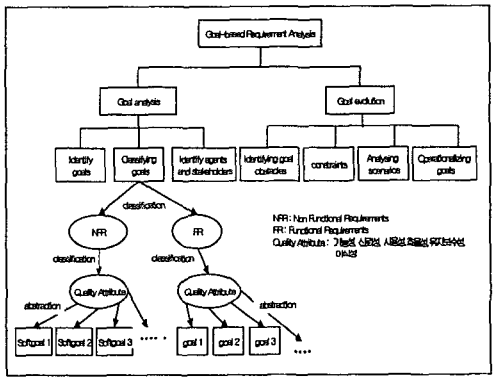
<표 2> S/W 품질속성 평가(ISO/IEC 9126)

기능성	규정된 기능 및 성능을 정확하게 충족하는 능력
신뢰성	고장 없이 성능수준을 지속적으로 유지하는 능력
사용성	사용자가 쉽게 이해하고, 학습할 수 있게 하는 능력
효율성	부하의 변화에 자원을 적절하고 효율적으로 운영하는 능력
유지보수성	소프트웨어의 수정, 개선을 용이하게 하는 능력
이식성	다양한 운영환경에 적응할 수 있는 능력

요구사항을 추출하고 분석하는 단계에서 전체 시스템에서 차지하는 요구사항 항목들을 정하고 분류하는 것은 향후 설계 및 아키텍쳐 생성, 구현단계에서 개발자와 사용자 모두에게 폭 넓은 개발 방안을 제공하게 된다. 따라서, 추출 단계에서 품질을 고려한다면 시스템 개발시 상당한 비용 감축을 가져올 수 있다.

3. 품질속성을 이용한 목표기반 요구사항분석

요구사항을 추출하고 분석하는 단계에서 체계적으로 아키텍처와 품질속성간의 관계를 잘 정리하면 분석 및 설계과정에서 큰 향상을 기대할 수 있다.
추출 단계에서 품질을 고려한다면 시스템 개발 시 상당한 비용 감축을 가져올 수 있다. 고객이 원하는 품질을 달성하려면 시스템의 기능적 요구사항과 비 기능적 요구사항을 모두 충족 시켜야한다. 그러나 비 기능적 요구사항과 관련된 품질 속성들은 서로 상충하는 경우가 많기 때문에 모든 품질속성을 100% 만족시킬 수는 없다. 그러므로 전체 품질은 각 품질속성의 목표를 적절한 수준에서 조율한다.
아키텍처는 시스템이 달성해야 하는 품질들을 실현할 때 필요한 큰 그림을 결정하기 때문에 중요하다. 따라서 품질을 달성할 수 있도록 아키텍처를 분석하고 설계해야 한다.



(그림 2) 제안된 품질속성을 이용한 요구사항분석

Algorithm : classifying process of classifying goals

```
if ( Requirement == NFR ){ /* Requirement == NFR */
/* Abstraction through the Six Quality Attribute */
        switch( Quality Attribute ){
                case Functionality :
                        softgoal_1;
                        break;
                case Requirement :
                        softgoal_2;
                        break;
                case Usability :
                        softgoal_3;
                        break;
                case Efficiency :
                        softgoal_4;
                        break;
                case Maintainability :
                        softgoal_5;
                        break;
                case Portability :
```

```
                softgoal_6;
                break;
        }
}else {   /* Requirement == FR */
/* Abstraction through the Six Quality Attribute */
        switch( Quality Attribute ){
                case Functionality :
                        goal_1;
                        break;
                case Requirement :
                        goal_2;
                        break;
                case Usability :
                        goal_3;
                        break;
                case Efficiency :
                        goal_4;
                        break;
                case Maintainability :
                        goal_5;
                        break;
                case Portability :
                        goal_6;
                        break;
        }
}
```

제안된 방법은 품질속성을 내포하는 비 기능적 요구사항에 대한 분류를 통해 품질속성들의 관점에서 시스템의 비 기능적 요구사항을 추가적으로 추출할 수 있다.
또한 목표분석 단계의 목표 분류를 통해서 전 단계에서 나타난 목표들의 리스트들을 분류하는 작업을 (그림 2)와 같이 세분화 시킬 수 있다. 먼저 목표분류 부분을 기능적, 비 기능적 요구사항으로 분류한다. 다시 ISO/IEC 9126의 소프트웨어의 품질속성을 이용하여 좀더 세부적으로 분류하여 softgoal과 goal을 추출한다. 이는 시스템의 최상위 레벨에서 요구사항을 정확히 분석해줌으로써 균형있는 요구사항분석을 가능하게 한다. 이에대한 처리 단계는 위의 알고리즘과 같이 이루어진다.

4. 결론 및 향후 연구방향

우리의 실생활에 소프트웨어가 밀접한 관계가 있게 되고 소프트웨어 품질은 이러한 소프트웨어 개발에 중요한 요소이기 때문에 소프트웨어 품질을 나타내는 비 기능적 요구사항에 대한 관심이 높아지고 있다.

본 논문에서는 목표기반 요구사항 분석을 기반으로 기존에 형식이 없고 모호하게 처리되는 비 기능적 요구사항을 품질속성을 이용하여 명확하게 추출할 수 있는 근거를 제시하였다. 또한 기능적 요구사항을 중심으로 이에 관련된 비 기능적 요구사항을 추출하고 이를 이용하여 추출되지 못한 시스템 레벨의 비 기능적 요구사항을 추가적으로 추출할 수 있다.

향후에는 이 연구를 좀 더 보완하여 추출된 기능적 요구사항과 비 기능적 요구사항간의 모델링을 하는 방법에 대한 연구와 더 나아가 요구사항 단계에서 다뤄진 비 기능적 요구사항을 설계 단계까지 연결시키는 방안에 대한 연구가 필요하다.

참고문헌

[1] 김진태, 이은미, 박수용 "시나리오 기반의 기능적, 비기능적 요구사항 분석 방안", 한국소프트웨어공학 회지 제16권 1호, pp3-18, March, 2003.

[2] [1] Richard H.Thayer and Merlin Dorfman Software Requirements Engineering, 2nd Edition IEEE Computer Society Press 1997.

[3] Ivar Jacobson, Grady Booch, James Rumbaugh, "The Unified Software Development Process", Addison Wesley, 1998

[4] Sutcliffe, A. "Scenario-based requirement analysis" Requirement Engineering Journal, 3 (1998), pp 48-65

[5] Annie I. Anton, "Goal-Based Requirements Analysis", Proceeding of ICRE '96, pp 136-144

[6] 정기원, 윤창섭, 김태현 "소프트웨어 프로세스와 품질",홍능과학출판사