한국산학기술학회논문지
Vol. 12, No. 1, pp. 420-426, 2011

개념클래스 다이어그램 도출 시스템의 설계 및 구현

박가영[1], 이용훈[1*], 이상범[1]
[1]단국대학교 전자계산학과

Design and Implementation of A Conceptual Class Diagram Elicitation System

Ga-Young Park[1], Yong-Hun Lee[1*] and Sang-Bum Lee[1]
[1]**Dept of Computer Science, Dankook University**

요 약 본 논문에서는 다양한 클래스 도출 방법들을 통합하여 문제 기술서로부터 개념클래스를 추출을 도와주는 시스템을 소개하였다. 문제 기술서는 개발하고자하는 시스템에 대한 고객의 요구사항을 개략적으로 작성한 문서이다. 개발자는 이 문서를 바탕으로 문제영역에서 중요한 개념클래스를 도출할 수 있는데, 이것은 객체지향 분석 시에 생성되는 중요한 산출물이다. 지금까지 개념클래스 도출은 주로 개발자의 경험과 직관에 의존하는 경향이 있기 때문에 초보 개발자에게는 여러 가지 어려움이 있다. 따라서 개발할 시스템의 규모가 커지면 수작업으로 도출하는데 무리가 있다. 이러한 문제점 개선을 위해서 개념클래스 도출 도구를 구현하였는데, 이를 적용하면 빠르고 손쉽게 개념클래스 다이어그램을 구축할 수 있게 되었다.

Abstract In this paper, we introduce a system which helps to draw a conceptual class diagram from the problem description by combining various class diagram deriving methods. Generally, problem description is a kind of statements including user requirements in the early development phase. The system developer can derive a conceptual class diagram from this document, which playes an important role during the object-oriented software development. Until now, it is not easy for the novice to elicit classes because it requires good experience and intuition. In addition, there are also some difficulties of finding valid conceptual classes by hand when the size of system becomes larger. Therefore, we develop a system to solve these problems so that the developer is able to construct the conceptual class diagram easily.

Key Words : Conceptual class diagram, Object-oriented, Requirements specification, UML

1. 서론

고객의 요구사항을 만족하는 소프트웨어 개발은 여러 단계의 작업으로 이루어진다. 프로젝트의 규모가 커지고 기능이 복잡해짐에 따라 소프트웨어는 개발 비용 뿐만 아니라 유지보수 비용이 크게 증대되었기 때문에 이러한 비용을 최소화하기 위한 노력이 필요하게 되었다. 고객의 요구사항이 정확히 추출되지 않거나 모호하게 기술된다면 전체 프로젝트의 개발비가 약 30～50%가 추가적으로 소요된다고 알려져 있다[1]. 따라서 정확하게 요구사항을 파악하는 것이 중요하며, 산출물인 요구사항 명세서(requirements specification)는 다음 단계인 설계를 위한 바탕 자료로 활용될 뿐만 아니라 소프트웨어 테스팅을 위한 검증 기준으로 사용된다. 일반적으로 요구사항을 정의하기 위해 개발 초기 단계에서 작성되는 문제 기술서(problem description)는 개발될 시스템이 수행해야 할 기능들을 포괄적으로 기술한 문서로 비전문가도 쉽게 이해할 수 있는 자연어로 기술된다[2].

개발자는 이러한 문제 기술서와 고객과의 대화를 통해 요구사항을 파악하고 이를 유즈케이스 모델링을 통해 좀 더 정확하게 분석한 후, 클래스 및 기타 모델링을 통해 설계 모델링을 한다[3]. UML의 8개 모델 중에서 클래스

이 연구는 2009년도 단국대학교 대학연구비의 지원으로 연구되었음.
*교신저자 : 이용훈(karismanj@dankook.ac.kr)
접수일 10년 10월 21일 수정일 (1차 10년 12월 07일, 2차 10년 12월 30일) 게재확정일 11년 01월 13일

다이어그램은 정적인 관점에서 도식화한 것으로 소프트웨어 개발에 가장 중요한 역할을 담당하며, 여러 레벨의 다이어그램이 존재한다. 일반적으로 분석 단계에서는 개념과 분석 클래스 다이어그램을, 설계과정에서는 설계 클래스 다이어그램을 작성한다. 개념클래스 다이어그램은 문제 기술서와 요구사항 명세서를 근거로 작성되며, 개발할 시스템을 이해하기 위한 목적으로 활용된다. 개념클래스들을 기반으로 주변 클래스 (boundary class)와 컨트롤 클래스 (control class)를 추가하여 분석 클래스 다이어그램을 도출할 수 있으며, 인터페이스와 같이 구현이 필요한 클래스들을 추가하고 속성과 행위를 더욱 상세하게 명시하여 설계 클래스 다이어그램을 작성할 수 있다. 따라서 개념클래스 다이어그램이 시스템 개발 전체에 중요한 역할을 한다고 할 수 있다[4].

객체지향 개발 방법의 핵심이 되는 클래스 다이어그램을 작성하는 것은 결코 쉽지 않으며 또한 반복적이고 지루한 작업이기도 하다. 따라서 본 논문에서는 개념클래스 다이어그램을 쉽게 도출하기 위해 개발된 도구를 소개하고자 한다. 이 도구는 문제 기술서의 형태소 분석을 통해 명사를 추출하고 이를 클래스 목록법과 부적절한 클래스를 제거하는 소거법을 연속적으로 적용하여 적절한 클래스를 최종적으로 도출할 수 있는 기능을 갖고 있다. 2장에서는 본 연구의 관련 연구인 여러 클래스 도출 방법들을 소개하였으며, 3장에서는 개념클래스 다이어그램 도출 도구에 관해 자세히 소개하였다. 4장에서는 결론 및 향후 연구 과제에 대하여 언급하였다.

2. 관련연구

시스템 개발에 필요한 클래스를 찾고자 하는 방법이 여러 연구자에 의해서 제안되었으며, 숙련된 개발자도 경험에만 의존해서 찾아내기에는 한계가 있기 때문에 클래스 도출 방법들은 효과적이다. 하지만 대부분 방법들이 수동적이기 때문에 실제 프로젝트 적용에는 한계가 있다. 본 장에는 알려진 클래스 도출 방법들 몇 가지를 소개한다.

2.1 Case Grammar를 활용한 클래스 도출방법

논문 [5]에서 제안한 클래스 도출 및 모델링 방법은 Case Grammar[6] 방법을 통해 사용자의 요구 데이터를 자연어로 전환하고 이를 시나리오기반 분석방법[7]을 이용하여 클래스로 도출한다. 이 방법에서는 요구사항 식별을 위해 액티비트 다이어그램을 사용하며, 요구사항은 명세를 위해 유즈케이스로 모델링한 후 클래스 다이어그램을 작성한다. 제안하는 방법은 몇 가지의 다이어그램을 생성해야 하는 불편한 점이 있으며, UML 다이어그램에 능숙하지 않은 개발자는 클래스 도출에 어려움이 따른다. 하지만 행동분석방법을 통해 클래스를 도출하기 때문에 고객의 요구사항을 정확히 파악할 수 있으며 자세한 클래스 도출이 가능하다.

2.2 정보구조 모델링을 활용한 클래스 도출방법

논문 [8]에서는 고객의 이벤트를 기반으로 비즈니스 시스템을 분석하고, 정보구조 모델링에서 클래스를 도출하는 방법을 제안한다. 정보구조 모델링(information structure modeling)이란 이벤트/응답 중심(event/response -driven)의 방식으로 비즈니스 고객이 기업에 서비스를 요구하는 행위인 이벤트를 일으키고, 기업의 직원은 업무규칙과 업무지침서에 의해서 이벤트에 응답해주는 내부 활동과 이 활동의 대상이 되는 기업의 재산을 기반으로 분석하는 시스템 분석 모델이다[9]. 논문 [8]에서의 클래스 도출 과정은 먼저 컨텍스트 다이어그램에서 주요 사용자인 액터를 추출하고, 각 액터 별 이벤트에 대한 속성구조 다이어그램과 행위구조 다이어그램을 작성하고 정련하면서 객체 타입별로 객체를 도출하여 클래스로 정의한다.

제안한 방법은 비즈니스 관점에서 클래스를 도출하기 때문에 업무에 관한 지식이 필요하며, 업무 분석이 제대로 이루어져야 한다. 또한 속성구조와 행위구조를 통해 직관적으로 객체를 도출하므로 경험이 많은 개발자에게 유리한 방법이다. 하지만 정보구조 모델링은 시스템에 필요한 기능을 속성구조 다이어그램 작성을 통하여 한 눈에 쉽게 파악할 수 있기 때문에 복잡하게 시나리오를 작성하여 분석하지 않더라도 필요한 클래스를 도출할 수 있다.

본 논문에서에서 제시한 방법은 문제 기술서를 이용하는 반면 [5]는 고객의 행동 데이터를 이용하고 [8]은 고객의 요구 데이터를 이용하기 때문에 직접적인 비교 분석은 어렵다고 할 수 있다.

2.3 형태소 분석을 통한 클래스 추출방법

형태소 분석이란 자연어 분석의 첫 단계로써 단어(한국어의 경우 어절)를 구성하는 각각의 형태소들을 인식하고 불규칙 활용이나 축약, 탈락 현상이 일어난 경우 원형을 복원하는 과정을 말한다[10].

Abbott는 주어진 문제를 해결하기 위한 방법을 제안하였는데, 먼저 한 개의 구절을 쓴 후 이를 검토하는 방법으로 명사, 명사구, 대명사 등은 Ada 패키지의 후보가 되고, 동사들은 Ada 패키지에 들어가는 기능과 프로시저

후보들이 될 수 있다는 것이다[11]. 여기서 Ada 패키지는 각 객체의 정적인 구조와 기능을 함께 포함하여 그룹화 한 것으로 객체지향의 클래스와 유사함을 볼 수 있다. 이 방법을 통합하여 객체지향 설계 기법을 제안하였고, 클래스를 도출하는 하나의 방법이 되었다[12,13].

하지만 Abbott방법은 개발 프로젝트의 규모가 커지면서 단지 요구사항 명세서에서 명사와 동사만으로는 좋은 클래스를 도출 할 수 없다는 한계점이 생겼다. 그래서 본 논문에서는 Abbott 접근 방법 외에 다른 접근 방법을 통합한 클래스 도출 방법을 제안한다.

2.4 대상 클래스 목록을 이용한 방법

적절한 클래스의 대상이 되는 것을 목록화 시킨 방법을 Larman[14]이 제안하였다. 표 1은 총 클래스가 될 수 있는 18가지 대상들을 목록으로 나타낸 것이다. 개발자는 어떤 명사가 아래 표의 어떤 목록에 해당하는 것이면 클래스로 간주할 수 있기 때문에 편리함을 주지만, 이러한 것을 수작업으로 수행해야하기 때문에 어려움이 있다.

[표 1] 대상 클래스 목록

대상클래스 목록
1) 물리적 혹은 만질 수 있는 객체들
2) 사물에 대한 명세, 설계, 혹은 기술들
3) 장소
4) 트랜잭션
5) 트랜잭션 라인 아이템
6) 사람의 역할
7) 다른 것들을 담는 컨테이너
8) 컨테이너에 담긴 것들
9) 현 시스템 외부의 컴퓨터나 전자 기계적 시스템
10) 추상 명사 개념
11) 조직
12) 이벤트
13) 프로세스
14) 법칙과 정책
15) 카탈로그
16) 재정, 작업, 계약, 법률적 사항의 기록
17) 재무 도구와 서비스
18) 매뉴얼, 문서, 참고문헌, 책

2.5 부적절한 클래스 소거법

다른 형태를 가진 명사지만 중복된 의미를 가질 수 있으며, 클래스가 아닌 속성이나 연산인 명사가 있을 수 있다. 이 방법에서는 클래스로 부적절한 명사들을 소거함으로써 결과적으로 시스템에서 필요로 하는 클래스만 도출할 수가 있다. 소프트웨어 개발에 합당하지 않은 부적절한 클래스들에 대한 좀 더 구체적인 설명은 아래와 같다[15].

○ 시스템 외부에 존재하는 개체

추출한 개념 중에서 명사이고 클래스로 적절한 목록에 포함되어 있더라도 개발할 시스템 외부의 대상이나 존재를 뜻하는 경우에는 제외한다. '고객'이 일반적으로 클래스의 좋은 예가 되지만 만약 '고객'이 직접 시스템을 접근하는 경우가 아닌 경우에는 외부 개체로 분류된다.

○ 중복된 클래스

명사 중에는 서로 다른 형태를 가졌지만 동일한 대상이나 개념을 뜻할 수 있다. 예를 들면 편의점 시스템에 있어서 '손님'과 '고객'은 형태는 다르지만 동일한 대상을 뜻하고 있다. 이와 같이 동일한 대상과 개념을 뜻하는 명사 중에서 가장 구체적이고 명확한 이름을 가지는 개념을 선정하고 나머지 개념을 제외한다.

○ 시스템의 기능과 무관한 클래스

명사 추출 방법과 대상 클래스 목록 열거법에서 추출한 명사의 개념이 개발하는 시스템과 무관한 것은 제외한다.

○ 속성/연산

독립적인 개념을 나타내기 보다는 다른 대상과 개념의 의미를 상세화하거나 행위나 기능을 수행하는 역할을 하는 명사일 수 있다. 이런 명사는 개념클래스가 아니라 해당 속성 또는 연산으로 표현하므로 제외한다.

○ 구현과 관련된 클래스

시스템에 제공할 기능 측면 보다 이를 구현하기 위한 기술들은 제외한다. 예를 들면 'TCP/IP 통신'의 개념은 분석 단계에서는 고객이 이해하기 어렵고 이해할 필요 없는 개념이다. 이런 개념들은 의사소통에서 방해가 되므로 제외한다. 하지만 필요에 따라서는 향후 설계 클래스에 추가될 수 있다.

3. 개념클래스 도출 도구의 구현

2장에서 소개된 2.3, 2.4, 2.5 방법들은 각각 독립적으로 적용되어 사용하고 있을 뿐만 아니라 이를 지원하는 개발도구가 없다. 간단한 시스템 개발에는 수작업을 통해 클래스를 도출하고 이들 관계를 설정할 수 있지만, 규모

가 큰 시스템에는 이를 도와주는 도구의 개발이 절실하다고 볼 수 있다. 본 장에서는 2장에서 소개한 방법들을 통합하여 효율적으로 클래스를 도출하기 위해 구현한 도구를 소개하고자 한다. 그림 1은 도구의 전체적인 과정을 보여준다.

[그림 1] 개념클래스 도출 도구의 실행과정

실행 순서를 살펴보면, 먼저 문제 기술서를 입력하여 형태소 분석기를 통해 명사와 동사를 추출한다. 이 과정에서 추출된 명사나 명사구는 개념클래스 후보가 되고, 동사는 추후 클래스 간의 관계를 설정하는데 활용된다. 활용한 형태소 분석기는 서울대학교 지능형 데이터베이스 랩에서 개발한 '꼬꼬마 형태소분석기'이다[16].

그 다음 과정은 대상 클래스 목록을 활용하여 추출된 명사들 중 적절한 개념클래스 목록에 해당하는 명사들만을 선택하고, 선택되지 못한 명사들은 제외한다. 이 과정에서 클래스 선택은 개발자의 판단에 의해서 이루어진다. 왜냐하면 추출된 명사가 어떤 의미를 포함하고 있는지를 도구가 판단하기에는 아직 기능적으로 부족하기 때문이다.

마지막으로 소거법 과정을 적용하여 개념 클래스 목록에 포함된 명사들 중 부적절한 개념클래스에 해당하는 명사를 선택하여 제거하는데, 최종적으로 남아있는 명사들이 개념 클래스로 선택된다.

선택된 클래스들 간의 관계를 설정하는 과정이 중요한데, 개념클래스 다이어그램은 중요한 개념들과 그들 간의 관계를 이해하는데 목적이 있으므로 클래스 간의 구조적인 관계를 표현하기 위해선 가장 단순한 연관관계만을 사용한다. 집합관계, 의존관계 등과 같은 상세한 관계를 설정하는 것은 설계 클래스 다이어그램에서는 꼭 필요하지만 개념 클래스들 간에는 의미가 약하며, 또한 문제 기술서 내 문장에서 동사만으로 상세한 관계를 찾기 어렵다. 그 이유는 도구가 동사 의미를 분석하여 명사 간의 관계를 파악해야 하기 때문이다. 따라서 본 도구에서는 연관관계만을 설정하였으며, 추후 동사의 의미에 따른 다양한 관계를 목록으로 작성하고 이를 근거로 집합 및 의존 관계 등을 파악할 수 있도록 시스템을 개선할 계획이다.

본 도구에서 연관관계를 설정하는 방법은 다음과 같다.

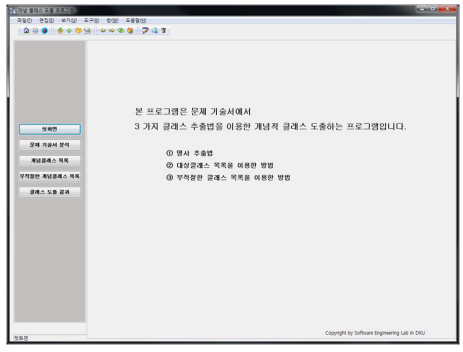
첫 번째, 형태소 분석을 통해 추출한 동사는 앞과 뒤의 명사를 파악하고 두 명사는 관계가 있다고 판단하여 설정한다.

두 번째, 한 문장에 있는 명사는 상호간에 관련이 있는 명사라고 판단하여 같은 문장번호를 가진 명사들의 관계를 설정한다.

여기서 소개한 개념클래스 도출 과정에서는 개발자의 개입이 불가피 하다. 대상 클래스 목록 또는 부적절한 클래스 목록에서 명사를 선택하는 과정을 자동화하기에는 거의 불가능하다고 할 수 있다. 컴퓨터를 통해 문장을 분해하는 것은 가능하나 각 단어가 의미를 갖는다는 것은 현실적으로 어려운 과정이다. 또한 시스템의 분석이나 설계 과정은 컴퓨터의 계산이 아닌 개발자의 경험과 사고에 의해서 결정되는 정확성에 문제가 있기 때문이다. 본 도구는 개념클래스를 도출하는 일련의 과정을 지원하며 분석 과정을 빠르게 진행하는 것에 목적을 둔다.

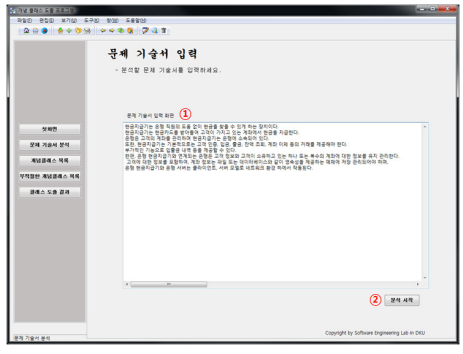
3.1 개념클래스 도출 도구의 주요 화면

본 논문에서 제안하는 개념클래스 도출 도구의 초기 화면은 그림 2와 같다.



[그림 2] 개념클래스 도출 도구의 첫 화면

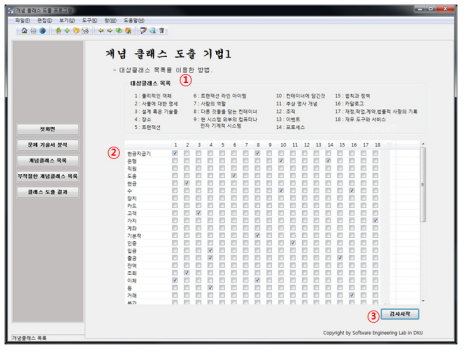
위 그림의 왼쪽에는 다른 메뉴로 이동하는 버튼들이 있고 위에는 툴바 메뉴들로 구성된다. 문제 기술서를 입력하는 화면은 그림 3과 같다.



[그림 3] 문제 기술서 입력 및 분석화면

위의 그림 3은 문제 기술서를 입력하는 창①과 '분석시작'이라는 버튼②로 구성된다. 문제 기술서를 입력한 후 '분석시작'이라는 버튼을 누르면 형태소 분석이 진행되어 명사와 동사를 추출한다.

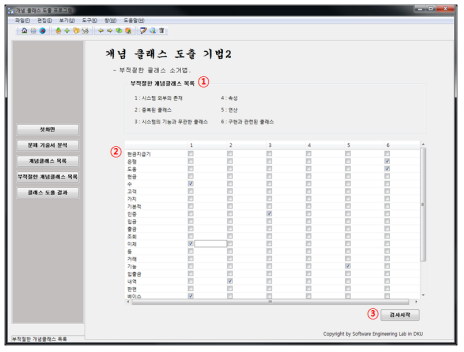
추출된 명사는 후보 클래스(candidate class)가 된다. 하지만 부적절한 클래스들이 포함되어 다음 단계인 개념클래스로 인정될 수 있는 목록 열거법을 이용하여 해당 명사를 도출하는 과정을 거친다. 그림 4가 이 과정을 수행하기 위한 화면이다.



[그림 4] 대상 클래스 목록을 이용한 열거법

대상 클래스 목록에는 18가지(①)가 있으며 해당하는 항목에 명사를 찾아 체크 박스(②)에 선택한다. 선택과정이 완료되면 '검사시작'이라는 버튼(③)을 누름으로써 선택된 명사만이 도출된다.

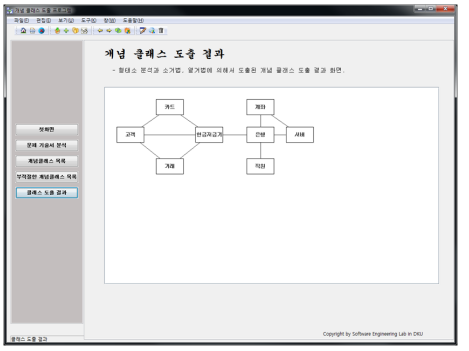
개념클래스 목록의 열거법을 적용한 후에 최종적으로 부적절한 클래스 소거법을 적용한다. 그림 5는 소거법을 적용하는 과정을 보여준다.



[그림 5] 부적절한 클래스 소거법을 적용

6가지 목록(①)에서 해당사항이 있는 명사를 체크박스(②)에 선택한다. 선택된 명사들은 개념클래스 후보에서 소거된다.

위에서 열거한 3가지 방법을 적용한 후에 최종적으로 남겨진 명사는 개념클래스가 된다. 최종 결정된 클래스들의 연관 관계는 명사들이 문장에서 동시에 출현할 경우 명사 간에 관계가 있는 것으로 판단하여 클래스 간 선으로 연결하여 연관관계가 있음을 표현한다. 그림 6은 최종적으로 선택된 개념클래스이며 클래스 간의 관계를 표현한 화면이다.



[그림 6] 최종 개념클래스 도출 결과 화면

3.2 은행 문제 기술서를 이용한 사례

본 절에서는 금융관련 문제 기술서를 제안한 도구에 적용한 결과를 소개한다. 먼저 표 2는 현금지급기 소프트웨어 개발을 위한 문제 기술서이다.

[표 2] ATM 문제 기술서 예

현금지급기는 은행 직원의 도움 없이 현금을 찾을 수 있게 하는 장치이다. 현금지급기는 현금카드를 받아들여 고객이 가지고 있는 계좌에서 현금을 지급한다. 은행은 고객의 계좌를 관리하며 현금지급기는 은행에 소속되어 있다. 또한, 현금지급기는 기본적으로는 고객 인증, 입금, 출금, 잔액 조회, 계좌 이체 등의 거래를 제공해야 한다. 부가적인 기능으로 입출금 내역 등을 제공할 수 있다. 한편, 은행 현금지급기와 연계되는 은행은 고객 정보와 고객이 소유하고 있는 하나 또는 복수의 계좌에 대한 정보를 유지 관리한다. 고객에 대한 정보를 포함하여, 계좌 정보는 파일 또는 데이터베이스와 같이 영속성을 제공하는 매체에 저장 관리되어야 하며, 은행 현금지급기와 은행 서버는 클라이언트, 서버 모델로 네트워크 환경 하에서 작동된다.

위의 은행 시스템의 문제 기술서는 일반 자연어로 쓰였으며 문법적 오류가 내포될 가능성이 있는 문서이다. 이 문제 기술서를 클래스 도출 방법의 첫 번째 방법인 명사추출법으로 추출한 결과가 표 3이다.

[표 3] 문제 기술서에서 추출된 명사

현금지급기, 은행, 직원, 도움, 현금, 수, 장치, 카드, 고객, 가지, 계좌, 기본적, 인증, 입금, 출금, 잔액, 조회, 이체, 등, 기능, 부가, 입출금, 내역, 한편, 정보, 소유, 하나, 복수, 유지, 파일, 데이터, 베이스, 영속성, 매체, 저장, 서버, 모델, 네트워크, 환경

형태소 분석의 중의성으로 인해 추출된 명사 중에는 잘못된 명사가 있다. 표 3에서도 '가지', '등', '부가', '기본적'과 같은 단어는 명사로 판단되지 않는 단어들이다.

추출된 명사들을 대상으로 두 번째 방법인 적절한 개념클래스 목록을 통한 열거법을 시행한다. 표 3의 명사들 중에서 표 1에 근거로 18가지 해당하는 클래스들을 도출한다. 표 4는 이러한 과정을 통해 선택된 명사들을 나타낸 것이다.

[표 4] 열거법 후 선택된 명사들

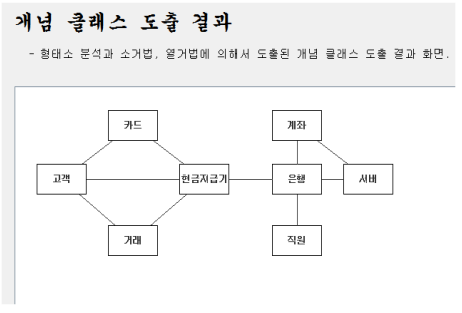
현금지급기, 은행, 직원, 현금, 장치, 카드, 고객, 계좌, 인증, 입금, 출금, 잔액, 조회, 이체, 거래, 파일, 데이터, 저장, 서버, 모델

마지막으로 적용한 것은 부적절한 개념클래스를 소거하는 방법이다. 이때 부적절한 클래스에 해당하는 명사들은 개념클래스에서 탈락된다. 표 5는 최종 소거법을 거친 후 남겨진 명사들이다.

[표 5] 최종 개념클래스의 명사들

현금지급기, 은행, 직원, 카드, 고객, 계좌, 거래, 서버

이와 같이 세 가지 도출 방법을 모두 적용한 후에 최종적으로 선택된 명사들은 개념클래스가 되며 그 결과로 표현한 것이 그림 7이다. 선택된 명사들의 동사 관계를 파악하여 클래스간의 연관관계를 선으로 연결하였다.



[그림 7] 최종 개념클래스 다이어그램 화면

결과 화면으로 8개의 개념클래스들이 선택되었으며 크게 2개의 군을 이룬 형태로 나타났다. '고객'이라는 관점에서 시스템을 보았을 때 '현금지급기', '카드', '거래'와 같은 개념이 필요할 것이며 그림 7에서 보는 것처럼 연관관계를 가지는 모습을 볼 수 있다. 또한 '은행'의 개념과 연관이 있는 것들은 '서버', '직원', '계좌'가 나타났으며 은행이라는 관점에서 연관을 가지는 개념들이다. 두 개의 군은 '현금지급기'와 '은행'이 연결되면서 서로의 연관 관계가 표현되었다.

비록 시스템 구축에 있어서 완벽한 모습을 갖춘 클래스를 도출하지는 못하였지만 기본이 되는 개념클래스와 이들 간의 연관관계로 어느 정도 표현 가능한 결과를 얻었다.

결론적으로 본 논문에 구현한 개념클래스 도출 도구를 이용하여 금융 관련 문제 기술서를 분석한 결과 어느 정도 시스템 구축에 필요한 개념클래스를 찾을 수 있었다. 이를 기본으로 활용하고 더 많은 문서를 분석한 후에 설계 과정에서 필요한 클래스를 추가하여 최종적인 설계 클래스 다이어그램을 만들어가는 과정이 필요하다. 따라서 여기서 소개한 도구는 분석 과정에서 필요한 개념클래스를 제공하여 다음 설계 과정을 도울 수 있을 것이다.

4. 결론 및 향후 연구과제

본 연구에서는 기존의 개념클래스 도출 방법인 명사 추출 방법과 대상 클래스 목록을 이용한 열거법, 부적절한 클래스 소거법을 적용하여 손쉽게 개념클래스 다이어그램을 작성할 수 있는 시스템을 소개하였다. 여러 가지 방법을 통합하여 이용함으로써 클래스 도출에 있어 범하기 쉬운 오류들을 최소화하였고 초보 개발자에게는 가이드라인을 제공하였다.

개발한 시스템을 통해 도출한 과정은 개발자의 판단이 어느 정도 필요하기 때문에 자동적인 작업은 어렵다. 또한 자연어는 중의성을 가지고 있어서 완벽한 형태소 분석은 어렵다. 그러므로 개발자의 관여가 필요하며 문제 기술서에서 나타나 있지 않은 개념클래스에 대해서도 주의를 기울여야 한다.

따라서 향후 연구과제로는 형태소분석에서 비롯되는 중의성을 개선하는 방안을 연구해야 할 것이며 열거법과 소거법 과정에서 개발자의 개입을 최소화하더라도 일관된 개념클래스를 도출하는 방안에 대하여 연구되어야 할 것이다. 이를 위해서는 적용할 분야에 대한 단어들의 온톨로지를 구축하여 각 단어들에 대한 의미를 좀 더 파악하여 집합관계, 의존관계 등의 다양한 관계를 확장해 나아가는 연구가 활발하게 진행된다면, 훨씬 더 유용한 시

스템으로 발전할 수 있으리라 기대한다.

참고문헌

[1] B. Boehm, and P. Philip, "Understanding and Controlling Software Costs," IEEE Transactions on Software Engineering, Vol.14, No.10, pp. 1462-1476, Oct, 1988.

[2] 신종철, “요구사항 관리범위 확대를 위한 명세화 개선방안”, 한국 OA학회논문지, 제6권, 제4호, pp. 30-37, 12월, 2001.

[3] Roger Pressman, "Software Engineering : A Practitioner's Approach", Seventh Editon: McGraw-Hill Science, 2009.

[4] 박현철, “UML 이해와 실제”, 한국소프트웨어연구원, 2005.

[5] 안성빈, 김동호, 서채연, 김영철, “Fillmore의 Case Grammar를 통한 사용자 요구사항으로부터 객체 추출 및 모델링 방법”, 한국정보과학회논문지, 제16권, 제10호, pp. 985-989, 10월, 2010.

[6] Fillmore, Ch. J, "Some Problems for Case Grammar", In: Georgetown University Round Table on Language and Linguistics, Hrsg. R.J.O' Brien. Washinon.

[7] S.D. Ahn, C.S. Kim, Y.R.Kim, "Study On Need Extraction Method Based On User Behavior Analysis," Proc. of 2010 Korea Conference on Software Engineering, vol.12, no.1, pp.413-418, 2010. (in Korean)

[8] 이혜선, 박재년, “사용자 이벤트 기반의 정보구조 모델링을 이용한 비즈니스 업무 분석에서의 클래스 추출 방법”, 정보처리학회논문지, 제12-D권, 제7호, pp.1071-1078, 12월, 2005.

[9] 박재년, “정보 구조 모델링에 의한 시스템 분석”, 숙명여자대학교 논문집, 제33집, 1992

[10] 강승식, “음절 특성을 이용한 한국어 불규칙 용언의 형태소 분석”, 정보과학회논문지, 제22권, 제10호, pp. 1480-1487, 10월, 1995.

[11] Abbott, R, "Report on Teaching Ada", Technical Report SAI-81-313-WA, Science Applications, Inc, Dec, 1980.

[12] Abbott, R. "Program Design by Informal English Descriptions", Communications of the ACM vol. 26(11), 882-894, Nov, 1983.

[13] Nik Boyd. "Using Natural Language in Software Development", http://www.educery.com/papers/rhetoric/road.

[14] C. Larman, "Applying UML and Patterns An Introduction to Object-Oriented Analysis and Design", Second Edition: Prentice Hall, 2003.

[15] 채흥석, “클래스 구조의 이해와 설계: UML, Java, C++를 활용한 객체지향 모델링 실전”, 한빛미디어, 2004.

[16] 황인법, 이동주, 연종흠, 이상구, “웹의 협업 환경을 이용한 확장 형태소 사전 관리”, 한국컴퓨터종합학술대회 논문집, 제37권, 제1호, pp. 94-95, 1월, 2004.

박 가 영(Ga-Young Park) **[준회원]**

- 2009년 8월 : 단국대학교 컴퓨터과학과 (공학사)
- 2010년 3월 ~ 현재 : 단국대학교 전자계산학과 석사과정

<관심분야>
소프트웨어공학

이 용 훈(Yong-Hoon Lee) **[준회원]**

- 2009년 2월 : 단국대학교 컴퓨터과학과 (공학사)
- 2009년 3월 ~ 현재 : 단국대학교 전자계산학과 석사과정

<관심분야>
정보 검색, 데이터 마이닝, 기계학습

이 상 범(Sang-Bum Lee) **[정회원]**

- 1989년 12월 : 루우지애나주립대 (전산학석사)
- 1992년 8월 : 루우지애나주립대 (전산학박사)
- 1992년 9월 ~ 1993년 10월 : 전자통신연구원 선임연구원
- 1993년 10월 ~ 현재 : 단국대학교 교수

<관심분야>
소프트웨어공학, 정보검색, 모바일컴퓨팅