The Journal of The Institute of Internet, Broadcasting and Communication (IIBC)
Vol. 16, No. 1, pp.263-271, Feb. 29, 2016. pISSN 2289-0238, eISSN 2289-0246

http://dx.doi.org/10.7236/JIIBC.2016.16.1.263
JIIBC 2016-1-36

요구사항 시나리오 기계 학습을 이용한 자동 소프트웨어 요구사항 패턴 추출 기법

Automatic Software Requirement Pattern Extraction Method Using Machine Learning of Requirement Scenario

고덕윤*, 박수용*, 김순태**, 유희경***, 황만수****

Deokyoon Ko*, Sooyong Park*, Suntae Kim, Hee-Kyung Yoo***, Mansoo Hwang****

요 약 소프트웨어 요구사항 분석은 성공적인 소프트웨어 프로젝트를 위해 필수적 요소이다. 특히 불완전한 요구사항은 소프트웨어 프로젝트 실패의 가장 큰 원인으로 꼽힌다. 불완전한 요구사항은 소프트웨어 개발 시 개발자에게 이해 부족을 야기할 뿐 아니라, 소프트웨어 검증 시 에도 모호한 기준을 제공함으로써, 개발 후반부에 납기일 연기 및 비용 증가의 원인이 된다. 요구사항 패턴은 이러한 문제를 극복하는데 도움을 줄 수 있다. 요구사항 패턴은 요구사항 작성과 검토 시 참조모델이 될 뿐 아니라, 검증 기준이 될 수 있으며, 작성자가 누락한 부분을 보완해 줄 수 있다. 이와 더불어 요구사항 작성의 경험이 적은 작성자는 요구사항 패턴을 통해 더 쉽고, 빠르고 정확하게 요구사항을 작성할 수 있다. 본 논문에서는 다양한 요구사항의 시나리오를 통해 공통된 요구사항 시나리오를 추출하는 기법을 제안한다. 그리고 제안한 기법의 가시성 검증을 위해 여덟 개의 프로젝트에서 추출한 83개의 소프트웨어 시나리오를 통해 54개의 시나리오 패턴을 추출하고 이 패턴을 이용하여 누락된 행위를 찾는 과정을 사례연구를 통해 보여준다.

Abstract Software requirement analysis is necessary for successful software development project. Specially, incomplete requirement is the most influential causes of software project failure. Incomplete requirement can bring late delay and over budget because of the misunderstanding and ambiguous criteria for project validation. Software requirement patterns can help writing more complete requirement. These can be a reference model and standards when author writing or validating software requirement. Furthermore, when a novice writes the software scenario, the requirement patterns can be one of the guideline. In this paper proposes an automatic approach to identifying software scenario patterns from various software scenarios. In this paper, we gathered 83 scenarios from eight industrial systems, and show how to extract 54 scenario patterns and how to find omitted action of the scenario using extracted patterns for the feasibility of the approach.

Key Words : Software Requirement, Requirement Pattern, Use Case Pattern

*정회원, 서강대학교 컴퓨터공학과
**정회원, 전북대학교 소프트웨어공학과
***정회원, 강원대학교 컴퓨터공학과
****정회원, 신한대학교 IT융합공학부
접수일자 2015년 11월 10일, 수정완료 2016년 1월 5일
게재확정일자 2016년 2월 5일

Received: 10 November, 2015 / Revised: 5 January, 2016 / Accepted: 5 February, 2016
****Corresponding Author: mshwang@shinhan.ac.kr
School of IT Convergence Eng., Shinhan University, Korea

Ⅰ. 서 론

소프트웨어 요구사항 분석은 소프트웨어 개발 프로젝트에서 가장 초기에 수행하고 그 산출물은 소프트웨어 프로젝트 전 분야에서 지침으로 사용된다. 따라서 이 단계에서 정의된 요구사항은 이 후 개발 산출물에 큰 영향을 끼칠 뿐 아니라, 소프트웨어 프로젝트의 성패를 좌우한다. 요구사항의 품질은 완전성, 정확성, 타당성, 검증가능성 등으로 측정된다. 이 중 완전성은 소프트웨어에서 제공할 기능 전체를 다루어야 하는 것을 의미하며, 가장 주요한 품질 속성 중 하나이다[1]. 한 보고서에 따르면 소프트웨어 프로젝트 실패의 가장 큰 원인으로 불완전한 요구사항을 꼽고 있다[2].

앞에서 언급한 바와 같이, 완전한 요구사항을 고객의 요구사항을 빠짐없이 기록하는 것을 말하는데, 이는 즉 소프트웨어의 기능이 모두 요구사항에 서술되어야 함을 의미한다[1]. 완전한 요구사항 작성은 매우 어려우며, 많은 경험을 필요로 한다. 특히 요구사항을 써 본 경험이 적은 초심자는 작성한 요구사항 시나리오에서 사용자와 시스템의 행위나 예외사항 등을 누락시킬 가능성이 있다[15]. 요구사항 패턴은 이러한 한계를 극복하여 작성자의 완성도 있는 요구사항 작성을 도와준다. 그리고 요구사항 패턴은 요구사항 검증의 기준이 될 뿐 아니라, 요구사항 작성 경험이 없는 초심자에게 좋은 가이드라인이 된다[12]. 패턴을 통한 요구공학은 아래와 같은 장점이 있다[13][14].

무엇이 누락되었는지 찾을 수 있다.

요구사항을 더 쉽게 작성할 수 있다.

요구사항을 재사용할 수 있다.

다양한 연구에서 요구사항 시나리오 패턴을 다루고 있다. A. Mahfouz 외[4]는 서비스 기반 컴퓨팅 분야의 요구사항 패턴을 정의하였고, R. Biddle 외[5]는 필수 유스케이스(essential use case)에서 사용자 시스템 간의 대화 패턴을 발표하였다. X. Franch 외[16]은 FABRE라는 프레임워크를 제안하고, 요구사항 패턴의 개념과 구조, 메타모델 등을 정의하였다. S.Robertson[8]은 다양한 이벤트-유즈 케이스로부터 공통 이벤트-유즈 케이스를 추출하는 방안을 제시하였다. S. Ketabchi 외[6]는 놈(Norm)으로 불리는 공통적으로 사용되는 도메인 비즈니스 규칙 도메인 비즈니스 규칙을 제안한 바 있다.

본 논문에서는 요구사항 시나리오 문장의 동사를 이용하여, 동사의 의미적 유사도에 의한 기계 학습(machine learning)기반의 요구사항 시나리오 패턴 추출 기법을 제안한다. 첫째, 요구사항 사나리오의 문장을 분석하여 문장으로부터 본동사를 추출한다. 둘째, 추출된 동사를 사용 빈도에 따라 분류하고, 자주 사용되는 동사는 단어 유사도 기법을 통해 가까운 동사를 군집화(clustering)한다. 셋째, 이들 군집화를 하나의 방향성 있는 그래프로 표현하고, 각 군집 사이의 가중치를 구한다. 넷째, 작성된 그래프를 실제 시나리오의 흐름과 비교하여 분리 추출하고, 사용빈도나 노드간의 응집도, 패턴의 길이에 유사도에 따라 패턴의 점수를 계산한다.

본 논문은 다음과 같이 구성된다. 2장에서는 요구사항 패턴에 관한 관련연구를 기술한다. 3장에서는 본 논문에서 제안하는 시나리오 패턴 추출 기법을 설명하고, 4장에서는 사례연구의 결과를 기술한다. 마지막으로 5장에서 결론을 맺는다.

Ⅱ. 관련연구

이 장에서는 소프트웨어 요구사항 패턴에 관련된 연구를 기술한다. 관련연구는 두 가지로 분류될 수 있는데, 첫째는 소프트웨어 요구사항 패턴을 정의하고 소개하는 연구이고, 둘째는 소프트웨어 요구사항 패턴을 식별하는 기법을 제안한 연구이다.

1. 소프트웨어 요구사항 패턴 제안 연구

A. Mahfouz 외[4]의 연구에서는 서비스 지향 컴퓨팅(service-oriented computing) 시스템에서 발생할 수 있는 공통 요구사항을 패턴으로 제안하였다. 이 연구에서는 열 한 개의 요구사항 패턴을 정의하였다. 예를 들어 배리어(Barrier) 패턴은 소프트웨어 실행 시 사전 정의된 행위를 보호해야 하는 요구사항을 의미하고, 토큰(Token) 패턴은 소프트웨어 서비스의 행위에 앞서 승인을 얻는 요구사항을 말한다. 이 논문에서는 열 한 개의 패턴 뿐 아니라 각 패턴간의 관계를 제공한다. 배리어 패턴과 토큰 패턴은 '구성 가능한'(may compose) 관계이다. 이는 배리어 패턴은 토큰 패턴에 의해 제약받을 수 있음을 의미한다. 이 관계를 통해 열 한 개의 패턴에서 발생 가능한 상호작용을 명시하고 개략적인 패턴 적용 방안을 제안하고 있다.

R. Biddle 외[5]의 연구에서는 필수 유스케이스(essential use case)에서 발생 가능한 대화 형태의 요구사항 패턴을 제안하였다. 이 논문에서 제안한 패턴은 총 여섯 개 이고 그 내용은 아래와 같다.

- 요청(Request) : 사용자가 시스템에 정보를 요구하거나 시스템이 사용자에게 정보를 요청하는 경우
- 모니터(Monitor) : 시스템이 시스템의 상태를 사용자에게 보여주는 경우
- 알람(Alarm) : 시스템이 사용자에게 경고하는 경우
- 명령(Command) : 사용자가 시스템에 정보 변경을 요청하고 시스템은 요청을 수행하는 경우
- 대기(Prompt) : 시스템이 사용자에게 명령 입력상태를 제공하고 사용자의 요청을 기다리는 경우
- 결정(Confirm) : 정보가 변경되면, 시스템이 사용자에게 최종 결정을 요청하는 경우

X. Franch 외[16]는 PABRE라는 요구사항 패턴 프레임워크를 제안하였다. 이 프레임워크에는 요구사항 패턴의 개념과 구조, 메타모델과 함께 29개의 기능 요구사항과 37개의 비기능 요구사항의 카탈로그를 소개한다. 이들은 소개된 패턴 카탈로그를 통해 새로운 요구사항을 정의하거나 분석할 때 재사용되어 사용될 수 있다고 주장한다.

위의 세 연구들이 제안한 요구사항 패턴은 논문 저자의 경험적 직관에서 식별되었다. 따라서 해당 패턴들의 적용 가능 범위나 정확성의 판단이 어렵다는 한계점이 있다.

2. 요구사항 패턴 추출 기법에 관한 연구

요구사항 패턴을 추출하는 기법에 관한 연구는 두 가지가 있다. S. Robertson[8]은 이벤트/유스케이스(event/use case) 모델들로부터 공통 요구사항을 추출하는 기법을 제안하였다. 이 연구에서는 유스케이스가 특정 이벤트로부터 시작한다고 가정하고, 여러 이벤트/유스케이스 모델로부터 유사한 이벤트/유스케이스를 추출하여 해당 도메인에 공통적으로 사용되는 이벤트/유스케이스 모델을 추출하는 방법을 소개한다. 이 연구에서는 서점 관리 소프트웨어와 컨설팅 서비스 관리 소프트웨어를 사례로 들고 있는데, 서점관리 소프트웨어의 “책 구매” 유스케이스는 컨설팅 서비스 관리 소프트웨어의 “서비스 구매”와 그룹화 되고, 이를 추상화하여 “고객은 제품을 구매한다”라는 공통 유스케이스로 추출된다.

S.Ketabchi 외[6]는 도메인 모델로부터 요구사항 패턴을 추출하는 프로세스를 제안하였다. 이 프로세스는 먼저 도메인 모델로부터 문제 패턴(problem pattern)을 추출하여 저장소에 저장한다. 이 문제 패턴은 공통 도메인에서 사용하는 공통 비즈니스 규칙을 추출하는데 이 규칙은 비즈니스 규정, 상태, 제한사항들을 포함한다. 이 비즈니스 규칙은 놈(norm)의 형태로 표현되고, 이는 유사한 비즈니스 도메인에서 재사용될 수 있다. 요구사항 패턴을 식별하기 위해 도메인 분석가는 먼저 고객 모델(stakeholder model)과 프로세스 모델(process model)을 식별한다. 고객모델은 고객을 분류하고 그들의 역할을 나타내고, 프로세스 모델은 고객 모델을 통해 시스템과 고객 사이에 상호작용을 분석한다. 분석된 결과는 각 역할별 고객들에게 해당하는 기능 요구사항과 비기능 요구사항을 분류하여 놈(norm) 저장소에 저장한다.

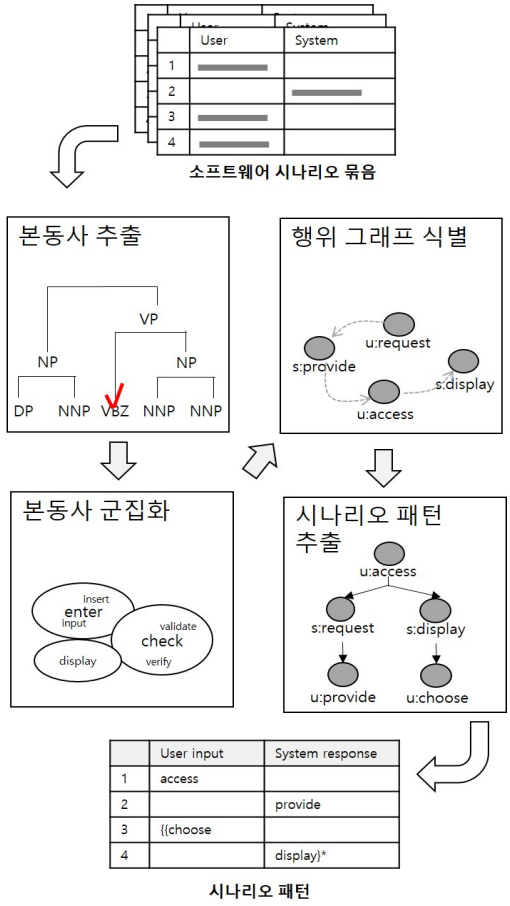

그림 1. 시나리오 패턴 추출 기법 개요
Fig. 1. Overview of pattern extracting approach

앞에서 언급한 연구들은 모두 구체적은 기법 보다는 개념적 절차를 제시하고 있으며, 적용하기 위해서는 전문가의 판단을 필요로 한다. 그리고 후자 연구의 경우 도메인 모델의 품질 수준에 따라 패턴의 품질도 결정되는 문제와, 추출된 패턴의 적용 도메인이 한정적인 한계점이 있다. 따라서 본 논문에서는 완전히 자동화 되고, 도메인에 제한 없이 사용할 수 있는 시나리오 기반의 요구사항 패턴 추출 기법을 제안하고자 한다.

Ⅲ. 요구사항 시나리오 패턴 추출 기법

본 논문에서는 소프트웨어 요구사항 시나리오들로부터 자동으로 패턴을 추출하는 기법을 제안한다. 그림 1은 본 논문이 제안하는 기법의 개요를 보여준다. 본 논문에서 제안하는 기법은 사용자와 시스템의 상호 대화로 이루어진 소프트웨어 시나리오 묶음을 입력으로 한다. 첫 번째 단계에서는 각 시나리오의 문장에서 본동사를 추출하고, 두 번째 단계에서는 이 각 시나리오의 본동사들을 의미상 유사한 동사들로 군집화 한다. 세 번째 단계에서는 군집화된 모든 동사를 가능한 모든 행위의 흐름을 갖는 방향성 있는 행위 그래프로 표현하고 네 번째 단계에서는 실제 시나리오와 행위 그래프를 바탕으로 자주 사용되는 시나리오의 흐름을 파악한다. 네 단계를 거치면 여러 시스템에서 공통적으로 사용되는 시나리오의 패턴을 추출할 수 있다.

1. 본동사 추출

이 단계에서는 입력된 여러 시나리오의 각 문장에서 본동사를 추출하는 것을 목적으로 한다. 시나리오 문장에서는 두 개 이상의 동사가 쓰일 수 있다. 본 논문에서 제안하는 기법은 본동사를 통해 행위를 추출하고 이를 통해 패턴을 식별하는 것을 목적으로 하므로, 시나리오 문장에서 본동사를 추출할 수 있어야 한다. 본동사 추출을 위해 자연어 처리 파서(natural language parser)를 이용한다. 각 문장을 파서를 통해 분석하면 그림 2와 같이 트리 형태의 결과가 나오는데, 트리를 루트 노드부터 탐색하는 도중 동사구에 해당하는 'V'로 시작하는 노드의 자식 노드를 따라가면, 본동사를 추출할 수 있다. 그림 2의 문장에서는 파싱 트리에서 루트 'S'로부터 'V' 노드를 탐색하면, 'VP-VBZ'를 통하여 문장의 본동사인 'request'를 식별할 수 있다. 한글 시나리오의 경우 사용할 만한 문장 파서 라이브러리가 존재하지 않아, 부득이하게 영어로 번역하여 추출하여야 한다. 만약, 한글 문장 파서 라이브러리가 있다면, 한글 시나리오도 분석이 가능하다.

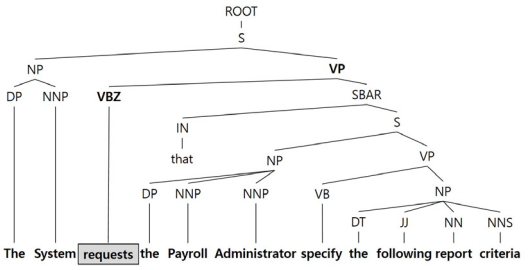

그림 2. 본동사 추출의 예
Fig. 2. The example of extracting main verb of sentence

2. 본동사 군집화

이 단계에서는 전 단계에서 추출한 본동사들을 단어간의 거리를 통해 의미상 거리가 가까운 것을 그룹화 하여 군집화 한다. 그림 3은 이 단계의 내용을 개괄적으로 보여준다. 먼저 동사를 주어에 따라 '사용자' 그룹과 '시스템' 그룹으로 분류한다. 그리고 각 분류에서 특정 기준점 이상의 발생 빈도를 보이는 '높은 빈도 동사'와 그 이하의 빈도를 보이는 '낮은 빈도 동사' 그룹을 각각 분류한다. 그리고 '높은 빈도 동사' 에서 동의어와 유의어를 묶어서 군집을 형성한다. 마지막으로 '낮은 빈도 동사'들은 각 군집의 모든 동사들과 거리를 측정하여 평균 거리가 가장 가까운 즉, 가장 유사도가 높은 동사 군집에 병합된다.

본 연구에서는 동사의 거리를 측정하기 위해 Resnik[9]이 제안한 워드넷(Word-net)[7] 기반의 거리 측정법을 이용한다. 워드넷은 영어의 단어를 의미적으로 분류한 거대한 단어 트리를 말한다. Resnik은 거리를 구하고자 하는 두 단어를 워드넷 트리에서 검색하고 한 단어에서 다른 단어를 찾는데 걸리는 최소의 거쳐 가는 노드의 수를 통해 단어 간의 거리를 계산하는 방법을 제안하였다. 본 연구에서는 이를 응용하여 한 단어에서 쓰이는 여러 의미(sense)와 각 의미의 사용빈도를 통해 수식 (1)의 방법으로 단어의 유사도(SIMilarity)를 측정한다. 이 수식을 통해 v_x, v_y 두 동사의 여러 의미 중 발생 비율과 의미적

유사도를 곱한 값 중 가장 큰 값을 둘 간의 유사도로 산정한다. 해당 식에서 n_x는 v_x의 총 의미 수를 뜻하며, freq(sense)는 전체 의미의 발생 빈도 중 해당 의미의 발생 비율을 뜻한다. similarity($sense_1$, $sense_2$)는 두 의미의 유사도를 말하며, 이는 앞에서 언급한 바 있는 Resnik[9]의 방법을 채택한다. 한글 요구사항 시나리오의 경우 워드넷 적용이 불가능하므로, 영어로 번역하여 사용하여야 한다. 만약, 적용 가능한 한글 동사의 거리 측정 기법이 있다면, 본 기법의 적용이 가능하다.

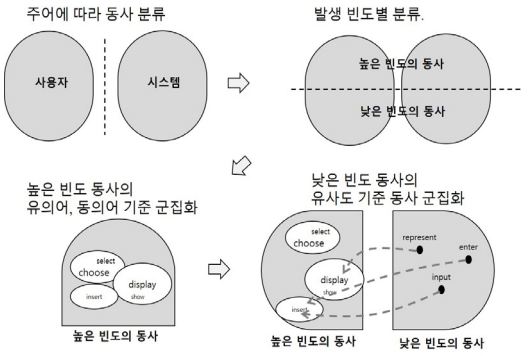

그림 3. 동사 군집화 과정
Fig. 3. The process of verb clustering

$$sim(v_x, v_y) = max_{\ 1 \le i \le n_x, 1 \le j \le n_y} (\frac{1}{2}(freq(s_{1i}) + freq(s_{2j})) \times similarity(s_{1i}, s_{2j})) \quad (1)$$

3. 행위 그래프 식별

이 단계는 이전 단계에서 식별된 군집의 관계를 추출하고 이를 방향성 있는 그래프 형태인 행위 그래프(action graph)로 표현하는 것을 목적으로 한다. 행위 그래프는 동사의 군집을 나타내는 노드와 해당 노드의 발생 순서를 나타내는 벡터로 구성된다. 먼저 모든 동사 군집을 노드로 배치한다. 그리고 입력된 모든 시나리오의 모든 문장을 동사 군집의 대표 동사로 치환하고, 시나리오의 순서대로 노드를 방향성 있는 벡터로 연결한다. 그 결과 하나의 큰 방향성 있는 그래프가 완성된다.

$$MF_{v_x} = \frac{occur.(v_x)}{\max_{0 \le i \le n}(occur.(i))} \quad (2)$$

이 그래프의 각 벡터는 가중치를 갖는다. 가중치가 높은 벡터일수록 패턴의 사용 빈도와 벡터가 연결하는 두 노드간의 응집성이 높고, 이는 상대적으로 더 가치 있는 벡터가 된다. 두 노드가 전체 시나리오에서 발생하는 빈도를 빈도 점수로 산정하고, 두 노드간의 응집력은 상대적 상호연결성으로 산정한다. 수식 (2)는 벡터 v_x의 빈도 점수(MF: Maximum based Frequency)를 구하는 방법을 나타낸다. 빈도 점수는 가장 높은 발생횟수를 갖는 벡터의 횟수에서 해당 벡터의 발생횟수의 비율로 구한다. 즉 전체 그래프에서 발생빈도가 가장 높은 벡터의 발생 횟수가 10이고, 구하고자 하는 벡터의 발생 횟수 2인 경우 0.2점의 빈도 점수를 획득한다. 수식에서 occur.(v_x)은 v_x의 발생 횟수를 의미한다. 여기서 구해진 빈도 점수가 일정 점수에 미치지 못하면, 해당 벡터는 그래프에서 제외된다. 수식 (3)은 두 노드간의 응집력을 구하는 방법인 상대적 상호 연결성(relative interconnectivity)[10]을 산정하는 방법을 보여주며, 그림 4는 상대적 상호 연결성의 이해를 위한 개요를 나타낸다. 동사A 노드로부터 동사B 노드로 가는 벡터의 경우 동사 A에서부터 출발하는 전체 빈도와, 동사 B로 들어가는 전체 빈도 중 동사 A에서 동사 B로 가는 빈도로 산정한다. 가중치를 갖는 행위 흐름 그래프를 추출함으로써 본 단계는 종료된다. 이 수식에서 EC(A,B)는 A에서 B로 가는 벡터의 발생 횟수를, EO(A), EI(B)는 각각 A에서 발산하는 벡터의 총 발생 횟수($a\text{-}out_1$ + $a\text{-}out_2$ + $a\text{-}out_3$. + ab)와 B로 수렴하는 벡터의 총 발생 횟수($b\text{-}in_1$ + $b\text{-}in_2$ + ab)를 의미한다.

$$RI = \frac{EC(A,B)}{\frac{1}{2}(EO(A) + EI(B))} \quad (3)$$

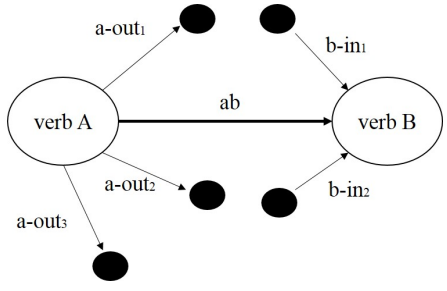

그림 4. 상대적 상호연결성의 이해도
Fig. 4. Conceptual figure for RI

4. 시나리오 패턴 추출

이 단계에서는 이전 단계에서 추출된 행위 흐름 그래프와 실제 시나리오에서 사용되는 정도에 따라 적용된 실제 패턴을 추출하고 각 패턴에 패턴 점수를 산정하는 것을 목표로 한다. 먼저, 모든 학습 대상 시나리오를 동사 군집의 대표 동사로 치환한다. 그리고 전체 시나리오에서 발생 했던 모든 세 개 이상의 동사 조합을 후보 패턴으로 추출한다. 예를 들면 총 다섯 문장(1,2,3,4,5)으로 이루어진 시나리오의 경우 총 여섯 개(123, 1234, 12345, 234, 2345, 345)의 후보 패턴이 추출된다. 추출 된 후보 패턴 중 전 단계의 행위 그래프를 작성할 때 낮은 빈도 점수에 의해 제거 된 벡터를 포함한 후보 패턴은 제외된다. 그리고 포함관계에 있는 두 패턴 (예: 1234, 234) 중, 발생 횟수가 동일한 경우 길이가 짧은(234) 패턴은 후보에서 제외된다. 두 가지의 조건에 위배되지 않는 패턴은 최종 패턴으로 채택된다.

추출된 각 패턴은 패턴점수를 갖는다. 패턴점수는 해당 패턴에 반영된 벡터의 평균 빈도점수, 평균 상대적 상호 연결성, 그리고 패턴의 길이를 정규화 하여 산정한다. 평균 빈도점수와 평균 상대적 상호연결성은 이 전단계의 행위 그래프에서 산정한 값을 패턴에 구성된 벡터의 점수의 평균으로 구한다. 다시 말해, 네 개의 문장으로 이루어진 패턴의 경우 세 개의 벡터를 갖게 되는데, 이 세 벡터의 빈도점수와 상대적 상호연결성을 각각 평균을 계산한다. 그리고 패턴의 길이 점수는 전체 후보 패턴의 평균 값을 기준으로 한 시그모이드 함수(Sigmoid function)를 이용한다. 시그모이드 함수를 이용하면 길이가 길수록 최대 값 1에 수렴하는 높은 값을 얻게 된다. 예를 들어 모든 후보 패턴의 평균 동사 개수가 4개인 경우 후보 패턴중 동사의 수가 4개면 0.5이 되고, 이를 기준으로 4개 미만이면, 낮을수록 0에 수렴하는 점수를, 초과이면 1에 수렴하는 더 높은 점수를 받게 된다. 수식 (4)는 본 단계에서 적용된 시그모이드 함수를 나타낸다. 이 식에서 x는 패턴의 평균 노드 수를 의미한다. 그림 5는 모든 후보 패턴의 평균 동사 수가 4인 경우 패턴 길이 당 점수를 보여준다.

$$y = \frac{1}{1 + e^{-(x-4)}} \tag{4}$$

Ⅳ. 사례 연구

본 논문에서는 기업체에서 작성한 시나리오를 수집하고 제안된 기법을 통해 패턴을 적용하는 방법을 보여준다. 그리고 추출된 패턴을 이용하여 시나리오 상에서 누락된 행위를 찾아주는 과정을 기술함으로써 패턴의 활용 사례를 통해 본 연구의 타당성을 검증한다.

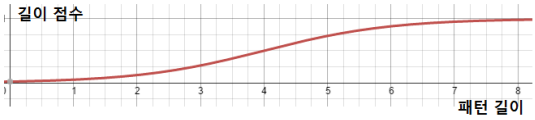

그림 5. 후보패턴의 평균 동사수가 4개인 경우 패턴 길이 가중치 그래프
Fig. 5. Length weight graph when the average of verb count is 4

1. 패턴 추출 사례 연구

본 사례연구를 위해 여덟 개의 프로젝트에서 83개의 시나리오를 수집하였다. 표 1은 수집된 프로젝트의 내역을 보여준다.

표 1. 사례 연구를 위한 프로젝트
Table 1. Project list for case study.

프로젝트	시나리오 수	문장 수
ATM	4	38
온라인 쇼핑몰	4	32
급여 시스템	13	132
카페테리아 주문 시스템	26	176
대학 정보 시스템	18	109
교통 관리 시스템	6	46
증권 주문 시스템	7	52
주문 진행 시스템	5	30
합계	83	615

표 2. 사례연구로부터 추출된 동사 군집의 사례
Table 2. Examples of generated verb cluster.

주어	동사 군집
user	pay, submit
user	choose, select
user	request, send, set
system	display,open
system	alert, notify, report
system	provide, recommend

먼저 여덟 개의 프로젝트에서의 시나리오에서 사용된 모든 문장에서 본동사를 추출한다. 그리고 두 번째 단계

인 '본동사 군집화' 과정을 통해 군집을 추출한다. '낮은 빈도의 동사'와 '높은 빈도의 동사'의 구분은 전체 발생 비율이 1%(615개의 문장 기준 6회) 미만인 경우는 '낮은 빈도의 동사'로 정한다. 본 사례연구에서는 총 32개의 군집이 추출되었으며, 표 2는 32개의 군집 중 일부를 보여준다. 예를 들면 첫 번째 줄의 "user pay", "user submit", "user send"는 하나의 군집으로 묶이며, 이 동사가 사용된 문장은 비슷한 행위를 한다고 간주한다.

다음은 세 번째 단계인 '행위 그래프 식별' 단계를 거친다. 그림 6는 본 사례연구에 의해 추출한 그래프의 일부분 이다. 즉 "user choose" 행위 이후에는 "system present", "system display", "system request"가 가능한데, 이들 중 "system display"가 가중치가 가장 높다. 이는 "user choose" 이후에는 "system display" 가 가장 자주 사용되고, 두 노드간의 응집력도 다른 노드에 비하여 높다는 것을 의미한다. 3장에서 언급한 바와 같이, 행위 그래프의 가중치는 빈도 점수와 상대적 상호연결성을 정규화 하여 산정 한다. 본 사례연구에서는 빈도 점수와 상대적 상호연결성을 3:7의 비율로 정규화 하였다.

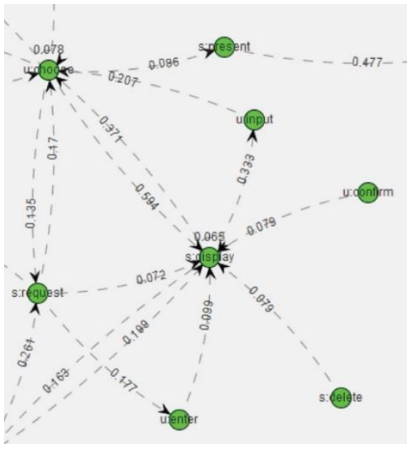

그림 6. 행위 그래프의 일부
Fig. 6. Part of scenario action flow graph

마지막 단계는 최종 시나리오 패턴을 추출하는 단계이다. 전 장에서 소개된 방법에 따라 행위 그래프를 기반으로 한 입력 시나리오에서 세 개 이상의 행위로 구성된 가능한 모든 행위 조합을 추출하고 후보패턴을 추출 한다. 본 사례연구에서는 총 54개의 패턴을 추출하였고, 그림 7은 그 일부를 보여준다. 이 그림은 총 아홉 개의 패턴을 보여주는데, 예를 들면 가장 왼쪽 줄의 "user choose - system provide - system log"로 구성된 패턴은 0.168의 패턴 점수를 획득하였다. 패턴점수는 앞에서 언급한 바와 같이 세 가지의 수를 정규화 하여 표현하는데 본 사례 연구에서는 빈도 점수, 상대적 상호연결성, 길이 점수를 각각 4:4:2의 비율로 정규화 하였다.

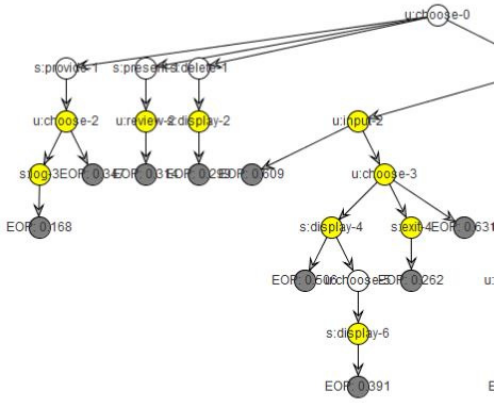

그림 7. 추출된 패턴의 일부
Fig. 7. Part of extracted patterns

1. 고객이 웹 쇼핑몰에 접속한다. (user access)
2. 고객이 주문할 상품을 고른다. (user choose)
3. 고객이 주문을 요청한다. (user request)
4. 고객이 결제 방식을 선택하여 결제를 요청한다. (user request)
5. 시스템은 결제를 진행한다. (system process)
6. 시스템은 결제결과를 보여준다. (system display)

그림 8. 컴퓨터공학과 학부생이 작성한 시나리오
Fig. 8. Software scenario written by undergraduate school student of computer engineering department

2. 패턴 적용 사례 연구

본 절에서는 앞 사례연구에서 추출된 시나리오 패턴을 이용하여 실제 시나리오에 적용하고, 어떻게 누락된 행위를 찾아내는지 보여주고자 한다. 그림8의 시나리오는 컴퓨터공학과 학부 2학년이 작성한 쇼핑몰 웹 애플리케이션에 '물품 구매'에 해당하는 요구사항 시나리오 이다.

이 시나리오는 한글 시나리오이기 때문에 앞에서 언급한 바와 같이 별도의 영어 번역이 필요하다. 문장 옆의 괄호는 해당 문장의 주어와 동사를 추출한 것이다. 이 시

나리오는 패턴 점수를 고려한 별도의 패턴 매칭 절차를 거치면, 시나리오에 적용된 패턴을 찾을 수 있다. 그림 9는 그림 8의 시나리오에 적용된 패턴을 보여준다. 첫 번째와 두 번째 문장은 “user access – system provide – user choose” 패턴으로 짝지어지고, 세 번째와 네 번째 문장은 “user access – system display – user access" 패턴이 으로 짝지어진다. "user request"는 동사 군집화 단계를 거치며 “user access"와 통합되었다. 적용된 두 패턴 모두에서 누락 행위가 발생하였는데, 1번 문장 이후에 "system provide"가, 3번 문장 이후에 “system display”가 생략 되었다. 본 사례는 추출된 패턴이 시나리오를 검증하는데 유용함을 보여준다.

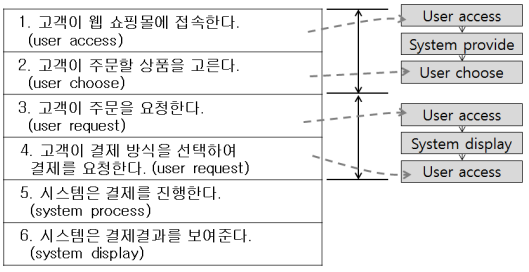

그림 9. 쇼핑몰 주문 결제 시나리오의 패턴 적용
Fig. 9. Applied patterns of 'Order item' scenario

V. 결 론

본 논문은 여러 소프트웨어 시나리오로부터 시나리오 패턴을 추출하는 완전 자동화된 방법을 제안하였다. 그리고 본 논문에서 제안하는 기법은 도메인에 관계없이 학습 후 패턴추출이 가능하다. 만약 특정 도메인에 관한 시나리오 패턴을 추출하고자 할 때는 해당 도메인의 시나리오를 입력하여 추출이 가능하다. 논문에서 제안하는 기법의 가시성을 위하여 8개의 실제 프로젝트에서 83개의 시나리오를 확보하여 사례연구를 실시하였고, 그 결과 총 54개의 패턴을 추출할 수 있었다. 추출된 패턴을 통해 경험이 적은 컴퓨터공학과 학부생이 작성한 시나리오의 누락된 행위를 찾아주는 사례를 보여주었다.

본 연구의 향후 연구로는 본 논문에서 소개한 기법에 의해 추출된 패턴을 실제 시나리오의 검증에 적용하는 연구를 진행한다. 사례연구에서 개략적으로 보여준 바와 같이 초심자가 입력한 시나리오와 추출된 패턴을 비교 분석하여 누락된 행위가 없는지 확인하고, 적합한 행위를 추천하는 완전 자동화된 기법 제안을 목표로 한다. 이 연구를 통해 본 논문에서 소개된 기법의 완성도를 한 번 더 검증할 예정이다.

References

[1] K. E. Wiegers, "Software Requirements", Microsoft press, 2013.

[2] R. L Glass, "The standish report: does it really describe a software crisis?", Communications of the ACM, vol. 49, no. 8, pp.15-16, 2006.

[3] S. Withall, "Software requirement patterns", Pearson Education, 2007.

[4] A. Malfouz, L. Barroca, R. Laney and B. Nuseibeh, "Patterns for Service-oriented Information Exchange Requirements", In Proceedings of the 2006 conference on Pattern Language of Programs, 2006.

[5] R. Biddle, J. Noble and E. Tempero, "Patterns for Essential Use Cases", Proceedings of KoalaPLoP, 2007.

[6] S. Ketabchi, N. K. Sani and K. Liu, "A Norm-based Approach towards Requirement Patterns", In Computer Software and Applications Conference(COMPSAC), 2011.

[7] C. Fellbaum, "WordNet", Wiley Online Library", 1998.

[8] S. Robertson, "Requirements Patterns vis Event/ Use Cases", Proceedings Pattern Languages of Programming, 1996.

[9] P. Resnik, "Using Information Content to Evaluate Semantic Similarity in a Taxonomy", arXiv preprint cmp-lg/9511007, 1995.

[10] T. Pang-Ning, M. Steinbach and V. Kumar, "Introduction to Data Mining", In Library of Congress, 2006.

[11] K. Ko, J. Lee, H. Moon and S. Lee, "Development of Data Fusion Human Identification System Based on Finger-Vein Pattern-Matching Method

and photoplethysmography Identification", The International Journal of Internet, Broadcasting and Communication, vol.7, no.2, pp.149-154, 2015.

[12] I. Alexander, "Scenario-driven search finds more exceptions", In Proceedings 11th IEEE International Workshop on Database and Expert System Application, 2000.

[13] S. Withall, "Software Requirement Pattners", Pearson Education, 2007.

[14] X. French, "Software Requirement Patterns". In Proceedings of the 2013 IEEE International Conference on Software Engineering, 2013.

[15] Y. Kim, D. Ko, S. Park and J. Kim. "Requirements Alternative Flow Detection Method Using Use Case Patterns at Use Case Scenario", Journal of KISS : Software and Applications, vol. 40, no. 9, pp. 491-501, 2013.

[16] X.Frach, C.Quer, S. Renault, C. Guerlain and C. Pal;omares, "Constructing and Using Software Requirement Patterns", In Managing Requirement Knowledge, pp. 95-116), 2013.

[17] Y. S. Im, E. Y. Kang, "MPEG-2 Video Watermarking in Quantized DCT Domain," The Journal of The Institute of Internet, Broadcasting and Communication(JIIBC), Vol. 11, No. 1, pp. 81-86, 2011.

저자 소개

고 덕 윤(준회원)

- 2011년 ~ 현재 : 서강대학교 컴퓨터 공학과 박사과정
- 2010년 : 서강대학교 정보통신대학원 석사
- 2006년 ~ 2011년 : 새한정보시스템 시스템 개발
- E-Mail : maniara.k@gmail.com

박 수 용(정회원)

- 1998년 ~ 현재 : 서강대학교 컴퓨터 공학과 교수
- 2012년 ~ 2014년 : 정보통신산업진흥원 원장
- 1995년 : 조지메이슨 대학교 정보기술학 박사
- 1988년 : 플로리다 주립대학교 컴퓨터학 석사
- E-Mail : sypark@sogang.ac.kr

김 순 태(정회원)

- 2014년 ~ 현재 : 전북대학교 소프트웨어공학과 조교수
- 2007년 : 서강대 학교 컴퓨터공학과 석사 및 박사
- 2003년 : 중앙대학교 컴퓨터공 학과 학사
- 2002년 11월 ~ 2004년 8월 : 소프트웨어 크래프트 컨설팅 선임컨설턴트
- E-Mail : stkim@jbnu.ac.kr

유 희 경(정회원)

- 1992년 ~ 현재 : 강원대학교 컴퓨터 공학과 교수
- 1995년 : 동국대학교 이학박사
- 1985년 : 동국대학교 이학석사
- E-Mail : hkyoo@kangwon.ac.kr

황 만 수(정회원)

- 1993년 ~ 현재 : 신한대학교 IT융합 공학부 교수
- 1987년 ~ 1993년 : LG소프트웨어 연구원
- 2001년 : 숭실대학교 공학박사
- 1986년 : 중앙대학교 이학석사
- E-Mail : mshwang@shinhan.ac.kr

※ 이 논문은 2016년도 정부(미래창조과학부)의 재원으로 한국연구재단-차세대 정보, 컴퓨팅기술개발사업의 지원을 받아 수행된 연구임(No. NRF-2014M3C4A7030503).