아키텍처 모델링을 위한 유스케이스 기반의 요구사항 정량화 기법

정창해, 양원석, 박수용
서강대학교 컴퓨터학
{zenlover, hiyws, sypark} @sogang.ac.kr

Usecase-based Requirements Quantitative Analysis Approach to Architecture Modeling

Changhae Jung, Wonseok Yang, Sooyong Kim
The Department of Computer Science, Sogang University

요 약

아키텍처는 요구사항을 기반으로 생성되어야 한다. 특히 비기능적 요구사항은 아키텍처의 생성에 많은 영향을 미치는 요인이다. 본 논문은 아키텍처를 생성하기 위한 요구사항 분석 기법과 전략을 제안한다. 제안하는 방법은 유스케이스를 이용하여 기능 요구사항을 추출하고, AHP(Analytic Hierarchy Process)를 이용하여 비기능 요구사항의 중요도를 정량적으로 분석한다.

1. 서론

요구공학(Requirements Engineering)과 소프트웨어 아키텍처(Software Architecture)는 성공적인 소프트웨어 개발을 위한 중요한 영역이다[1]. 요구공학은 개발될 소프트웨어 시스템의 기능 및 비기능적 속성을 정의한다[2]. 소프트웨어 아키텍처는 소프트웨어 시스템의 구성요소 및 구조를 나타낸다[3].

요구사항은 아키텍처 설계의 핵심 요소다[4]. 요구사항은 기능적 요구사항(Functional Requirements)과 비기능적 요구사항(Quality Requirements)으로 구분된다. 기능적 요구사항은 아키텍처를 구성하는 서브 시스템 또는 컴포넌트의 생성에 영향을 미친다. 반면에 비기능적 요구사항은 아키텍처의 구조에 영향을 미친다[6]. 따라서 요구사항은 아키텍처 생성의 근거를 제시해 준다.

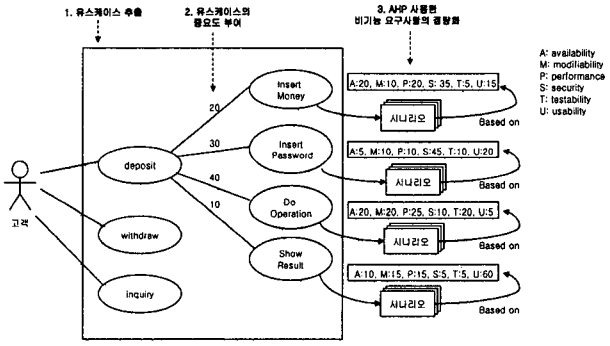
요구 공학과 아키텍처에 관한 연구는 여러 커뮤니티에서 이루어 지고 있다. 유스케이스는 기능 요구사항을 추출하기 위해 사용된다[5]. 그러나 유스케이스는 아키텍처 구조에 영향을 주는 비기능적 요구사항을 추출하는 부분이 미약하다. Chung은 비기능적 요구사항과 아키텍처 스타일의 관계를 연구하였다[6]. 비기능적 요구사항과 아키텍처 스타일의 관계 연구는 ABAS(attribute based architecture style)에서 좀더 정제된 기법을 제공한다[7]. 그러나 이들 방법은 비기능적 요구사항을 정성적(qualitative)으로 파악한다. 정성적 분석은 비기능적 요구사항에 대한 중요도를 매우 주관적으로 평가하게 된다. 따라서 요구사항으로부터 아키텍처를 설계하는데 객관적인 근거를 제시하기 어렵다.

본 논문은 유스케이스를 이용하여 비기능적 요구사항을 정량적으로 분석하는 방법을 제안한다. 제안하는 방법은 유스케이스를 이용하여 기능 요구사항을 추출하고, 비기능 요구사항의 중요도를 설정하기 위해 AHP(Analytic Hierarchy Process)를 이용한 정량적 분석 방법을 제안한다. 비기능적 요구사항의 정량적 중요도는 시스템에서 요구하는

비기능적 요구사항의 우선 순위를 객관적으로 평가해 주고 아키텍쳐 구조를 결정하기 위한 근거를 제시한다.

2. 본문

본 장에서는 요구사항으로부터 아키텍쳐를 생성하기 위해 해결해야 하는 문제점들을 나열하고 각각에 대한 전략을 설명한다. 본 연구에서 제시하는 절차는 그림1과 같다. 첫째, 유스케이스 기반의 분석을 통해 기능 요구사항을 추출한다. 둘째, 식별된 유스케이스의 상하위간의 중요도를 부여한다. 셋째, 각각의 하위 유스케이스에서의 비기능 요구사항을 시나리오를 근거로 하여 추출하고 정량화 하여 전체 시스템의 품질 속성을 예측한다



[그림 1] 전체 공정

2.1 유스케이스를 이용한 기능요소 분석

개발할 시스템의 기능적인 요구사항을 나타내는 유스케이스를 생성한다. 유스케이스는 시스템을 개발하는 과정에서 개발팀 별로 나누어서 개발하는 단위가 된다. 이 유스케이스를 본 논문에서는 최상위 유스케이스라고 한다.

최상위 유스케이스가 생성되면, 최상위 유스케이스를 좀더 세분화하여 좀 더 작은 유스케이스로 분화한다. 유스케이스를 분화할 때 너무 세분화되지 않도록 하기 위해서 실행 동사(operational verb)가 나올 때까지 분화한다.

최상위 유스케이스의 기능은 해당하는 최하위 유스케이스들이 가진 기능적 요구사항의 합으로 표현될 수 있다. 예를 들면, ATM 개발에서 최상위 유스케이스로 deposit, withdraw, inquery로 3개를 생성했다고 하면, 최상위 유스케이스는 deposit, withdraw, inquery 유스케이스가 된다. deposit 유스케이스를 좀더 분화하면 insert money, insert password, do operation, show result 유스케이스로 나누어 질 수 있다. 나누어진 유스케이스의 기능의 합이 바로 deposit 유스케이스의 기능이 된다.

2.2 정량적 분석

2.2.1 유스케이스의 중요도 계산

유스케이스의 분화가 끝나게 되면, 하위 유스케이스가 상위 유스케이스를 성취할 때의 기여도를 측정한다. 상위 유스케이스를 만족시키기 위해서 기능적으로 분화된 각각의 하위 유스케이스에 기여도를 부여하는데, 여기서 상위 유스케이스에 대한 하위 유스케이스의 기여도의 합이 100이 되도록 기여도를 조정하여 측정한다. 예를 들면, deposit 유스케이스를 만족시키기 위해서 하위 유스케이스인 insert money는 20, insert password는 30, do operation는 40, show result는 10의 기여도를 가지게 된다.

2.2.2 비기능적 속성의 중요도 계산

분화된 최하위 유스케이스에 대해서 비기능적 요구사항을 측정하여 정량화한다. 이때 SAP(Software Architecture in Practice)[8]에서 사용되는 6가지 비기능적 요구사항(availability, modifiability, performance, security, testability, usability)을 이용한다.

하나의 최하위 유스케이스가 가지는 6가지 비기능적 요구사항을 AHP[9]를 이용하여 상대적으로 비교한다. 상대적인 비교를 통해서 나타난 우선순위를 정량화한다. 비기능적 요구사항 간의 상대적 중요도는 최하위 유스케이스에 해당하는 basic flow 및 alternative flow를 근거로 하여 측정한다.

상위 유스케이스의 비기능적 요구사항의 중요도는 최하위 유스케이스에서 측정된 정량화된 비기능적 요구사항의 값과 상위 유스케이스에 대한 하위 유스케이스의 기여도를 각각 곱하여 측정한다. 이 과정을 계속적으로 반복하여 최상위 유스케이스에서의 비기능적 요구사항의 값을 정량적으로 측정한다. 표1 참조.

[표1] 비기능 요구사항의 정략적 분석

	Insert Money Weight	Insert Password Weight	Do Operation Weight	Show Result Weight	
Availability	0.2 * 20	+ 0.05 * 30	+ 0.2 * 40	+ 0.1 * 10	= 14.5
Modifiability	0.1 * 20	+ 0.1 * 30	+ 0.2 * 40	+ 0.15 * 10	= 14.5
Performance	0.2 * 20	+ 0.1 * 30	+ 0.25 * 40	+ 0.15 * 10	= 18.5
Security	0.35 * 20	+ 0.45 * 30	+ 0.1 * 40	+ 0.05 * 10	= 25
Testability	0.05 * 20	+ 0.1 * 30	+ 0.2 * 40	+ 0.05 * 10	= 12.5
Usability	0.15 * 20	+ 0.2 * 30	+ 0.05 * 40	+ 0.6 * 10	= 17

2.3 아키텍처 생성 및 디자인 패턴 적용

지금까지 과정을 통해서 최상위 유스케이스에 대한 6가지 비기능적 요구사항의 정량적인 값을 측정하였다. 각각의 최상위 유스케이스가 전체 시스템에 미치는 중요도를 같은 방법으로 측정하여 전체 시스템에서 요구하는 비기능적인 요구사항을 측정할 수 있다.

비기능적인 요구사항은 시스템의 아키텍처를 생성하는 중요한 요소가 된다. 예를 들면, Layered architecture style은 availability 및 security가 높은 시스템에 적합한 반면, performance는 좋지 않다. 아키텍처 스타일을 사용하여 아키텍처의 기본 골격을 만들 수 있다.

최상위 유스케이스는 시스템의 개별적인 개발 단위들이다. 따라서 각 유스케이스에 대한 시스템의 부분을 개발하기 위해서 개발 팀은 적절한 디자인 패턴을 사용하게 된다. 이때 앞에서 측정된 정량적인 비기능적 요구사항을 이용하여 적절한 디자인 패턴을 적용하여 유스케이스에 해당하는 부분을 개발할 수 있다.

3. 결론

요구 공학과 아키텍처는 성공적인 소프트웨어 개발을 위한 중요한 영역이다. 본 논문은 요구사항으로부터 아키텍처를 생성하기 위하여 유스케이스를 기반으로 비기능적 요구사항을 정량적으로 분석하는 방법을 제안하였다.

정략적으로 분석된 비기능적 요구사항은 아키텍처를 생성하기 위한 근거가 된다. 비기능적 요구사항의 정량화는 정성적인 문제를 정량적인 방법으로 해결함으로써 체계적인 의사결정과 객관적인 평가를 가능하게 했다. 이를 통해 시스템에서 요구하는 아키텍처 구조를 결정하기 위한 근거를 제시할 수 있다.

본 논문의 초점은 아키텍처 생성을 위한 기초자료를 요구사항으로부터 분석하는 것임으로 구체적인 아키텍처 생성 방법을 다루지는 않았다.

향후 연구 과제는 보다 체계적이고 정확한 방법으로 아키텍처 스타일을 비기능적 요구사항을 분석하는 작업이 필요하다. 이를 통해 정량화된 비기능적 요구사항으로부터 아키텍처 스타일을 선택하는 근거를 더욱 명확히 할 수 있다. 또한 본 연구에서 제시된 방법을 여러 도메인에 적용하고 검증해 보는 작업이 필요하다.

참고문헌

[1] Daniel M. Berry, et al. Foreword by the Workshop Co-Chairs, STRAW 03 in conjunction with ICSE 03, Oregon, 2003.

[2] Alan M. Davis, Software Requirements Analysis & Specification, Prentice-Hall, 1990.

[3] Mary Shaw and David Garlan, Software Architecture: Perspectives on emerging discipline, Prentice-Hall, 1996.

[4] Maarit Harsu, From architectural requirements to architectural design, Report 34, Institute of Software Systems, Tampere University of Technology, May 2003.

[5] I. Jacobson, M. Christerson, et al., "Object-oriented Software Engineering. A Use Case Driven Approach", Addison-Wesley, 1992.

[6] Lawrence Chung, Brian A. Nixon, Eric Yu, "An Approach to Building Quality into Software Architecture", IBM Centre for Advanced Studies Conference, Proceedings of the 1995 conference of the Centre for Advanced Studies on Collaborative research 1995

[7] M.H. Klein, R. Kazman, L. Bass, J. Carriere, M. Barbacci, and H. Lipson, Attribute-Based Architectural Style, Proceedings of the First Working IFIP Conference on Software Architecture(WICSA1), San Antonio, TX, 225-243, February 1999

[8] Len Bass, Paul Clements, and Rick Kazman, Software Architecture in Practice, Addison-Wesley, 1998.

[9] T.L Saty, The Analytic Hierarchy Process, McGraw-Hill, New York, 1980.