Fillmore's Case 메커니즘 이용한 요구사항 분석: 유스 케이스 모델링 방법

김보연, 양효석, 손현승, 박병호, 박용범**, 김영철*

홍익대학교 소프트웨어공학연구실*
bob@selab.hongik.ac.kr*,
단국대학교 AI & IA 연구실**

요약: 기존 Abbott 의 Textual Analysis 1983, noun-verb analysis 처럼 클래스 식별에 초점을 두었다[9]. 그리고 성공적인 분석을 위해 경험적으로 유스 케이스부터 시작하여 객체를 찾도록 가이드하고 있다. 또한 기존의 언어학자인 Fillmore 의 Case Grammar 의 의미론 개념을 요구공학(Requirement engineering) 적용하여, 유스 케이스 추출 방법을 제안했다[1,2]. 이는 대형 소프트웨어 개발에서 더 큰 단위인 유스 케이스로부터 하향식(Top-down) 개발에 목적이었다[1,2]. 이 논문에서는 경험적인 숙련(Empirical practice) 의존 없이도, 개선된 Fillmore 방법을 통해 유스 케이스를 식별과 이를 기반으로 유스 케이스 모델링 방법을 보인다. 이는 고품질 소프트웨어 개발을 위한 분석과 모델링 기반이 될 것으로 본다. 적용 사례를 우체국 시스템을 통해 low level 유스 케이스를 구체적으로 보였다.

핵심어: 유스 케이스, 요구사항 분석, 요구공학

1. 서 론

현재의 소프트웨어는 대규모화, 복잡화, 분산화되고 있다. 고품질의 소프트웨어 개발은 철저한 요구사항 분석이 매우 중요하다. 기존의 개발 방법론에는 기능들을 분석하는 방법인 구조적 방법론과 구성요소들을 식별하고 연관성을 통하여 객체를 분석하는 방법인 객체지향 방법론 등이 있다. 이는 너무 작은 단위로 고품질 소프트웨어 개발을 위한 요구사항 분석에 적합하지 않다. 앞으로의 대형 시스템 개발은 더 큰 단위인 유스 케이스부터 하향식(Top-down)방식으로 상세한 단위까지 분석하는 방법이 적용되어야 한다. 더 큰 단위인 유스 케이스에서 하향식분석으로부터 정확한 요구사항의 이해를 통하여 개발에 초석이 되고자 한다.

고객의 요구사항은 개발의 모든 과정에 영향을 미친다. 본 논문은 유스 케이스를 식별하는 표준을 제시하고자 언어학자인 Fillmore 의 Case Grammar 를 요구공학에 접목하여 고객요구사항으로부터 유스 케이스 추출 및 모델링 절차를 제안한다. 제안한 유스 케이스 추출 알고리즘은 자연어로 작성된 고객의 요구사항을 보다 쉽고, 명확한 분석으로 유스 케이스를 추출한다. 이 논문은 2 장에서 연구 배경, 3 장에서는 개선 된 Fillmore 의 case 메커니즘과 유스 케이스 추출 및 모델링을 언급하고, 4 장은 결론을 언급한다.

2. 연구배경

언어학[3,4,5]이란 언어의 특성에 대하여 연구하는 학문이다. 즉, 언어의 구성요소에 대하여 과학적인 분석과 추론을 통하여 언어를 체계적이고 심층적으로 이해하는 학문이다. 하지만 언어는 인간이 생각에 의하여 생성되기 때문에 과학적인 연구에 대상으로 다루기에는 매우 힘들었다. 이를 체계화한 언어학자들의 이론을 요구공학에 접목하여 자연어로 작성된 요구사항을 체계적으로 분석하고자 한다. 언어에 대하여 과학적으로 접근한 Chomsky 는 인간이 생각하는 의미가 말로 전환되는 체계를 설명하였다. 하지만 Chomsky 의 이론은 표면적 관계만 고려하고, 심층적 의미가 고려되지 않아 요구공학에 적용하기에는 적합하지 않았다. Fillmore 는 Chomsky 의 이론보다 더 깊은 심층구조가 필요함을 주장하며 Case Grammar 를 주장하였다[5,6]. 심층주조에서 구조 격 대신에, 논항이 동사에 대해 가지는 의미 역(Semantic role)이라는 개념으로 문장을 기술하고자 하였다. 즉, 한 문장 내에서 서술부(동상)를 중심으로 의미적인 관계를 유지한 논항(명사)을 격(Case)라는 범주로 분류하는 것이다. Fillmore(1968)는 격의

† 본 연구는 미래창조과학부 및 한국산업기술평가관리원의 산업원천기술개발사업[10035708, 고신뢰 자율제어 SW 를 위한 CPS(Cyber-Physical Systems) 핵심 기술 개발]과 2013 년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(2013R1A1A2011601).

범주로 동작주격(Agent), 도구 격(Instrumental), 여격(Dative), 대상 격(Objective), 작위 격(Factitive), 처소격(Locative)으로 6 개의 내면 격을 제시하였다. 기존의 Fillmore 의 Case Grammar 는 동사와 관련되어 있는 명사구들을 격의 범주에 분류가 가능하다. 동작주격(Agent)이란, 동사에 의해 표현되는 동작을 일으킨다고 인식되는 주체를 의미한다. 도구 격(Instrumental)은 동사가 나타내는 동작이나 상태의 원인이 되는 물체를 의미한다. 여격(Dative)은 동사에 의해 표현되는 상태나 동작에 영향을 받는 사람이나 동물을 의미한다. 대상 격(Objective)이란 동사가 나타내는 동작과 상태의 영향을 받는 사물을 의미하며, 작위 격(Factitive)은 동사가 나타내는 행위와 상태의 결과로서 존재하는 사람이나 동물을 의미한다. 처소 격(Locative)은 동사가 나타내는 상태나 그 동작이 일어났던 위치를 나타내는 격이다. 이후로, 격 이론학자들에 의해 계속 추가되고 수정되고 있다. 기존의 Fillmore 의 Case Grammar 가 가진 장점은 매우 뚜렷하다. 같은 의미의 다른 문법형태로 작성된 문장들은 구조적 방법으로 접근하였을 때는 서로 다른 결과값으로 분석이 되지만, 의미론적 방법으로 접근하였을 때는 모두 같은 결과값을 얻을 수 있다. 이처럼 동사와 관련된 동일한 의미를 지닌 명사구들을 식별하고, 동일한 방식으로 처리가 가능하다.

3. 본론

3.1 개선된 Fillmore 의 Case 메커니즘

제안한 유스 케이스 추출 방법은 기존의 Fillmore 의 Case Grammar 의 의미화 개념을 기반으로 개선하여 요구공학에 적용한다[1,2]. 자연어로 작성된 고객 요구사항으로부터 유스 케이스를 추출하기 위하여 개선된 Fillmore 의 Case Grammar 모델링 절차를 사용한다. 이는 더 큰 단위인 유스 케이스 식별로부터 하향식(Top-down)방식으로 작은 단위의 객체 및 기능을 식별하고자 함이다. 이를 통하여 철저한 요구사항 분석으로 고품질의 소프트웨어 개발이 가능하다.

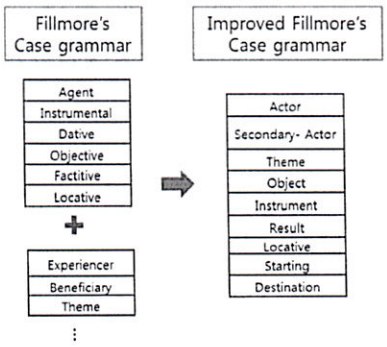


그림 1 개선된 Fillmore 의 Case 메커니즘 재 정의

그림 1 은 기존의 언어학이론인 Fillmore 의 Case Grammar 에서 유스 케이스 추출을 위한 UML 기반의 개선된 Fillmore 의 Case 메커니즘 변경사항을 보여준 것이다. 기존의 언어학 기반의 Fillmore 의 Case Grammar 는 6 개의 내면 격(1968)으로 정의하였다. 이밖에 다른 언어학자들에 의해 Case Grammar 는 계속적으로 수정되고 있다. 이를 UML 에 적용하기 위하여 필요한 격의 범주를 재 정의하였다. UML 기반의 개선된 Fillmore 의 Case 메커니즘은 총 9 가지로 재 정의하였다. 기존의 격인 동자주격(Agent)은 유스 케이스 다이어그램의 액터(Actor)와 같은 단어인 행위자격(Actor, A)으로 변경하였다. 기존 격인 여격(Dative)은 동반 격은 부차 액터의 이름을 인용하여 대 행위자격(Secondary Actor, SA)로 변경하였다. 기존 6 개의 격에서 원천 격(Source), 목적격(Goal), 주제격(Theme)이 추가되었다. 원천 격(Source)는 원천 격(Starting, S)으로, 목적격(Goal)은 목적격(Destination, D)으로 소프트웨어에서 사용되는 용어와의 혼돈을 피하기 위하여 용어를 재정의하였다. 작위 격(Factitive)는 후에 결과 격(Result, R)라고 변경된 용어를 그대로 사용한다. 대상격(Object, O), 주제격(Theme, T), 도구격(Instrument, I), 장소격(Locative, L)은 동일한 용어를 사용한다. 재 정의된 격의 범주들은 UML 의 유스 케이스 다이어그램 추출에 적용된다. Fillmore 의 Case Grammar 는 동사를 중심으로 의미론적으로 영향을 받는 논항을 식별하고 격을 부여한다.

논항의 역할	표기법	의미
행위자격(Actor)	A	행위 하는 주체
대 행위자격(Secondary Actor)	SA	실행되는 행동에 대항하는 힘이나 대상
대상격(Object)	O	상대로 하여금 행위가 수행되거나 변화를 겪는 대상
주제격(Theme)	T	행동에 의해서 변화되는 주체
도구격(Instrument)	I	사건의 원인에 사용되는 대상
원천격(Starting)	S	사람이나 물체가 움직임을 시작하는 위치
목적격(Destination):	D	사람이나 물체가 움직임을 끝마치는 최종장소
장소격(Locative)	L	동사가 나타내는 상태나 행동의 위치적 기점
결과격(Result)	R	행동의 결과로 존재하는 실체

그림 2 개선된 Fillmore 의 격(Case)범주

그림 2 는 개선된 Fillmore 의 격의 범주들과 그 표기법과 의미를 나타낸다.

이처럼, 개선된 Fillmore 의 Case Grammar 메커니즘의 격의 범주는 자연어로 작성된 고객요구사항을 분석하여 유스 케이스 추출을 위한 의미 분석의 기준이 된다. 각각의 동사에 관련된 논항(명사구)들의 격의 분류로 인하여 철저한 요구분석을 위한 유스 케이스 추출이 가능하다.

3.2 유스 케이스 추출 및 모델링

기존 use case formulating approach 절차처럼 1) name use case, 2) finds the actors, 3) concentrate on the flow of events, 4) use case association, 그리고 5) identify object from use case 단계와 use case 명세를 그대로 적용한다[7,8,9].

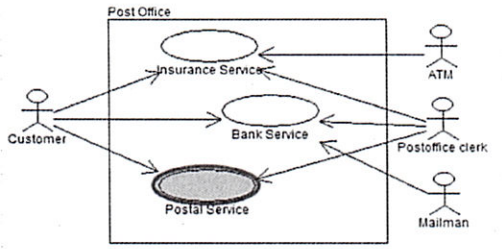
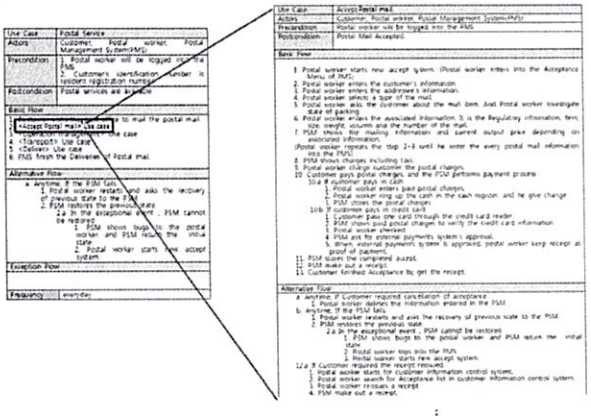


그림 3 우체국 High Level 유스 케이스

그림 3 은 우체국 시스템의 주요 업무이며 크게 3 가지이다. 주요 High Level 업무는 우편 서비스(Postal Service), 은행 서비스(Bank Service), 보험 서비스(Insurance Service)이다[2]. 본 논문에서는 상세 유스 케이스를 추출하는 절차를 언급한다.
이를 설명 하기 위해, 그림 3 에서 언급한 유스 케이스 중에 Postal_mail 유스 케이스의 상세(Low-Level) 유스 케이스인 Accept postal_mail 을 예제로 사용한다.



(a) High Level Use Case : *'Postal Service'*　　(b) Detailed Use Case : *'Accept postal_mail'*

그림 4 가상의 우체국 시스템 유스 케이스 명세서

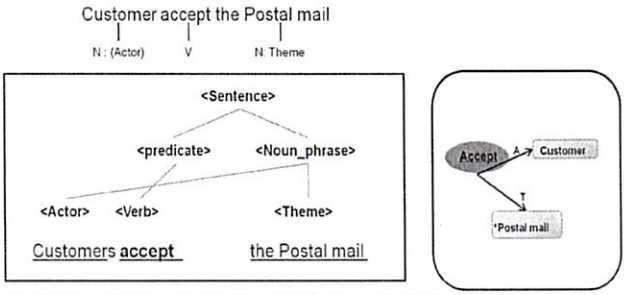
그림 4 는 가상의 우체국 시스템의 유스 케이스 명세서이다. 그림 3(b)의 Detailed Use Case 로 유스 케이스 추출과정을 보인다.

-Step 1. 동사들을 리스트화해 한 문단단위로 주된 동사를 식별

> Customer accept the Postal mail. Postal worker starts new accept system. that Postal worker enters into the Acceptance Menu of the Postal Management System(PMS). Postal worker enters the customer's information. Postal worker enters the addressee's information. Postal worker selects a type of the mail. Postal worker asks the customer about the mail item.
> ..... (The rest is omitted)

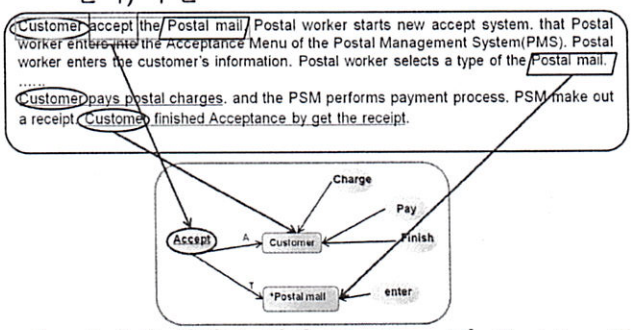
먼저, 유스 케이스 명세서의 동사들을 리스트화한다. 리스트화된 동사들 중 주된 동사(Main Verb)를 식별하여 그 동사가 포함된 문단을 추출한다.

-Step 2: 주된 동사의 특성에 따라 취할 수 있는 명사식별 및 그 명사의 역할 부여



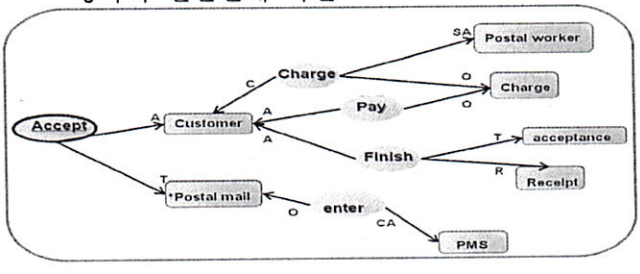
식별된 주된 동사의 문장을 개선된 Fillmore 의 Case 메커니즘에 적용하여 비쥬얼 모델링화 한다. 'Customers *accept* the Postal mail'이라는 문장에서 동사 *'accept'*의 영향을 받는 명사인 'Customer'와 'Postal mail'에 역할을 부여한다. 동사에 영향을 받지 않는 명사는 제외시킨다.

-Step 3: 식별된 각 명사들이 취하고 있는 동사(서술어) 추출

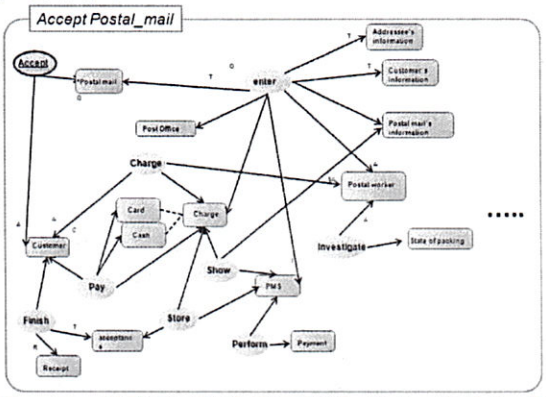


Step 2 에서 식별된 명사 'Customer'와 'Postal mail'이 포함된 문장들을 (문단 내에서) 식별한 뒤 각 명사들이 취하고 있는 동사들을 추출하여 비쥬얼 모델링 화한다.

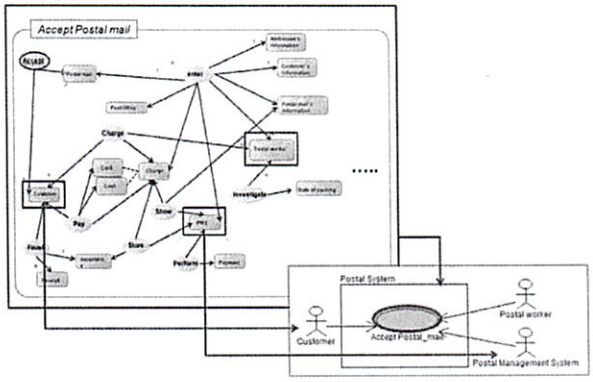
-Step 4: Step 2 과 같은 방법으로 추출된 동사와 명사의 연관관계 식별



Step 2 과 같은 방법으로 동사의 특성에 따라 취할 수 있는 명사들을 식별하고 그 명사에 역할을 부여한다. 영어는 동일한 단어사용보다는 유사한 의미의 다른 단어들을 바꾸어 사용하여 작성하기 때문에 같은 의미를 가진 단어를 의미적으로 분석하여 하나로 취급한다. Step 2 와 Step 3 의 과정을 문단이 끝날때까지 반복한다.



-Step 5: 연관된 다이어그램의 유스 케이스 추출



비쥬얼 모델링화로 묶인 하나의 덩어리는 주된 사 'Accept postal' 유스 케이스로 변환된다.
UML 의 유스 케이스 다이어그램의 액터(Actor)역할은 개선된 Fillmore 의 Case 메커니즘에서는 동작주격(Actor)와 대 행위자격(Secondary Actor)가 사용된다. 따라서 동자주격(A)와 대 행위자격(SA)의 명사의 역할을 부여 받은 명사인 'Postal Worker'와 'Postal Management System'은 액터로 변환된다.

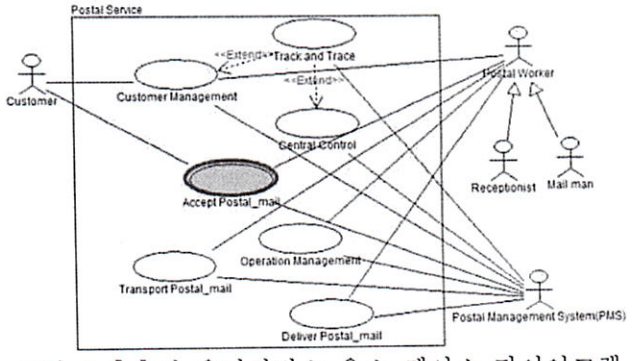


그림 5 추출된 우편서비스 유스 케이스 다이어그램

그림 5 는 유스 케이스 추출 및 모델링 알고리즘 프로세스를 통하여 얻은 결과값이다.

4. 결 론

제안하는 유스 케이스 추출 및 모델링 방법은 기존의 Fillmore 의 Case Grammar 를 개선하여 요구공학에 적용하고자 한다. 고객 요구사항으로부터 유스 케이스 추출 알고리즘을 제안한다. 현재의 소프트웨어가 대규모화, 복잡화, 분산화 됨에 따라, 작은 단위인 객체나 기능 보다는 더 큰 단위의 유스 케이스 식별에 초점을 두고자 한다.

개선된 Fillmore 의 Case Grammar 메카니즘으로 관련된 모든 사람들에게 정확하게 요구사항을 전달하고, 좀 더 쉽게 유스 케이스 다이어그램을 생성할 수 있다. 유스 케이스 추출 프로세스는, 고객으로부터 작성된 자연어 요구사항 기술서에 개선된 Fillmore 의 Case Grammar 를 적용하여 유스 케이스 추출 및 비쥬얼 모델링화가 가능 한다.

참고문헌

[1] 김보연, "개선된 Fillmore 의 Case 메커니즘 기반 고객요구사항에서 유스 케이스 추출방법", 석사학위논문, 홍익대학교, 2014.

[2] 김보연, 손현승, 서채연, 박병호, 김영철, "고객 요구사항 식별을 위한 Fillmore's Case grammar 을 적용한 유스 케이스 추출 방법", 정보과학회, pp524-526, 2013.11.

[3] Noam Chomsky, "Noam Chomsky on Nature and Language", Cambridge, 2002.

[4] Robins, R. H., "A Short History of Linguistics", Addison Wesley Longman, 1997.

[5] Walter Anthony Cook, "Case grammar theory", Georgetown University Press, 1989.

[6] Walter Anthony Cook, "Case Grammar Applied", Intl Academic Bookstore, 2008.

[7] Craig Larman, "Applying UML And Patterns", Prentice Hall,2004.

[8] Paul R. Reed Jr., Developing Applications with Java and UML, 2001.

[9] Bernd Bruegge & Allen H. Dutoit, Object-oriented Software Engineering: Using UML, Patterns, and Java