제26권 제4호(통권 100호) 2013년 12월 ISSN 1738-6322

소프트웨어공학소사이어티 논문지

Journal of Software Engineering Society

VOLUME 26, NUMBER 4, December 2013

소프트웨어 산업체 요구사항을 반영한 자동화된 프로젝트 계획 생성 지원 기법 및 도구 ……박지훈, 신동환, 홍광의, 서동원, 화지민, 배기곤, 서영석, 배두환 77

메소드의 매개변수 리스트의 간소화를 위한 리팩토링 방안 ……함동화, 이준하, 박수진, 박수용 93

사단법인 한국정보과학회
소프트웨어공학소사이어티

Journal of Software Engineering Society

VOLUME 26, NUMBER 4, December 2013

Automatic Project Planning Technique and Tool Based on Software Industry Requirements	Jihun Park Donghwan Shin Gwangui Hong Dongwon Seo Jimin Hwa Gigon Bae Yeong-Seok Seo Doo-Hwan Bae	77
Removing Long Parameter List Using Semantic Matrix	Dong Hwa Ham Jun Ha Lee Soo Jin Park Soo Young Park	93

Korean Institute of Information Scientists and Engineers
Software Engineering Society

소프트웨어 산업체 요구사항을 반영한 자동화된 프로젝트 계획 생성 지원 기법 및 도구[†]

(Automatic Project Planning Technique and Tool Based on Software Industry Requirements)

박지훈[‡] (Jihun Park) 신동환[§] (Donghwan Shin) 홍광의[§] (Gwangui Hong) 서동원[§] (Dongwon Seo)
화지민[‡] (Jimin Hwa) 배기곤[‡] (Gigon Bae) 서영석[‡] (Yeong-Seok Seo) 배두환[¶] (Doo-Hwan Bae)

요 약 소프트웨어 프로젝트 계획 생성 과정은 (1)프로젝트를 수행하기 위한 작업 구조(WBS)를 작성하고, (2)각 작업에 필요한 공수를 예측한 뒤, (3)작업에 인력을 할당하여, (4)전체 일정을 예측하는 과정으로 이루어진다. 프로젝트의 규모가 커질수록 가능한 작업 구조, 공수, 인력 할당의 조합의 수가 급격히 많아지며 이에 따라 프로젝트 계획 생성 과정의 복잡도가 매우 높아지게 된다. 따라서 이를 지원하기 위한 프로젝트 계획 생성 지원 기법이 필요하다.

본 연구에서는 실무 전문가 그룹과의 논의를 통해 소프트웨어 프로젝트 계획 생성 지원 기법에서 고려해야 할 여러 실무 요구사항들을 도출했다. 도출된 실무 요구사항을 고려하여 개발된 프로젝트 계획 생성 지원 도구 APP(Automatic Project Planner)는 개발 조직의 과거 지식 데이터를 활용한 공수 예측을 지원하며, 실무 이슈가 고려된 자동 인력 할당을 제공한다. 본 도구를 통해 합리적이고 현실적인 프로젝트 계획의 기반을 마련할 수 있다.

키워드 소프트웨어 프로젝트 계획, 소프트웨어 프로젝트 관리, 인력할당, 공수예측

Abstract To plan a software project, the manager (1)make a work breakdown structure (WBS), (2) estimate efforts for each task, (3) assign employee to each task, and (4) estimate overall schedule. When software project becomes complicated, the possible combination of WBS, effort, and employee assignments dramatically becomes larger. Software planning tool can help software project managers to deal with this complexity.

In this research, we discuss with a group of experts who work in software industry, to elicit practical requirements that should be considered in the software planning technique. Considering these requirements, we develop a software project planning tool APP (Automatic Project Planner) which provide effort estimation based on historical knowledge data and automatic human resource allocation. Our technique can be the basis of reasonable and practical software project planing.

Key words Software project planning, software project management, human resource allocation, effort estimation

[†] 본 연구는 방위사업청과 국방과학연구소의 지원으로 수행되었습니다.

[†] 비 회 원 : KAIST 전산학과
jhpark@se.kaist.ac.kr (Corresponding author)
jmhwa@se.kaist.ac.kr
ggbae@se.kaist.ac.kr
ysseo@se.kaist.ac.kr

[§] 비 회 원 : KAIST 전산학과
donghwan@se.kaist.ac.kr
gwangui.hong@se.kaist.ac.kr
dwseo@se.kaist.ac.kr

[¶] 종신회원 KAIST 전산학과
bae@se.kaist.ac.kr

1. 서 론

소프트웨어 프로젝트를 계획하는 단계에서 프로젝트 관리자는 (1)프로젝트를 수행하기 위한 작업 구조(Work Breakdown Structure, WBS)를 작성하고, (2)과거 지식과 데이터 혹은 경험, 직관 등을 이용해 각 작업에 필요한 공수를 예측한 뒤, (3)각 작업의 해당 공수를 수행할 수 있는 인력을 배치하여 (4)전체 개발 일정을 산출한다. 프로젝트 계획 수립 절차는 프로젝트의 규모가 커질수록 고려할 수 있는 작업 구조, 공수, 인력 할당 등의 조합의 수가 급격히 늘어나기 때문에 그 복잡성이 기하급수적으로 증가한다. 따라서 대규모 소프트웨어의 프로젝트 계획을 수립할 때 방대한 양의 지식 정보를 충분히 활용하여 최적의 계획을 수립하기 위해서는 프로젝트 계획 생성 지원 도구의 도움이 필요하다.

기존 연구를 통해 프로젝트의 각 작업에 필요한 공수를 예측하기 위해 COCOMO[1,2]와 같은 모델이 제시되기도 하였다. 또한 프로젝트 인력 할당 및 일정 산출을 위한 PERT(the Project Evaluation and Review Technique), CPM(the Critical Path Method)[3], RCPSP(the Resource-Constrained Project Scheduling Problem)[4] 등의 방법이 제시되었다.

하지만 기존 기법들은 실무에 적용하기 힘들 정도로 복잡한 입력을 사용자로부터 요구하거나, 주어진 조건 하에서 이론상으로 가장 짧은 시간에 프로젝트를 끝낼 수 있는 프로젝트 계획을 만드는데 집중하였을 뿐, 실제 프로젝트 진행에서 생길 수 있는 실무 이슈들을 고려하지 못하였다. 예를 들어 프로젝트 계획 생성 도구의 입력으로 개발 조직에서 관리하고 있지 않은 정보를 요구한다면 해당 정보를 만들기 위해 추가적인 노력이 필요하며, 입력이 복잡할수록 도구의 실제 활용도가 떨어질 것이다. 또한 소프트웨어 프로젝트의 계획 생성 지원을 위해서는 MS Project[11]와 같은 범용 프로젝트 계획 지원 도구에서 고려하지 못하는 소프트웨어 특화 요구사항들을 함께 고려해주어야 한다. 추가적으로, 인력 할당 및 일정 산출 시 업무 수행에 영향을 미칠 수 있는 실무 이슈들을 고려하지 않는다면, 실제 업무를 수행할 때의 효율이 이론적 수치보다 매우 낮아질 수 있다.

본 연구에서는 이러한 이슈들을 반영하기 위하여 실무 전문가 그룹과의 논의를 통해 소프트웨어 프로젝트 계획 생성 지원 도구가 반영해야 할 실무 요구사항들을 도출하였다. 저자들의 기존 연구[8]에서는 실무 이슈를 고려한 자동 인력할당 기법에 대해 연구하고, 그 효율성에 대해 검증하였다. 본 논문에서는 기존 연구의 인력할당 기법에 대한 추가 요구사항을 포함해 프로젝트 계획 생성 지원 기법을 위한 전반적인 실무 요구사항을 도출하였다. 이를 기반으로 WBS 입력, 지식 기반 공수 예측, 자동 인력 할당 및 일정 산출이 가능한 도구인 APP(Automatic Project Planner)를 개발했다. 또한 시뮬레이션 기능에 대해 요구사항을 도출하여 APP 도구에 반영하였고, 이를 통해 다양한 프로젝트 환경 변경에 따른 일정 변화를 검토해볼 수 있게 되었다.

APP 도구는 프로젝트 관리자에게 프로젝트 계획 생성을 위한 가이드라인을 제시해줄 수 있으며, 이는 합리적이고 현실적인 프로젝트 계획의 기반이 될 수 있다.

본 논문의 구성은 다음과 같다. 1장의 서론에 이어 2장에서는 실무 요구사항 도출에 대해 설명하고, 3장에서는 프로젝트 계획 생성 기법들에 대해 설명한다. 4장에서는 사용자 시나리오를 포함하여 APP 도구에 대해 설명하고, 5장에서는 관련 연구에 대해 설명한다. 6장에서는 결론 및 향후 과제에 대해 논의한다.

2. 실무 요구사항 도출

본 연구를 위해 프로젝트 계획 생성 지원 기법이 고려해야 하는 실무 요구사항들에 대해 실무 전문가 그룹과 논의하였다. 그 결과로 지식 기반 공수 예측과 자동 인력 할당 및 일정 산출에 대한 실무 요구사항들이 도출되었으며, 이 요구사항들에 대해 2장에서 설명한다.

2.1. 지식기반 공수 예측의 실무 요구사항

일반적인 기존의 프로젝트 공수 예측 기법들은 프로젝트 단위의 공수를 예측하는 기법이 대부분이다. 프로젝트 전체의 공수를 예측한다면, 각 하위 작업들의 공수를 예측하기 위해서 프로젝트 전체 공수를 각 작업에 할당하는 노력이 추가로 필요하다. 또한 기존에는 공수 예측 시 수치형 데이터의 사용이 많았는데, 예를 들어 COCOMO [1,2]는 신뢰성 (RELY), 제품 복잡도 (CPLX) 등의 복잡한 수치형 데이터를 입력으로 받고 있다. 이들을 개선하기 위해 도출된 실무 요구사항은 다음과 같다.

R1. 작업 단위 공수 예측 : 프로젝트 단위의 공수를 예측하여 각 작업에 분배하는 것보다 각 작업의 공수를 바로 예측하는 것이 프로젝트 관리자에게 더 유용한 정보를 제공할 수 있다. 실무에서 작업 특성에 따라 필요 공수가 유사하며, 이를 이용해 공수 예측 정확도를 높일 수 있다.

R2. 선택형 데이터 활용 : 실무환경에서 정보 축적 시 주로 관리 하는 정보는 입력이 복잡한 수치형 데이터보다 'DB 사용여부', '사용 도구' 와 같은 선택형 데이터가 많다.

2.2 자동 인력 할당의 실무 요구사항

기존 인력 할당 기법들은 프로젝트의 빠른 종료나 비용최소화만이 인력 할당의 목표가 되는 경우가 많았고, 이는 실무적 요인으로 인해 이론적 수치보다 업무 효율이 현저히 낮아질 가능성을 갖고 있다. 또한 프로젝트 계획 시 소프트웨어 특성을 고려해야 한다. 이들을 고려하여 도출된 실무 요구사항은 다음과 같다.

R3. 개발 기간 최소화 : 인력 할당의 가장 기본적인 목표로, 비용 최소화를 위해 개발 기간이 짧게 끝날 수 있는 인력 할당 필요하다.

R4. 개발자의 동시 수행 고려 : 개발자가 여러 업무를 동시에 수행하는 계획을 생성할 수 있어야 하지만 동시에 너무 많은 작업에 투입되면 업무의 효율성이 하락될 수 있다.

R5. 선후행 작업 연관성 고려 : 한 작업을 수행한 개발자가 되도록 후행 작업을 수행하도록 할당해야 문맥 전환 비용을 줄일 수 있다.

R6. 관리 체계 고려 : 높은 직급은 의사 결정 권한, 업무 진행 경험 등을 갖고 있으며, 낮은 직급이 일반적으로 개발 업무를 직접 수행하므로, 한 작업에 하나의 직급만 할당하는 것 보다 여러 직급을 같이 할당해야 한다.

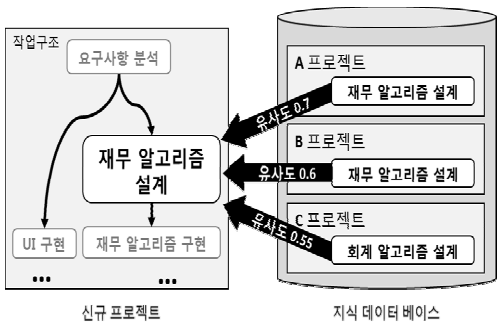
위의 요구사항들은 실무 환경을 고려하여 최적의 인력할당을 만들기 위한 실무 요구사항이다. 기존 인력할당은 주어진 WBS와 개발자 정보를 토대로 일회성 인력 할당을 만들어낸다. 할당 결과의 예측 종료 시점이 프로젝트 마감 기한보다 늦은 경우 관리자가 다른 개발 환경을 적용하여 인력 할당을 시뮬레이션 해볼 수 있는 기능에 대한 요구사항도 도출되었다. 추가적으로, 특정 개발자가 특정한 업무에 반드시 투입되어야 하는 상황에도 다른 작업들에 대해 인력할당을 진행할 수 있는 기능의 필요성 또한 도출되었다.

R7. 프로젝트 개발환경 변경 시뮬레이션 : 프로젝트에 개발자가 추가/변경/삭제되는 상황에 대하여 시뮬레이션이 필요하다. 일일 근무 시간, 주말/공휴일 근무 여부 및 근무 시간 등에 대한 변경 시뮬레이션도 필요하다.

R8. 작업 투입 개발자 고정 할당 : 개발자의 도메인 전문 지식이나 조직 내부 사정에 따라 특정한 업무에 개발자가 고정된 후, 다른 작업들에 대해 할당할 수 있는 기능이 필요하다.

3. 공수 예측 및 인력 할당 기법

본 논문에서 연구한 프로젝트 계획 생성 지원 기법은 (1)지식 기반 공수 예측 기법과 (2)자동 인력 할당 및 일정 산출 기법으로 구성된다. 작업 구조(WBS)를 입력으로 받아 개발 조직의 기존 지식을 기반으로 공수를 예측하며, 예측된 공수와 WBS를 이용하여 인력을 할당한 뒤 일정을 산출한다. 3장에서는 두 가지 기법을 중심으로 설명한다.



[그림 1] 지식 기반 공수 예측 기법 개요

3.1. 지식 기반 공수 예측 기법

지식 기반 공수 예측 기법은 기존 프로젝트 수행 결과인 지식 데이터베이스를 이용해 앞으로 수행할 프로젝트의 작업 공수를 예측하는 기법이다. 본 기법에서는 공수 예측에 대한 두 가지 실무 요구사항 (R1, R2)를 만족시키기 위해, (1) 작업 단위로 공수를 예측하며, (2) 수치형 데이터 외에 선택형 데이터를 포함한 다양한 데이터 타입을 이용해 공수를 예측할 수 있는 기법을 개발하였다. 본 기법은 J. Li의 기법[5]을 기반으로 실무 요구사항들을 반영하였다.

[그림 1]은 공수 예측 기법의 개요를 나타낸다. 지식 데이터베이스에서 같은 단계 (분석/설계/구현/테스트)의 가장 유사한 작업 n개를 선발한 뒤, 유사 작업들의 공수를 이용해 신규 작업의 공수를 예측한다. 유사도는 프로젝트 속성 정보와 작업 속성 정보를 이용하여 계산되며, 유사도가 높은 작업이 새로운 작업의 공수를 예측하는데 더 큰 영향을 미치게 된다.

3.1.1 지식 정보를 이용한 공수 도출

아래 수식 (1)은 지식 기반 공수 예측에 사용되는 공수 예측 공식이다.[5]

$$Effort(t,H)=\frac{\sum_{t_k^H\in H_{\text{topN}}}(Effort(t_k^H)\times Gsim(t,t_k^H))}{\sum_{t_k^H\in H_{\text{topN}}}Gsim(t,t_k^H)}\ldots.(1)$$

수식 (1)의 t는 공수를 예측하고자 하는 신규 프로젝트의 작업(task), H는 지식 데이터베이스(History)를 나타낸다. 수식 (1)의 좌변은 공수 예측이 지식 데이터베이스를 이용하며, 신규 프로젝트의 작업 단위로 예측이 된다는 것을 뜻한다.

우변의 공식 중 $Gsim(t,t_k^H)$의 경우 신규 작업 t와 유사작업인 t_k^H 사이의 유사도를 측정하는 공식이다. 작업 간 유사도는 프로젝트 속성 정보와 작업 속성 정보를 이용하여 아래 수식을 통해 계산된다.

$$Gsim(x,y)=\sum_{i=1}^{m}(w_i\times Lsim(x_i,y_i))\ldots(2)$$

수식 (2)의 x와 y는 유사도를 계산 할 두 작업을 나타낸다. m개의 속성 값에 대해, $Lsim(x_i,y_i)$ 함수를 이용해 각 속성 타입 별 유사도가 계산된다. w_i는 각 속성의 중요도(weight)를 나타내며, w_i값의 총 합은 1이다. w_i 값은 각 속성의 중요도에 따라 관리자가 변경할 수 있다.

수식 (1)의 $Effort(t_k^H)$의 경우 기존 작업 t_k^H이 수행되었을 때의 실제 공수를 뜻한다. $t_k^H\in H_{\text{topN}}$은 지식 데이터베이스에 있는 작업 t_k^H가 지식 데이터베이스에 있는 작업 중에 현재 작업 t와 유사도가 높은 순으로 정렬하였을 때, topN 순위 이내에 있다는 것을 뜻한다.

3.1.2 프로젝트/작업 속성

공수예측 시 프로젝트 간 유사도를 계산하기 위해 프로젝트/작업 속성이 사용된다. 유사도가 높은 프로젝트가 새로운 프로젝트의 공수를 예측하는데 더 큰 영향을 미치기 때문에 프로젝트/작업 속성 값은 공수예측에서 매우 중요한 정보이다. 이러한 프로젝트/작업 속성은 총 여섯 가지의 타입이 사용되며, 아래 [표 1]과 같이 분류된다. BIN, NOM, ORD, SET, FUZ, RNG 타입의 유사도 계산 방법은 J. Li의 기법[5]과 동일한 방법을 사용한다.

<표 1> 공수 예측 속성 타입

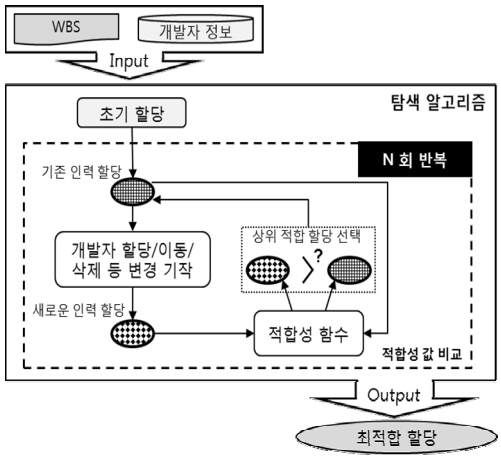
타입	설명	예시
BIN	이진 선택형	DB사용여부 [**예**/아니오]
NOM	선택형	주 언어 [**JAVA**/C++/C#]
ORD	숫자형	KLOC [250000]
SET	집합형	사용도구 [■A■B□C]
FUZ	Fuzzy set 형	언어비율 [C#=0.7,C=0.3]
RNG	범위 지정	참여인력 [10~15명]

3.2. 자동 인력 할당 및 일정 산출 기법

자동 인력 할당 및 일정 산출 기법은 공수가 입력 된 작업 구조(WBS)에 실무 이슈를 고려한 최적 인력 할당을 도출하여 일정을 산출하는 기법이다. 본 기법은 저자들의 기존 연구[8]를 기반으로 추가적인 요구사항을 반영하여 개발되었다.

[그림 2]는 인력 할당 및 일정 산출 기법의 개요를 나타낸다. 작업 구조와 개발자 정보를 입력으로 받아 임의의 초기할당을 생성한다. 탐색 알고리즘에서는 기존의 할당을 변경 기작을 이용하여 변경시키고, 새로운 할당이 기존 할당보다 좋은지 적합성 함수를 이용하여 평가한다. 이와 같은 과정을 N 회 반복한 뒤 가장 적합성 값이 높은 최적합할당을 찾는 것이 탐색 알고리즘의 목표이다.

2장에서 도출된 인력할당 기법의 실무 요구사항(R3, R4, R5, R6)을 만족하기 위해 이들 각각을 적합성 함수의 요소로 반영하였다. 시뮬레이션 요구사항(R7) 의 경우 기법 상에는 일정 산출 부분에서 반영이 되었고, 개발자 고정 할당 요구사항(R8)은 변경 기작에서 고려한다.



[그림 2] 인력 할당 기법 개요

3.2.1 인력 할당 기법 입력

인력 할당 및 일정 산출 기법은 공수가 입력 된 작업 구조(Work Breakdown Structure, WBS)와 개발자 정보를 입력으로 받는다.

작업 구조 (WBS)는 각 단위 작업들과 작업들 간의 선후행 관계로 이루어져 있다. 하나의 작업은 여러 개의 선행 작업을 가질 수 있으며, 모든 선행 작업이 종료된 후에 해당 작업이 수행될 수 있다. 작업의 공수는 공수 예측 기법의 결과를 토대로 프로젝트 관리자가 최종 수정 과정을 거쳐 확정한 값을 인력 할당 기법에 이용한다.

각 개발자는 할당된 작업을 수행하는 주체이다. 개발자 정보는 일반적으로 개발 조직에서 관리되며, 본 기법의 입력으로 필요한 정보는 개발자 ID, 직급, 업무 별 능력 수준이다. 개발

자의 직급은 기본적으로 사원, 대리, 과장으로 구분되는데, 이는 개발 조직 별로 설정이 가능하다. 직급이 높다는 것은 프로젝트 개발 업무에 대한 경험이 풍부하고 좀 더 큰 결정 권한을 갖는다는 것을 의미한다. 일반적인 조직에서 상위 직급일수록 해당되는 개발자의 수가 적으며, 따라서 각 작업에 의사 결정을 위한 상위 직급의 개발자와 실제 업무를 수행할 하위 직급의 개발자가 적절히 분포되어야 한다. 각 개발자의 능력은 실수로 표현되며, 능력의 Scale은 각 조직별로 설정 가능하다. APP 도구에서는 1.0을 기준으로 판단하며, 1.0이란 1 time unit (1시간) 동안 1MH (Man-hour)를 수행할 수 있다는 것을 뜻한다.

3.2.2 초기 할당 생성

탐색 알고리즘의 수행을 위해 초기 솔루션을 생성해야 한다. 일반적으로 탐색 알고리즘은 random-start(무작위 생성된 초기 솔루션에서 탐색을 시작하는 것) 방법을 많이 사용하지만, 적절한 초기 시작점을 잡아주는 것이 탐색 알고리즘의 효율을 높일 수 있다. 탐색 공간 (Search space) 상에서 최적 솔루션과 전혀 동떨어진 곳부터 탐색을 시작한다면, 탐색 알고리즘의 효율이 떨어질뿐더러, 최적해에 도달하지 못할 수도 있다. 본 기법에서는 개발자를 각 작업에 무작위로 할당하되, 비현실적인 초기 할당을 제외하기 위해 초기할당이 두 가지의 제약조건을 만족하도록 한다.

- 개발자는 최소 하나 이상의 작업에 투입이 되어야 한다.
- 작업에는 최소 한 명 이상의 개발자가 투입되어야 한다.

3.2.3 변경 기작

본 기법 에서는 아래와 같은 네 가지 변경 기작을 사용한다.

- 개발자 추가 할당: 임의의 작업에 개발자를 할당한다.
- 개발자 이동: 한 작업에 할당되어 있는 개발자를 다른 작업으로 이동시킨다.
- 개발자 삭제: 한 작업에 할당되어 있는 개발자를 삭제한다.
- 개발자 변경: 한 작업에 할당되어 있는 개발자를 다른 개발자로 변경시킨다.

위의 네 가지 변경 기작을 적용할 때, 개발자가 고정되어 있는 작업에 대해서는 변경 기작을 적용하지 않는다(R8). 변경 기작을 적용해 새로운 솔루션을 만들어 내고, 새로운 솔루션과 기존 솔루션 중 어떤 솔루션이 더 우수한지를 판단하기 위해 적합성 함수를 이용한다.

3.2.4 적합성 함수

$$FS = w_{CM} \times CM + w_{BM} \times BM + w_{VC} \times VC + w_{EA} \times EA$$

인력 할당 기법에 사용된 적합성 함수는 위와 같다. 실무에 영향을 미칠 수 있는 네 가지 요소를 식별하였으며, [표 2]는 각 적합성 함수 요소를 설명한다. 네 가지 적합성 함수 요소는 각각 인력 할당 기법의 네 가지 실무 요구사항 (R3, R4, R5, R6)을 반영한 것이다. 각 적합성 함수 요소의 값을 구해 중요도를 바탕으로 가중합을 구하는데, APP 도구에서는 각 적합성 함수 요소의 중요도가 동일하다고 가정하고, 균등한 적합도(w_k=0.25)를 사용했다. 적합도는 프로젝트 관리자의 판단에 따라 조정이 가능하다.

최적합 인력 할당이 도출되면, 이를 바탕으로 일정을 산출한다. 각 작업에 투입된 개발자가 Time unit(단위 시간)을 기준으로 능력에 맞게 업무를 수행하며, 여러 개의 작업을 동시에 수행할 경우 개발자의 능력이 분할된다. 한 작업이 끝나면 해당 작업의 후행 작업들이 차례대로 수행되며, 이를 통해 각 작업의 종료 날짜를 구하고, 일정을 산출할 수 있다.

3.2.5 일정 산출

<표 2> 적합성 함수 요소

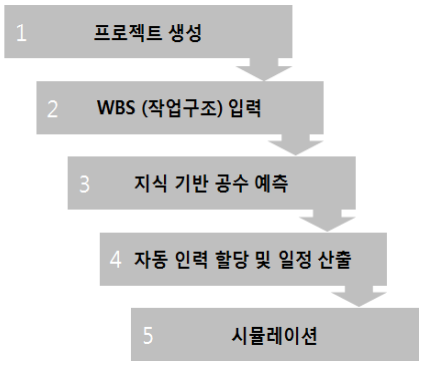
CM (Cost Minimization)	솔루션 할당대로 수행할 때의 예상 종료 시간이 짧은 수록 높아진다. 프로젝트 수행 시간을 최소화 하여 비용을 최소화한다. (R3)
BM (Burden of Multi-tasking)	개발자가 동시 수행 업무를 적게 수행할 수록 높아진다. 개발자가 동시에 너무 많은 작업을 진행하지 않도록 한다. (R4)
VC (Violation of Continuity)	연관성 있는 작업으로의 연속적인 할당이 많을수록 높아진다. 연관성 없는 작업으로의 할당이 최소화 되도록 한다. (R5)
EA (Evenness of Allocation)	각 작업에 직급이 고르게 분포되어 할당된 경우 높아진다. 하나의 작업에 특정 직급만 할당되지 않도록 한다. (R6)

적합성 함수 값이 가장 높은 최적합 할당을 찾으면, 해당 할당으로 프로젝트를 수행하는 일정을 산출한다. 일정은 time unit을 기반으로 계산되는데, APP 도구에서는 1시간을 단위로 생각한다. (time unit의 기준은 개발 조직 별로 다르게 설정할 수 있다.) 1 time unit이 지날 때마다 각 개발자는 본인의 능력에 따라 업무를 수행하게 된다. 한 개발자가 동시에 여러 개의 업무를 수행하는 경우에는 능력이 1/n으로 나누어져 수행하게 된다. 업무를 수행하며 선행 작업이 모두 끝나면 후행 작업이 시작하게 되고, 일일 근무시간을 고려하여 각 작업의 시작과 종료 날짜를 계산할 수 있다. 이 때, 개발 환경에 따른 시뮬레이션(R7)을 위해 일일 근무시간이 변경되거나, 휴일 근무가 추가되는 경우에는 이를 고려하여 날짜가 계산된다.

4. APP: 자동화된 프로젝트 계획 생성 지원 도구

4장에서는 본 연구를 통해 개발된 APP 도구를 프로젝트 관리자의 입장에서 사용 시나리오를 중심으로 APP 도구에 대해 자세히 설명한다.

[그림 3]은 일반적인 프로젝트 관리자가 APP 도구를 사용하여 프로젝트 수행 계획을 수립하는 단계를 나타낸다. 먼저 프로젝트 관리자는 신규 프로젝트를 생성하면서 해당 프로젝트에 관한 정보를 입력한다. 이어서 WBS 구조를 통해서 어떤 단위 작업들이 수행되어야 하는지 입력한다. 1단계와 2단계에서 입력된 정보를 바탕으로 3단계에서는 각 작업에 대한 공수를 자동으로 예측할 수 있으며, 4단계에서 각 작업에 대한 인력을 할당하고 소요 일정이 어떻게 되는지도 확인해 볼 수 있다. APP 도구는 또한 프로젝트 관리자가 원하는 방식으로 인력할당 및 일정 산출에 대한 시뮬레이션 하는 기능도 지원한다.



[그림 3] 프로젝트 계획 수립 시나리오

4.1 프로젝트 생성

새로운 프로젝트 계획을 수립하기 위해서 APP 도구에 신규 프로젝트를 생성하고 관련 정보를 저장해야 한다. 프로젝트 신규등록 버튼을 누르면 신규 프로젝트를 생성하기 위해서 필요한 정보를 입력할 수 있는 윈도우가 열린다. 해당 신규 프로젝트 정보 입력 윈도우에서 (1) 프로젝트 이름, (2) 프로젝트 매니저, (3) 프로젝트 속성 정보, (4) 프로젝트 참여 인력 정보 등을 입력하게 된다.

프로젝트 속성 정보는 지식 기반 공수 예측에 사용된다. APP 도구를 사용하는 조직의 특성에

맞추어 프로젝트 속성 정보를 사용할 수 있는데, 예를 들면 프로젝트 예산, 조직의 성숙도, 프로젝트 재사용 비율, 예상 KLOC, 프로젝트 난이도 등 다양한 측면에서 프로젝트의 속성을 나타내는 정보를 입력하게 된다. 많은 정보를 입력할수록 작업 간 유사도 계산에 이용할 수 있는 정보가 많아지므로, 공수 예측의 정확도가 높아진다. [그림 4]는 프로젝트 속성 입력 창을 보여준다. 작업의 기본 속성도 유사하게 프로젝트 생성 시에 입력할 수 있으며 개별 작업의 속성 설정은 이후에 WBS 입력에서 설명한다.

프로젝트 생성 시에 프로젝트에 참여할 개발자들을 설정한다. 각 개발자의 업무 수행 능력을 직급별로 다르게, 단계별로 다르게 설정할 수 있다. 예를 들면 프로젝트에 참여하는 두 명의 개발자가 특정 업무에 대해 능력이 다를 때, 업무 능력을 다르게 설정해주면 된다. 1.0 (1시간 동안 1MH의 업무 수행 능력)을 기준으로 직급에 따라서, 개발자 개인의 역량에 따라 다르게 설정한다. 개발자 별 능력의 차등 설정을 통해 보다 정확한 인력 할당 및 일정 산출이 가능하다.

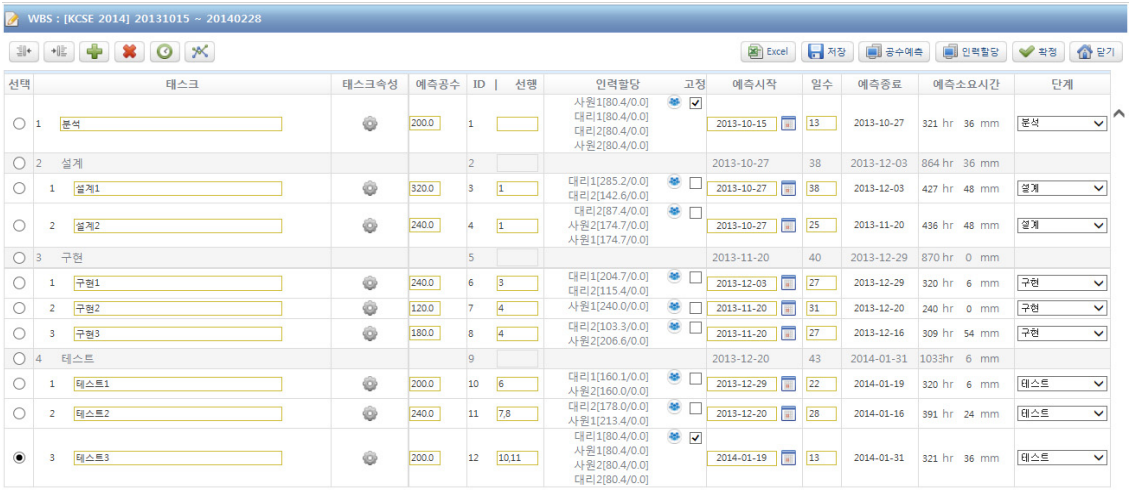
[그림 4] 프로젝트 속성 정보 입력 창

4.2 WBS 입력

신규 프로젝트에 대한 정보를 입력한 뒤, 다음으로 해당 프로젝트에서 실제로 수행하여야 할 작업 정보를 입력해야 한다. APP 도구에서는 소프트웨어 개발 프로젝트를 구성하는 단위 작업에 대한 작업 구조(WBS)를 입력하도록 설계되었다. WBS는 단순히 단위 작업을 나열한 것이 아니라 각 단위 작업들 사이의 작업 순서 관계까지도 직관적으로 나타낼 수 있는 장점이 있다.

[그림 5]는 WBS 관리 창에서 실제 인력 할당까지 모두 종료된 결과를 보여준다. WBS을 처음 입력하는 시점에서는 WBS 관리 윈도우가 빈 창으로 나타나며, 왼쪽 위의 + 버튼을 눌러서 원하는 단위 작업을 하나씩 입력할 수 있다. 각 단위 작업에 대해서 작업의 이름, 작업의 선행 작업, 작업 속성 정보 등을 입력할 수 있다. 이때 작업의 선행 작업은 WBS 상에서의 선후행 작업 관계를 나타낼 수 있도록 하는 정보이며, 선후행 관계가 없는 작업들은 병렬적으로 수행될 수 있다. 작업 속성 정보는 프로젝트 속성 정보와 마찬가지로 지식 기반 공수 예측을 위해서 사용되는 작업 단위의 정보이다. 작업 속성 정보 는 APP 도구를 사용하는 조직의 특성에 맞게 유동적으로 활용할 수 있는데, 예를 들어 예상 테스트 케이스 수, 예상 KLOC, 작업 복잡도, 작업 난이도 등의 정보를 단위 작업의 속성 정보로 활용할 수 있다.

만약 사용자가 입력해야 할 단위 작업의 개수가 너무 많거나 복잡한 경우 이것을 하나씩 APP 도구에 입력하는 것이 어려울 수 있는데, 이러한 문제를 해결하고 사용성을 높이기 위해서 본 APP 도구는 MS Excel 기반으로 작성된 파일을 이용하여 WBS의 입력 및 출력이 가능하도록 하였다. MS Excel을 이용하여 작성한 WBS 파일을 APP 도구에 불러오거나, 반대로 APP 도구에서 작성한 WBS 구조를 MS Excel 형태로 내보내어 다른 관리자 및 프로젝트 참여 인원들과 공유할 수 있다.



선택		태스크	태스크속성	예측공수	ID	선행	인력할당	고정	예측시작	일수	예측종료	예측소요시간	단계
○	1	분석		200.0	1		사원1[80.4/0.0] 대리1[80.4/0.0] 대리2[80.4/0.0] 사원2[80.4/0.0]	☑	2013-10-15	13	2013-10-27	321 hr 36 mm	분석
○	2	설계			2				2013-10-27	38	2013-12-03	864 hr 36 mm	
○	1	설계1		320.0	3	1	대리1[285.2/0.0] 대리2[142.6/0.0]	☐	2013-10-27	38	2013-12-03	427 hr 48 mm	설계
○	2	설계2		240.0	4	1	대리2[87.4/0.0] 사원2[174.7/0.0] 사원1[174.7/0.0]	☐	2013-10-27	25	2013-11-20	436 hr 48 mm	설계
○	3	구현			5				2013-11-20	40	2013-12-29	870 hr 0 mm	
○	1	구현1		240.0	6	3	대리1[204.7/0.0] 대리2[115.4/0.0]	☐	2013-12-03	27	2013-12-29	320 hr 6 mm	구현
○	2	구현2		120.0	7	4	사원1[240.0/0.0]	☐	2013-11-20	31	2013-12-20	240 hr 0 mm	구현
○	3	구현3		180.0	8	4	대리2[103.3/0.0] 사원2[206.6/0.0]	☐	2013-11-20	27	2013-12-16	309 hr 54 mm	구현
○	4	테스트			9				2013-12-20	43	2014-01-31	103hr 6 mm	
○	1	테스트1		200.0	10	6	대리1[160.1/0.0] 사원2[160.0/0.0]	☐	2013-12-29	22	2014-01-19	320 hr 6 mm	테스트
○	2	테스트2		240.0	11	7,8	대리2[178.0/0.0] 사원1[213.4/0.0]	☐	2013-12-20	28	2014-01-16	391 hr 24 mm	테스트
◉	3	테스트3		200.0	12	10,11	대리1[80.4/0.0] 사원1[80.4/0.0] 사원2[80.4/0.0] 대리2[80.4/0.0]	☑	2014-01-19	13	2014-01-31	321 hr 36 mm	테스트

[그림 5] WBS 입력 화면 및 인력 할당 결과

4.3 지식 기반 공수 예측

프로젝트 관리자가 WBS 정보를 모두 입력하였다면 APP 도구의 지식 기반 공수 예측 기능을 사용할 수 있다. WBS를 이루는 단위 작업에 대한 예측 공수를 산출하는 가장 기본적인 방법은 프로젝트 관리자의 경험을 이용한 방법이지만, 기존 프로젝트 수행 정보가 지식 데이터베이스에 남아있는 경우 이를 활용하여 더욱 정확하게 신규 프로젝트의 단위 작업에 대한 공수를 예측할 수 있다. 이를 위해서 신규 프로젝트의 프로젝트 속성 정보들과 단위 작업의 작업 속성 정보들이 사용된다.

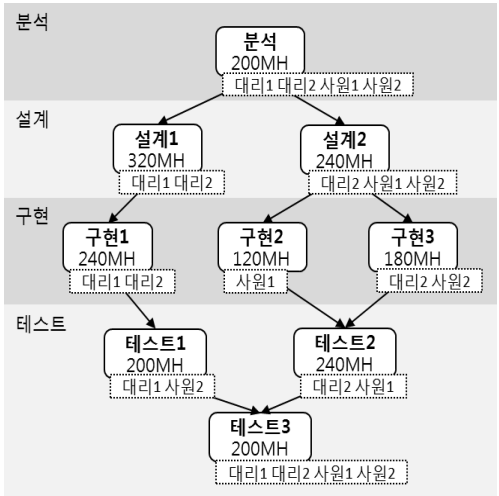
APP 도구의 중요한 특징은 이러한 지식 기반 공수 예측에 사용될 프로젝트 및 작업 속성 정보가 고정되어 있지 않으며, APP 도구를 사용하는 조직의 특성에 맞추어 변경 가능하다는 것이다. APP 도구에서는 특정한 속성 정보를 입력하도록 강제했던 기존의 방법[1, 2]과는 다르게 각 조직에서 이미 관리하고 있던 정보들을 활용할 수 있도록 속성 정보의 유연성을 제공하였다.

지식 기반 공수 예측은 [그림 5]에서 오른쪽 상단에 위치한 공수예측 버튼을 눌러서 수행할 수 있다. APP 도구는 자동으로 기존 프로젝트 수행 정보인 지식 데이터베이스에 접근하여 프로젝트 생성 시 입력했던 프로젝트 정보와 [그림 5]의 태스크 속성 버튼을 클릭해 입력할 수 있는 작업 속성 값 들을 활용하여 유사도 기반의 공수 예측을 수행한다. 지식 데이터베이스에서 기존 프로젝트 중 속성 정보 값들이 유사한 작업을 식별하여 유사 작업 들의 공수를 이용하여 현재 계획하고 있는 프로젝트 작업의 공수를 예측한다. 공수 예측을 수행하기 전에 예측 공수 항목은 공란으로 남아있고, 공수 예측 버튼을 클릭하면 [그림 5]의 분석 작업에 대한 공수가 200.0으로 입력된 것처럼 자동으로 공수가 예측된다. 이것은 분석 작업의 경우 200 MH의 공수가 필요할 것이라는 의미이다. 프로젝트 관리자는 각 작업에 대해 예측공수를 검토하여 최종적으로 각 작업의 공수를 확정하며, 이 공수를 바탕으로 인력 할당 및 일정 산출을 진행한다.

4.4 자동 인력 할당 및 일정 산출

지식 기반 공수 예측 단계를 거치면서 각 단위 작업에 대한 예측공수가 확정되면, 프로젝트 관리자는 각 단위 작업에 인력을 할당하여 그 결과로 종료되는 일정을 파악할 수 있다. [그림 5]에서 오른쪽 상단에 위치한 인력할당 버튼을 누르면 APP 도구는 주어진 WBS 정보와 예측공수 정보 및 참여 인력 정보를 이용하여 최적의 인력 할당 및 그에 대한 일정을 자동으로 산출한다.

[그림 5]의 인력할당 및 예측 시작, 예측 종료 항목들은 모두 프로젝트 관리자가 인력할당 버튼을 누른 뒤에 APP 도구에 의해서 자동으로 채워지는 항목들이다. 예를 들면 분석 작업의 경우 자동 인력 할당에 의해서 대리1, 대리2, 사원1, 사원2가 모두 투입된 것을 인력할당 항목에서 확인할 수 있으며, 이렇게 4명의 인력을 투입했을 때 2013년 10월 15일에 작업이 시작되고, 예측종료일은 2013년 10월 27일이라는 것을 알 수 있다.



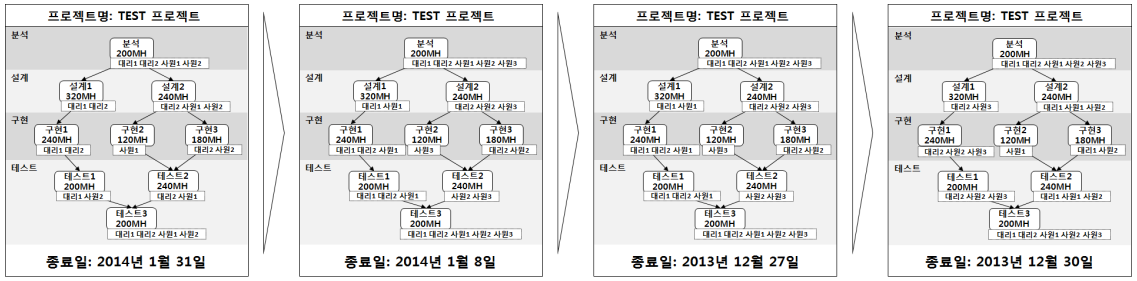
[그림 6] 자동 인력 할당 결과

[그림 6]은 [그림 5]에서 APP 도구가 자동 인력 할당의 결과로 도출한 내용들을 한 눈에 보기 쉽도록 WBS 구조 위에 도식화 한 것이다. 각 단계별 둥근 모서리 박스는 단위 작업을 의미하며, 작업 명 아래 숫자는 해당 작업의 예측공수를 의미한다. 각 작업별 하단의 점선 테두리 박스는 투입된 인력을 나타낸다. 예를 들면 분석 작업의 예측공수는 200이고, 자동 인력 할당 결과 대리1, 대리2, 사원1, 사원2이 모두 투입된 것을 알 수 있다.

[그림 6]을 통해서 자동 인력 할당 기법의 특징을 확인할 수 있다. 전체적인 할당 결과를 보면 대부분의 경우에 연관성 있는 작업으로 인력이 할당되는 것을 확인할 수 있다. 예를 들어 대리1의 경우 분석 → 설계1 → 구현1 → 테스트1 → 테스트3의 작업의 연속성이 지켜지는 경로를 따라서 지속적으로 투입되는 것을 볼 수 있다. 사원2의 경우 분석 → 설계2 → 구현3까지 지속적으로 할당되다가 테스트1 작업으로 넘어가기도 하는데, 이것은 작업의 연속성뿐만 아니라 수행 시간의 최소화를 함께 고려했기 때문이다. 만약 테스트2의 선행작업인 구현2와 구현3에 투입되었던 개발자가 모두 테스트2에 투입된다면, 각 개발자는 연관된 작업으로 투입되긴 하지만, 그만큼 테스트1에 할당될 개발자가 부족하거나 특정 개발자가 분할되어 동시에 두 개의 작업을 하느라 작업 효율성이 떨어지면서 전체 프로젝트 종료 일정이 늦어지는 문제가 발생할 수 있다. 이러한 상황을 종합적으로 고려하여 수동으로 최적의 인력 할당을 찾는 것은 매우 복잡한 일이지만, APP 도구를 이용하면 손쉽게 실무 이슈를 반영한 최적에 가까운 인력 할당 및 일정 산출을 할 수 있다.

4.5 시뮬레이션 기능

APP 도구의 개발 과정에서 실무 전문가 그룹과의 논의를 통해 시뮬레이션 기능에 대한 요구사항을 도출하였으며, 시뮬레이션 기능은 APP 도구의 가장 큰 특징으로 볼 수 있다.



[그림 7] 시뮬레이션 시나리오

프로젝트 관리자가 APP 도구를 이용하여 공수 예측 및 인력 할당을 진행하였을 때, 정해진 프로젝트의 마감 기한 내에 프로젝트를 종료할 수 있는 계획 도출이 불가능할 수도 있다. 예를 들어 특정 프로젝트가 반드시 12월 안에 개발이 완료되어야 하는데, 현재 개발자들이 정규 업무 시간 동안 하는 업무로는 최적 인력 할당을 이용해도 12월 안에 종료할 수 있는 일정 도출이 불가능할 수 있다. 이런 경우에 프로젝트 관리자는 개발자를 추가로 할당하거나, 기존의 개발자를 다른 개발자로 변경하는 등 프로젝트에 투입되는 개발자를 조정할 수 있다. 또한 개발자들이 정규 업무 시간 이외에 하루에 개발 업무에 투입되는 시간을 늘려보거나, 주말이나 공휴일에 추가적으로 업무를 수행하도록 업무 환경을 바꿀 수 있다. 이러한 개발 환경까지 프로젝트 계획 생성 도구를 이용하여 변경해볼 수 있어야 한다는 것이 실무 전문가들과 논의를 통해 도출 된 요구사항이다. (R7)

프로젝트의 업무 중에 특정 개발자가 반드시 투입되어야 하는 업무가 있을 수 있다. 이를테면 특정 분야의 도메인 지식을 갖고 있는 개발자가 투입되어 해당 개발자는 반드시 특정 컴포넌트의 설계 작업에 투입 되어야 하는 상황이 있을 수 있다. 또한 프로젝트의 진행 도중에 기 수행된 작업에 인력을 고정으로 할당하여 변경을 방지하고, 잔여 작업들에 대한 인력할당을 하는 경우에도 APP 도구를 사용할 수 있어야 한다는 실무적 요구사항이 도출되었다. (R8)

4.5장의 나머지 부분에서는 프로젝트에 투입된 개발자들과 기본 개발 환경 하에서 마감 기한 안에 프로젝트를 종료할 수 없는 가상 시나리오를 기반으로 개발자 추가 할당, 개발환경 변경, 개발자 고정 할당 세 가지 기능의 시나리오를 설명할 것이다. [그림 7]은 시뮬레이션 시나리오를 나타낸다.

4.5.1 시뮬레이션 기능 시나리오 환경

시뮬레이션 기능 시나리오를 설명하기 위한 가상 환경은 다음과 같다. 프로젝트 관리자는 프로젝트 'TEST'를 수행하기 위한 계획을 APP 도구를 이용하여 생성했다. 반드시 지켜야 하는 프로젝트의 마감 기한은 2013년 12월 31일이고, 현재 프로젝트에 투입된 개발자는 대리1, 대리2, 사원1, 사원2 네 명이다. 지식 기반 공수 예측을 바탕으로 각 작업의 공수는 확정되었으며, 도구의 최적 할당에 기반한 프로젝트 종료 예측 일은 2014년 1월 31일로, 프로젝트 마감 기한보다 한달이나 더 걸리는 것으로 예측되었다. 이 환경은 [그림 7]의 왼쪽 첫 번째 그림에 해당한다.

4.5.2 시뮬레이션 시나리오1: 개발자 추가 할당 및 변경

프로젝트 관리자는 프로젝트 종료 일정을 앞당기기 위해 개발자를 추가로 할당하는 방안을 우선적으로 고려했다. 개발자 추가 할당을 시뮬레이션 해보기 위해 추가로 투입 될 가능성이 있는 개발자들 중 한 명을 선택했다. 사원3의 추가 투입을 설정한 뒤에 프로젝트의 종료일이 2014년 1월 8일로 앞당겨진 것을 확인할 수 있었다.

프로젝트 관리자는 여러 명의 개발자를 투입하는 시나리오에 대해서도 시뮬레이션 해볼 수 있으며, 개발자를 추가하거나, 기존에 투입된 개발자를 다른 개발자로 변경하는 시나리오에 대해서도 시뮬레이션 해볼 수 있다. [그림 7]의 시나리오에서는 새로운 개발자를 추가하여 종료일을 앞당겼지만, 원하는 마감 기한 내에 프로젝트를 종료할 수 없는 상황이다.

4.5.3 시뮬레이션 시나리오2: 개발환경 변경

프로젝트 관리자가 개발자 추가 할당 이후에도 원하는 마감 기한 내에 프로젝트를 종료할 수 없을 때, 개발환경 변경을 고려해볼 수 있다. 개발환경에는 일일 근무 시간, 휴일/공휴일 근무 여부 및 근무 시간 등이 포함된다. 현재 도구의 기본 일일 근무 시간은 8시간으로 설정되어 있다. 이 기본 일일 근무 시간을 조정하여 특정한 기간 (집중 업무 기간)에 1일 10시간으로 근무를 하도록 설정을 하는 것이 가능하다. 또한 개발 조직 혹은 사내 이벤트로 인해 근무 시간을 축소/연장해야 하는 날 또한 고려하여 계획을 생성하는 것이 가능하다. 추가적으로 휴일 및 공휴일 근무 여부 및 근무 시간 설정이 가능하다. 기본적으로 휴일 및 공휴일은 근무 시간이 0으로 설정되어 있는데, 이 시간을 증가시킴으로써 휴일 및 공휴일 근무를 추가할 수 있다.

본 시나리오에서는 주말 근무 시간을 4시간씩 추가하여 시뮬레이션 해 본 결과, 종료일을 12월 27일로 프로젝트 마감 기한보다 앞당길 수 있었다.

4.5.4 시뮬레이션 시나리오3: 개발자 고정 할당

프로젝트 관리자가 특정 업무에 투입되는 개발자를 고정하기 위해 개발자 고정 할당 기능을 이용할 수 있다. 개발자 고정 할당 기능은 특정 개발자가 반드시 투입되어야 하는 임무가 있거나, 특정 상황에 따라 일부 작업에 개발자들을 강제로 할당해야 할 때 사용한다. 본 시나리오에서는 프로젝트 관리자의 판단에 따라 개발자 대리2와 사원3이 설계1 업무에 고정되어야 한다고 판단하여, 해당 개발자들을 설계1 업무에 고정한 뒤 재할당하였다. 이 결과로 전체적인 일정이 3일정도 연장되었지만, 관리자의 의도를 반영하여 인력 할당을 시뮬레이션 해 볼 수 있었다.

추가적으로 개발자 고정 할당 기능은 프로젝트의 수행 도중 향후 업무에 대한 계획을 수립할 때 도움을 줄 수 있다. 기 수행된 업무에 대해서 개발자를 강제할당 한 뒤 고정하면, 기 수행된 업무들의 연속성과 직급분포를 모두 고려하여 향후 업무들에 대해 최적의 할당을 산출할 수 있다.

4.5.5 시뮬레이션 기능 활용 방안

시뮬레이션 기능은 한 번의 최적 인력 할당으로 할당을 끝내는 것이 아니라 인력 할당에 필요한 다양한 변수를 조정하며 반복적으로 인력 할당을 해볼 수 있다는 점에서 프로젝트 관리자에게 큰 도움을 줄 수 있을 것이다. 주어진 조건 하에서 프로젝트가 계획되고 수행되는 경우도 있지만, 주어진 프로젝트 마감 기한 조건을 맞추기 위해 팀 구성원과 업무 환경을 사전에 계획하는 단계에서도 APP 도구가 이용될 수 있을 것이다.

5. 관련연구

본 연구는 소프트웨어 프로젝트 계획 생성의 각 단계인 작업구조 작성, 공수 예측, 인력할당, 일정 산출 기법을 모두 포함한다. 각 단계 중 공수예측, 인력할당에 해당하는 기존 연구들의 기법들을 본 연구의 기법들과 비교할 수 있으며 프로젝트 계획 생성을 지원하는 기존의 다른 도구들과의 비교도 가능하다.

공수 예측 기법의 기존 연구 중 대표적인 연구로, 제품 요소, 플랫폼 요소, 인적 요소, 프로젝트 요소로 예측된 소프트웨어 규모를 이용하여 공수를 예측하는 기법인 COCOMO 모델[1,2]이 있다. COCOMO 모델은 모델에서 정의된 요소들만 공수 산정에 이용하기 때문에 과거 지식을 활용하기 힘들고, 개발 조직에 해당 입력 정보가 없다면, COCOMO 모델을 활용하기 위해 새롭게 입력 정보를 만들어내야 한다는 단점이 있다. 지식 데이터 베이스에서 기 수행되었던 프로젝트 혹은 작업과의 유사도를 기반으로 공수를 예측하는 유사도 기반 방법도 제시가 되었다[5, 13, 14]. 본 연구에서는 타입별 유사도 계산 및 유사도 계산 공식을 제안한 J. Li et al.[5]의 기법을 이용하여 프로젝트와 작업 속성 정보를 기반으로 공수를 예측한다. 본 연구에서 제안한 기법은 각 작업 속성을 이용해 작업 레벨에서 공수를 예측하기 때문에, 기존에 프로젝트 단계의 유사도를 이용하여 공수를 예측하는 연구[13, 14]보다 단위 작업에 대해 좀 더 정확도가 높은 예측이 가능하며, 프로젝트의 공수를 각 작업에 분할하는 노력 없이 작업의 공수를 바로 예측할 수 있다는 장점이 있다.

C. K. Chang et al.의 연구[7]와 W. N. Chen et al.의 연구[9]는 프로젝트 개발에 투입되는 총 급여(비용)를 계산하는 방법을 정의한 뒤, 총 급여가 가장 적게 드는 인력 할당을 찾아내는 연구를 수행했다. [7]의 연구는 시간 별로 무작위 할당을 하며, [9]의 연구는 한 작업이 종료되거나 개발자가 추가/삭제되는 시점에 새로운 인력 할당을 하게 되는데, 이 때문에 한 개발자가 시간에 따라 계속해서 다른 작업에 할당되는 경우가 발생한다. 이러한 계획은 실제 수행 시 문맥전환 비용을 발생시키게 되며, 이로 야기된 비효율 때문에 이론상 값보다 훨씬 더 큰 비용이 소요될 수 있다.

기존의 대표적인 프로젝트 계획 및 관리 지원 도구는 Microsoft Project[11], OmniPlan[12] 등이 있다. 이러한 도구들은 WBS 설계, 자원 계획을 포함한 계획 기능과 Baseline관리, 진행율 관리 등의 관리 기능을 지원한다. 이들은 CPM[3], PERT 기법 등을 이용하여 일정 산출 등을 지원하지만, 생성된 계획에 대한 일정 산출만을 지원할 뿐, 계획 생성 과정에서 기존 지식을 이용한 공수 예측이나 자동 인력 할당 등의 기능을 지원하지 않는다.

6. 결론 및 향후 과제

본 연구에서는 실무 전문가 그룹과의 논의를 통해 실무 요구사항을 도출하여 지식 기반 공수 예측과 자동 인력 할당 및 일정산출을 포함한 프로젝트 계획 생성 지원 기법에 대하여 연구하고, 이를 반영한 도구인 APP(Automatic Project Planner)를 개발하였다. 공수 예측 기법의 경우 프로젝트 속성과 작업 속성을 이용해 기존 지식 데이터베이스에서 유사 작업을 추출하여 신규 프로젝트 작업의 공수를 예측하는 기법이다. 인력 할당 기법은 탐색 알고리즘을 이용해 실무 이슈를 고려한 최적 할당을 제공한다. 실무 전문가 그룹과의 논의를 통해 도출된 실무 이슈가 네 가지 적합성 함수 요소로 반영되었으며, 인력 할당 시뮬레이션에 대한 실무 요구사항 또한 기법에 반영되었다.

프로젝트 계획 생성 지원 도구인 APP의 지원을 통해 프로젝트 관리자는 WBS를 입력하고, 기존 지식 정보를 기반으로 각 작업의 필요 공수 추정치를 제공받을 수 있다. 프로젝트 관리자가 공수 추정치를 참고하여 각 작업의 필요 공수를 확정하면, 공수가 입력된 WBS와 개발자 정보를 이용해 APP 도구로부터 실무 이슈가 고려된 최적 인력할당을 제공받을 수 있다. 추가적으로, 본 기법의 가장 큰 특징인 프로젝트 수행 환경에 따른 시뮬레이션 기능을 제공함으로써 프로젝트 관리자가 개발자 추가/삭제/변경 등의 개발자 투입 조건, 일일 근무 시간, 휴일 근무 여부 등의 개발 환경 조건에 따른 프로젝트 일정을 시뮬레이션 해 볼 수 있다. 또한 APP 도구는 프로젝트의 특정 작업에 개발자가 고정 할당된 상태에서 다른 작업들에 대해 자동 인력 할당을 제공받을 수 있는 개발자 고정 할당 기능을 제공한다.

본 연구의 프로젝트 계획 생성 지원 기법을 이용함에 따라 프로젝트 관리자는 프로젝트 계획 과정의 시간적, 금전적 비용을 절약할 수 있다. 또한, 프로젝트 관리자가 모두 고려할 수 없는 방대한 양의 지식 정보들을 효과적으로 활용할 수 있다. 따라서 경험에 의존한 공수예측/인력 할당 보다 더 정확한 결과를 본 기법을 이용해 제공받을 수 있다.

본 기법의 개선을 위해 공수 예측 시 지식 데이터베이스의 정보가 충분하지 않을 경우에 대비하여 Function point 등의 입력을 통해 공수를 예측하는 방안에 대해 연구할 것이다. 현재 기능을 이용하여도 Function point 등을 이용한 유사도 기반 공수 예측은 가능하지만, Function point 기반 공수 산정 등의 기존 연구의 결과를 이용해 공수 산정 수식들을 추가로 적용해볼 수 있을 것이다.

또한 실제 사례로의 적용을 통해 인력 할당 기법에서 활용할 수 있는 적합성 함수 요소를 추가할 예정이다. 현재는 네 가지 적합성 함수 요소를 이용하고 있지만, 인력할당에 실무 이슈를 반영할 수 있는 추가적인 요구사항이 도출된다면, 이를 인력 할당 적합성 함수 요소에 반영할 수 있을 것이다.

추가적으로, 1일 근무 시간 연장, 주말 추가 근무 등의 개발 환경 변경에 따른 추가적인 비용이나, 개발자의 피로도 증가로 인한 소프트웨어 품질 저하 등을 시뮬레이션에 반영하는 기법에 대해 연구할 것이다. 업무 강도를 높인다면 프로젝트의 수행 기간이 짧아질 수 있지만, 소프트웨어의 완성도가 낮아질 가능성이 있으며, 이러한 효과에 대해 연구하고 도구에 반영할 것이다.

본 기법의 유용성을 검증하기 위해 실제 데이터를 이용하여 공수 예측 및 인력 할당의 정확도를 평가하고, 소프트웨어 전문가들을 대상으로 하여 사용자 study를 진행할 예정이다.

참 고 문 헌

[1] B. W. Boehm, Software Engineering Economics, Prentice-Hall, 1981.

[2] B. W. Boehm, C. Abts, A. W. Brown, S. Chulani, B. K. Clark, E. Horowitz, R. Madachy, D. J. Reifer, B. Steece, Software Cost Estomation with COCOMO II, Prentice-Hall, 2000.

[3] A. Shtub, J. F. Bard, S. Globerson, Project Management: Processes, Methodologies, and Economics, second ed. Prentice Hall, 2005.

[4] P. Brucker, A. Drexl, R. Mohring, K. Neumann, E. Pesch, “Resource-Constrained Project Scheduling: Notation, Classification, Models and Methods”, European Journal of Operational Research, vol.112, no.1, pp.3-41, Jan. 1999.

[5] J. Li, G. Ruhe, A. Al-Emran, M. M. Richter, “A flexible method for software effort estimation by analogy”, Empirical Software Engineering, vol.12, no.1, pp.65-106, Feb. 2007.

[6] S. Russell, P. Norvig, Artificial Intelligence: A Modern Approach, 2nd Ed., Prentice Hall, 2002.

[7] C. K. Chang, H. Jiang, Y. Di, D. Zhu, Y. Ge, “Time-line based model for software project scheduling with genetic algori-thms“, Information and Software Technology, vol.50, no.11, pp.1142-1154, Oct. 2008.

[8] J. Hwa, J. Park, D. Shin, G. Hong, G. Bae, Y-S. Seo, D-H. Bae, “Automated Human Resource Allocation based on Practical Feedback from Software Industry“, Journal of KIISE : Software and Applications, vol.40, no.7, pp.369-380, Jul. 2013.

[9] W. N. Chen, J. Zhang, “Ant Colony Optimization for Software Scheduling and Staffing with an Event-Based Scheduler”, Software Engineering, IEEE Transactions on, vol.39, no.1, pp.1-17, Jan. 2013.

[10] D. Kang, J. Jung, D-H. Bae. “Constraint-based human resource allocation in software projects”, Software – Practice and Experience, vol.41, no.5, pp.551-577, Sep. 2011.

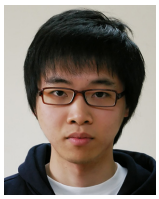
[11] Project, Microsoft, [Online]. Available: http://www.microsoft.com/project/en-us/preview/project-requirements.aspx(downloaded 2014, Jan. 22)

[12] OmniPlan, The Omni Group, [Online]. Available: http://downloads2.omnigroup.com/software/MacOSX/Manuals/OmniPlan-2-Manual.pdf (downloaded 2014, Jan. 22)

[13] M. Auer, A. Trendowicz, B. Graser, E. Haunschmid, S. Biffl, “Optimal Project Feature Weights in Analogy-Based Cost Estimation: Improvement and Limitations”, Software Engineering, IEEE Transactions on, vol.32, no.2, pp.83-92, Feb. 2006.

[14] Y. F. Li, M. Xie, T. N. Goh, "A study of project selection and feature weighting for analogy based software cost estimation", The Journal of Systems and Software, vol.82, no.2, pp.241–252, Feb. 2009.

저자 소개



박 지 훈

2010년 KAIST 전산학과 학사
2012년 KAIST 전산학과 석사
2012년~현재 KAIST 전산학과 박사과정
관심분야는 소프트웨어 버그 예측, 소프트웨어 저장소 마이닝

신 동 환

2006년~2010년 KAIST 전산학과 학사
2010년~2012년 KAIST 전산학과 석사
2012년~현재 KAIST 전산학과 박사과정
관심분야는 소프트웨어 모델 테스팅, 뮤테이션 분석

홍 광 의

2012년 아주대학교 정보컴퓨터공학부 학사
2012년~2014년 KAIST 전산학과 석사
2014년~현재 KAIST 전산학과 박사과정
관심분야는 소프트웨어 정형 검증

서 동 원

2012년 아주대학교 정보컴퓨터공학부 학사
2012년~현재 KAIST 전산학과 석사과정
관심분야는 소프트웨어 모델 테스팅, 소프트웨어 프로젝트 계획

화 지 민

2006년 KAIST 전산학과 학사
2006년~현재 KAIST 전산학과 석박사 통합과정
관심분야는 소프트웨어 재구조화, 아키텍쳐, 소프트웨어 리팩토링, 소프트웨어 이해, 소프트웨어 품질, 소프트웨어 프로젝트 계획

배 기 곤

2006년 KAIST 전산학과 학사
2006년~현재 KAIST 전산학과 석박사 통합과정
관심분야는 소프트웨어 테스팅, GUI 테스팅, 자동화 도구

서 영 석

2006년 숭실대학교 컴퓨터학부 졸업(학사)
2008년 KAIST 전산학과 졸업(석사)
2012년 KAIST 전산학과 졸업(박사)
2012년~2013년 KAIST 정보전자연구소 연수연구원
2014년~현재 한국산업기술시험원 선임연구원
관심분야는 소프트웨어 비용 산정, 소프트웨어 데이터 마이닝, 소프트웨어 프로세스 개선, 소프트웨어 아키텍처

배 두 환

1980년 서울대학교 조선공학 졸업(학사)
1987년 Univ. Of Wisconsin -Milwaukee 전산학과 졸업(석사)
1992년 Univ. Of Florida 전산학과 졸업(박사)
1995년~현재 KAIST 전산학과 교수
관심분야는 소프트웨어 프로세스, 객체지향 프로그래밍, 컴포넌트 기반 프로그래밍, 임베디드 소프트웨어 설계, 관점 지향 프로그래밍

메소드의 매개변수 리스트의 간소화를 위한 리팩토링 방안†

(Removing Long Parameter List Using Semantic Matrix)

함동화‡ (Dong Hwa Ham) 이준하‡ (Jun Ha Lee) 박수진§ (Soo Jin Park) 박수용¶ (Soo Young Park)

요 약 소프트웨어의 규모는 시간이 지남에 따라 복잡성과 유지보수 비용이 증가한다. 이로 인해 최근 유지보수의 중요성이 더욱 대두되고 있다. 소프트웨어가 진화 할수록 유지보수를 어렵게 하는 징후인 코드의 나쁜 냄새(Bad Smell)가 점점 심해지기 때문에 나쁜 냄새가 나는 코드를 제거하여 유지보수를 용이하게 개선해야 한다. 최근에는 이러한 나쁜 냄새를 위해 소프트웨어 리팩토링 기법에 대한 연구가 많이 연구되고 있다. 본 논문에서는 나쁜 냄새의 한 종류인 긴 매개변수 리스트(Long Parameter List)를 식별하고 해결하여 소프트웨어의 유지보수성을 향상시키는 방안을 제안한다. 제안되는 방안은 매개변수간의 의미적인 유사도를 측정하여 이를 군집화 하여 새로운 객체가 될 수 있는 매개변수들을 식별한다. 제안되는 방안은 경력 있는 객체지향 소프트웨어 개발자들이 군집화한 매개변수리스트와의 비교를 통해 평가되고, 그 결과가 통계적으로 검증된다.

키워드 리팩토링, 긴 매개변수 리스트, 유지보수, 품질

Abstract Complexity and maintenance cost of software increase as much as software has been evolved, therefore importance of software maintenance recently arise. There are many signs that are difficulties to maintain software, called bad smell, in a large-scale software. The bad smell should be removed to improve maintainability. Recently, many software refactoring methods have researched to terminate the bad smell. In this paper, we propose how to identify long parameter list, which causes bad smell, and how to solve the problem for increasing software maintainability. In our approach, we classify the parameters for creating new objects by measuring semantic similarity among them. This is evaluated by experienced software developers, and the result is statistically verified.

Key words Refactoring, Long Parameter List, Software Maintenance, Software Quality

† 이 논문은 2013년도 정부(미래창조과학부)의 재원으로 한국연구재단-차세대정보·컴퓨팅기술개발사업의 지원을 받아 수행된 연구임 (No. 2012M3C4A7033348).

‡ 비 회 원 : 서강대학교 컴퓨터공학과
donghwa@sogang.ac.kr
juna@sogang.ac.kr
§ 정 회 원 : 서강대학교 기술경영전문대학원 조교수
psjdream@sogang.ac.kr
¶ 종신회원 : 서강대학교 컴퓨터공학과 교수
sypark@sogang.ac.kr

1. 서 론

오늘날의 소프트웨어는 과거에 비해서 그 규모와 복잡성이 증가하고 있다[1]. 이에 따라 유지보수 비용 또한 증가하는데, 이때의 비용이 전체 비용의 67%를 차지한다[2, 3]. 따라서 소프트웨어의 유지보수성을 향상시켜 변경하기 용이한 코드로 개선시킬 필요가 있다. 소프트

웨어의 유지보수성을 향상시키기 위한 방법 중 하나로 리팩토링을 사용할 수 있다[4-7]. 리팩토링이란 기능은 그대로 유지한 채, 소스코드를 읽기 쉽고 수정하기 편하도록 내부를 수정하는 작업으로 정의될 수 있다[2]. M. Fowler는 소스코드에서 리팩토링의 대상이 될 수 있는 부분을 Bad Smell이라 정의했는데, 이는 소스코드에서 잠재적으로 문제를 발생시킬 수 있는 부분을 말한다[2].

코드의 나쁜 냄새 중 하나로 메소드에 선언된 매개변수가 과도하게 많을 때 이것을 긴 매개변수 리스트(Long Parameter List)라 정의하였다[2]. 매개변수 리스트가 길어질수록 코드의 가독성을 떨어뜨리기 때문에 코드를 이해하고 유지보수하기 어려워진다. 이러한 이유로 긴 매개변수 리스트를 식별하고, 이를 간소화하는 리팩토링이 필요하다.

M. Fowler는 긴 매개변수 리스트를 리팩토링하기 위해 매개변수를 객체로 전하는 방안을 소개하였다[2]. 이는 의미적으로 유사한 매개변수를 하나의 객체로 묶어서 관리하는 방법으로서, 메소드의 가독성을 증가시켜 유지보수를 용이하게 한다. 그러나 규모가 큰 소프트웨어의 경우 모든 메소드의 매개변수리스트를 조사하여 그들 간의 의미적인 유사성을 판단하고 객체로 묶는 작업은 시간과 노력이 많이 드는 작업일 뿐만 아니라, 사람에 따라 의미적인 유사성을 달리 판단할 수도 있다[2].

본 논문에서는 긴 매개변수 리스트문제를 해결하기 위해 매개변수 사이의 의미적 유사도를 측정하여 긴 매개변수 리스트를 식별하고, 군집화하는 방안을 제안한다. 제안하는 방법은 두 단계로 이루어진다. 첫 번째 단계에서는 대상으로 하는 전체 메소드의 매개변수의 식별자를 추출한 후 모든 매개변수 쌍에 대해 의미적 유사도를 측정한다. 두 번째 단계에서는 첫 번째 단계에서 측정한 의미적 유사도를 기반으로 긴 매개변수 리스트를 식별하고, 이에 대해 군집화 한다. 본 논문에서는 소프트웨어의 모듈화를 목적으로 하는 다양한 연구에서 그 효용성이 검증된 계층적 응집 군집화 알고리즘을 사용한다[8]. 계층적 응집 군집화 알고리즘은 상향식 군집화 알고리즘으로써 가장 유사한 군집간의 병합을 통해 군집화를 수행한다[9].

본 논문의 구성은 다음과 같다. 2장에서는 긴 매개변수리스트를 리팩토링하기 위해 M. Fowler가 소개한 방안에 대하여 설명하고, 이 방안들의 문제점을 기술한다. 3장에서는 긴 매개변수 리스트를 정의하고, 본 논문에서 문제를 해결하기 위해 제안한 접근방법을 기술한다. 4장에서는 본 논문에서 제안한 접근방법을 이용하여 세 개의 오픈소스 시스템을 대상으로 실험을 진행한 후 그 결과를 평가한다. 마지막으로 5장에서는 본 논문의 결론과 향후 연구가 기술된다.

2. 관련 연구

다른 나쁜 냄새에 비해 긴 매개변수 리스트는 거의 연구된 바가 없다. 따라서 본 장에서는 M. Fowler가 처음에 소개한 긴 매개변수 리스트를 해결하기 위한 리팩토링 기법을 설명하고, 이 기법들의 특징과 한계를 기술한다. M. Fowler가 소개한 3가지 리팩토링 기법은 다음과 같다[2].

1) 메소드로 전환(Replace Parameter Method)
2) 객체 전체를 전달(Preserve Whole Object)
3) 매개변수를 객체로 전환
   (Introduce Parameter Object)

첫 번째는, 어떤 메소드 B가 메소드 A의 수행결과를 매개변수로 사용할 때, 메소드 B의 구현부에서 메소드 A를 직접 호출할 수 있으면, 그 매개변수를 제거하고 메소드 B의 구현부에서 메소드 A를 직접 호출하도록 변경하는 리팩토링

기법이다. 두 번째는, 특정 객체로부터 가져온 데이터를 매개변수로 사용할 때, 기존의 매개변수를 그 객체로 대체하는 리팩토링 기법이다. 마지막은, 다수의 매개변수를 하나 혹은 여러 객체로 전환하는 리팩토링 기법이다. 첫 번째와 두 번째 리팩토링 기법은 특정 상황을 만족해야 하기 때문에 일반적으로 사용하기 어렵다. 반면에, '매개변수를 객체로 전환' 리팩토링 기법은 상황에 상관없이 사용할 수 있지만, 어떻게 다수의 매개변수를 객체로 전환할지는 여전히 개발자의 주관적인 판단에 의존한다[2]. 즉, 여러 개의 매개변수를 하나의 객체로 전환할지, 다수의 객체로 나누어 전환할지를 판단하는 기준은 개발자 마다 다를 수 있다. 따라서 본 논문에서는 긴 매개변수 리스트를 정형화 하여 정의하고, 이를 식별하고 간소화하기 위한 방안을 제시한다.

3. 접근 방법

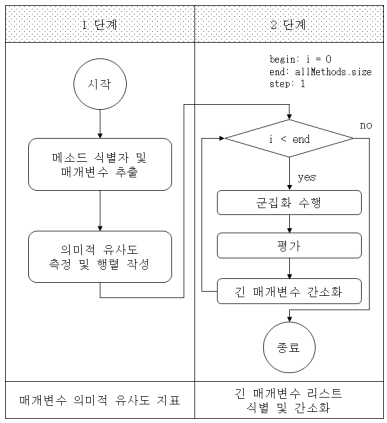
본 장에서는 긴 매개변수 리스트를 정의하고, 정의된 긴 매개변수리스트를 식별하고 이를 군집화 하여 매개변수의 수를 줄이기 위한 방안을 기술한다.

M. Fowler는 하나의 메소드에서 사용하는 매개변수가 과도하게 많을 때 이것을 긴 매개변수 리스트라 정의하였는데[2], 이는 주관적으로 해석될 수 있다. 본 논문에서는 긴 매개변수 리스트를 매개변수 리스트 중 한 쌍 이상의 매개변수가 의미적으로 관련되어 있어 하나의 클래스로 그룹화 되어 관리되어야 할 필요가 있는 매개변수 리스트라고 정의한다. 더 정형화한 정의를 위해 메소드 m의 매개변수의 집합을 $P_m = \{p_0, p_1, \ldots, p_i\}$라 하자. 그러면 의미적으로 유사한 매개변수 쌍의 집합 SSP_m은 아래와 같이 정의될 수 있다.

$$SSP_m = \{(p_i, p_j) | p_i \neq p_j \wedge SS(p_i, p_j) > \tau\}$$

의미적으로 유사한 매개변수의 집합이 $SSP_m \neq \Phi$인 조건을 만족할 때, 메소드 m의 매개변수집합 P_m은 긴 매개변수 리스트로 규정될 수 있다. 이 때 의미적인 유사도 $SS(p_i, p_j)$는 다음 장에서 자세히 기술 되며, 임계치 τ는 실험을 통해 경험적으로 결정된다.

[그림 1]는 긴 매개변수 리스트를 식별하기 위한 전체적인 프로세스를 보여준다.



[그림 1] 긴 매개변수 리스트 문제를 해결하기 위한 프로세스

본 논문에서 제안하는 방법은 크게 2단계로 나눈다. 1단계에서는 모든 메소드에 대해서 매개변수의 식별자를 추출하고 그들 간의 의미적인 유사도를 측정한다. 의미적인 유사도는 추출된 매개변수의 식별자가 같은 메소드 내에서 동시에 변수(local variable, parameter)로 선언될 조건부 확률로 정의된다. 매개변수 쌍의 식별자와 같은 식별자를 가지는 변수가 같은 메소드 내에 많이 선언될수록 그 매개변수 쌍의 의미적인 유사도는 높다. 2단계에서는 전 단계에서 측정된 의미적 유사도를 이용하여 군집화를 수행하고, 군집화의 결과로 생성된 군집의 수와 형태에 따라 긴 매개변수 리스트를 식별한다.

3.1 매개변수 의미적 유사도 지표

매개변수의 의미적 유사도를 측정하기 위해 매개변수의 식별자가 같은 메소드 내에서 얼마나 같이 등장하는지에 대한 정도를 측정하였다. 메소드는 한 가지 목적을 위해서 작성되기 때문에 [10], 메소드 내에 식별자 쌍이 동시에 많이 등장한 다는 것은 그 식별자 쌍으로 선언되어 있는 매개변수가 같은 목적을 가지고 사용된다는 것을 보여준다. 따라서 같은 메소드 내에서 많이 등장하는 식별자를 가지는 매개변수 쌍일수록 그 매개변수 쌍의 의미적 유사도는 높다고 할 수 있다. 아래의 그림 2는 본 논문에서 제안하는 방법을 설명하기 위해 임의로 작성한 예제 소스코드이다.

```
public void foo1(int a, int, b, int c) {
    a++;
    b = 10;
    c = 20;
}
public void foo2() {
    int b, c;
    b = 50;
    c = 100;
}
public void foo3() {
    int a = 10;
}
```

[그림 2] 식별자 추출을 위한 소스코드 예제

첫 번째 메소드인 foo1 메소드의 세 개의 매개변수의 식별자는 a, b, c이다. 이 매개변수의 식별자간의 의미적인 유사도를 추출하기 위해 전체 (foo1, foo2, and foo3) 메소드 내에 선언된 변수의 식별자를 아래와 같이 추출하였다.

<표 1> 추출한 메소드 식별자

메소드	식별자
foo_1	a, b, c
foo_2	b, c
foo_3	a

foo1 메소드의 식별자는 매개변수로 받아 사용한 a, b, c이고, foo2 메소드의 식별자는 지역변수로 선언하여 사용한 b, c이다. 마지막으로, foo3의 식별자는 지역변수로 선언하여 사용한 a이다.

이를 이용하여 두 매개변수 의미적 유사도는 아래와 같이 측정될 수 있다. 의미적인 유사도의 정형화된 정의를 위해서 매개변수 a의 식별자가 포함된 메소드의 집합을 M_a라고 하고 정의하면 두 매개변수 p_i, p_j의 의미적 유사도 $SS(p_i,p_j)$는 Bayes' Rule[11]을 이용하여 아래와 같이 정의될 수 있다.

$$SS(p_i,p_j)= Max\left[\frac{|M_{pi}\cap M_{pj}|}{|M_{pi}|},\frac{|M_{pi}\cap M_{pj}|}{|M_{pj}|}\right]$$

두 매개변수 사이의 의미적 유사도는 전체 메소드에서 동시에 두 매개변수의 식별자를 이용하여 선언된 변수가 전혀 없을 때 최소값을 가지며, 반대의 경우 최대값을 가진다.

위의 소스코드를 예로 들어, foo1 메소드의 매개변수 a와 b의 의미적 유사도는 다음과 같이 측정된다. 첫 번째로, 전체 메소드 중에서 a라는 식별자가 등장하는 메소드의 수를 측정한다. 위의 표에 따라 2개의 메소드(foo1, foo3)에서 a라는 식별자가 등장함을 알 수 있다. 마찬가지로, 전체 메소드 중에서 b라는 식별자가 등장하는 메소드의 수를 측정한다. 위의 표에 따라 2개의 메소드(foo1, foo2)에서 b라는 식별자가 등장함을 알 수 있다. 마지막으로, 전체 메소드 중에서 a와 b라는 식별자가 동시에 등장하는 메소드의 수를 측정한다. 위의 표에 따라 1개의 메소드(foo1)에서만 두 개의 식별자가 동시에 등장함을 알 수 있다. 이 때 매개변수의 의미적 유사도의 정의에 따라 a와 b의 의미적 유사도는 0.5(1/2)로 측정될 수 있다.

3.2 긴 매개변수 리스트 식별

본 논문에서는 긴 매개변수 리스트 문제를 해결하기 위해 매개변수 사이의 의미적 유사도를 이용한 군집화 방법을 제안한다. 군집화 알고리즘으로써 계층적 응집 군집화 알고리즘[8, 9]을 사용하였는데, 이 알고리즘은 소프트웨어의 모듈화를 목적으로 하는 다양한 연구에서 그 효용성이 검증되었다[8]. 본 논문에서는 군집간의 유사도 대신 의미적 유사도를 사용하여 군집화를 수행하고 특정 임계치를 지정하여 적절한 시점에 군집화를 중단하도록 한다.

3.2.1 의미적 유사도를 이용한 군집화

본 논문에서 제안하는 접근방법은 계층적 응집 군집화 알고리즘을 이용하여 긴 매개변수 리스트를 군집화 한다. 계층적 응집 군집화 알고리즘은 여러 군집간의 유사도를 비교하여 상향식으로 군집화를 수행하는 알고리즘이다. 본 논문에는 군집간의 유사도 대신에 매개변수 사이의 의미적 유사도를 기반으로 계층적 응집 군집화 알고리즘을 수행한다. 또한 계층적 응집 군집화 알고리즘은 최종적으로 하나의 군집만 남기 때문에, 본 논문에서는 특정 임계치를 지정하여 적절한 시점에 군집화를 중단하도록 한다.

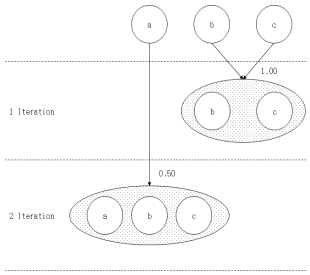
1) 계층적 응집 군집화 알고리즘

계층적 응집 군집화 알고리즘은 대상으로 하는 군집 중에서 유사도가 가장 높은 2개의 군집을 선택하여 새로운 군집을 생성하고 새롭게 생성한 군집과 기존 군집간의 유사도를 재 측정하여 군집화를 한다. 이때 새로 생성한 군집과 기존 군집 사이의 유사도를 계산하기 위하여 갱신 규칙 함수(Update rule function)를 사용하는데, 대표적으로 3가지 규칙 함수(i.e., 단일연결, 완전연결, 평균연결)가 있다[8].

본 논문에서는 평균 연결 함수를 사용하여 유사도를 계산하였다. 다음 표 2과 그림 3은 평균연결 함수를 적용한 계층적 응집 군집화 알고리즘을 사용한 예이다. 표 2는 위 소스코드에서 foo1 메소드가 사용하는 매개변수간의 의미적 유사도 행렬이고, 그림 3는 군집화 과정을 그림으로 표현한 것이다.

<표 2> foo_1 메소드의 매개변수 간에 의미적 유사도 행렬

foo_1	a	b	c
a	-	0.5	0.5
b	-	-	1
c	-	-	-



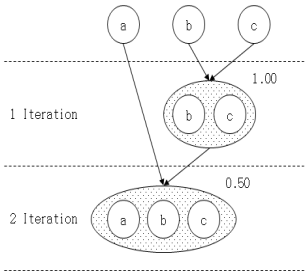
[그림 3] 계층적 응집 군집화 알고리즘의 수행결과

foo1 메소드를 대상으로 군집화하면 총 2단계로 수행된다. 첫 번째 단계는, 매개변수 중 가장 의미적 유사도가 높은 두 매개변수를 선택하여 새로운 군집을 만든다. 그러므로 1의 의미적 유사도를 가지는 b와 c를 선택하여 군집을 만든다. 다음 단계로, 새로 생성된 군집과 기존 군집간의 의미적 유사도를 평균연결 함수로 재 측정한다. a와 b, a와 c의 의미적 유사도를 계산하면, 각각 0.5이며 평균은 0.5이다. 이 값을 토대로 새로 생성한 군집과 b를 하나로 합친다. 결과적으로 3개의 매개변수로 구성된 군집 하나가 생성되고 이 군집의 의미적 유사도는 0.5이다.

위의 예와 같이 계층적 응집 군집화 알고리즘을 수행하면, 최종적으로 하나의 군집만 남는다. 즉, 반복수행의 중단시점을 결정하지 않으면 항상 하나의 군집만 남는 문제가 발생한다. 그러므로 본 논문에서는 적절한 시점에 군집화 수행을 중단하기 위하여 임계치를 이용하였고, 이는 실험을 통해서 선택하였다.

2) 임계치를 이용한 군집화 중단 지점 결정

계층적 응집 군집화 알고리즘을 그대로 적용할 시 최종적으로 하나의 군집만 남는다는 문제가 발생한다. 이를 해결하기 위해서 본 논문에서는 실험을 통해서 선택해 최선의 임계치를 지정하여 적절한 시점에 군집화 수행을 중단하도록 한다. 임계치를 지정하면, 매 반복 단계마다 군집간의 의미적 유사도와 임계치를 비교하여, 임계치 보다 큰 의미적 유사도를 가질 경우, 새로운 군집을 생성한다. 만약 임계치보다 높은 의미적 유사도를 가지는 군집이 존재하지 않으면 군집화를 중단한다. 임계치는 0.0부터 1.0까지 지정 가능하다. 0.0으로 지정하면 기존의 알고리즘 수행과 동일하므로 최종적으로 하나의 군집만 남는다. 반대로 1.0을 지정하면 임계치보다 높은 군집이 존재하지 않으므로 군집화를 중단하여, 결과적으로 매개변수의 수와 동일한 군집이 남는다. 아래의 [그림 4]는 [그림 3]의 예를 수정하여, 임계치를 지정해 적절한 시점에 군집화를 중단하도록 한 결과이다.



[그림 4] 임계치를 적용한 계층적 응집 군집화 알고리즘 수행결과

임계치 0.7을 지정하여 foo1 메소드를 대상으로 군집화하면 총 2단계로 수행한다. 첫 번째 단계는, 임계치 0.7 보다 큰 의미적 유사도를 가지는 b와 c를 선택하여 군집을 생성한다. 다음 단계로, 새로 생성한 군집과 기존 군집간의 의미적 유사도를 평균연결 함수로 재 측정한다. a와 b, a와 c의 의미적 유사도는 각각 0.5과 0.5로 측정된다. 두 경우 모두 임계치 0.7보다 작으므로 군집화를 중단한다. 결과적으로 b와 c를 포함하는 하나의 군집과 a를 포함하는 군집 하나, 총 두 개의 군집이 생성된다. 앞서 정의한 긴 매개변수 리스트의 정의에 따라 foo1 메소드는 긴 매개변수 리스트로 식별된다.

3.3 긴 매개변수 리스트 간소화

이렇게 군집화한 매개변수를 간소화 위해 '매개변수를 객체로 전환(introducing parameter object)' 기법을 사용한다. 이 기법은 두 가지 단계를 거쳐 수행한다. 첫 번째 단계는 하나로 합칠 매개변수를 멤버변수로 가지는 새로운 클래스를 정의한다. 두 번째로, 메소드의 매개변수를 새로 생성한 클래스로 대체하고, 구현부를 그에 맞게 수정한다. 리팩토링의 결과는 아래의 [그림 5, 6]과 같다.

```
public void foo1(int a, ParamObj paramObj) {
    a++;
    paramObj.setB(10);
    paramObj.setC(20);
}
```

[그림 5] '매개변수를 객체로 전환' 기법을 적용한 매개변수 대체

```
public class ParamObj {

    private int b;
    private int c;

    public ParamObj (int b, int c) {
        this.b = b;
        this.c = c;
    }
    public int getB() {
        return b;
    }
    public void setB() {
        this.b = b;
    }
    public int getC() {
        return b;
    }
    public void setC() {
        this.b = b;
    }
}
```

[그림 6] 매개변수 b와 c를 포함하는 새로운 클래스

제안하는 군집화 방법의 결과로써, 매개변수 b와 c는 의미적으로 유사하기 때문에, 두 매개변수를 하나의 객체로 전환할 수 있다. 긴 매개변수 리스트를 간소화하기 위해 첫 번째로, 매개변수 b, c와 동일한 멤버변수를 가지는 ParamObj 클래스를 [그림 6]처럼 새로 정의한다. 두 번째로, foo1 메소드가 사용하는 b와 a 매개변수를 ParamObj 클래스 타입의 paramObj 객체로 전환한다. 마지막으로, [그림 5]와 같이 paramObj 객체의 Getter, Setter를 이용하여 b와 c 데이터에 접근할 수 있도록 foo1 메소드를 수정한다. 위와 같이 본 논문에서 제안하는 군집화 방법을 이용하면, 긴 매개변수리스트를 식별하고, 의미적으로 유사한 매개변수를 하나의 객체로 전환하여 긴 매개변수 리스트를 간소화 시킬 수 있다.

4. 평 가

본 장에서는 제시되는 방안의 평가 계획, 평가 시행방법, 평가 결과가 기술된다.

4.1 평가 계획

제시되는 방안은 3가지 오픈소스 시스템(JHotDraw, JEdit, ArgoUML)을 대상으로 평가되었다. 평가 대상 시스템으로 사용하는 오픈소스의 정보는 다음과 같다.

<표 3> 대상 시스템 정보

	JHotDraw	JEdit	ArgoUML
개발 언어	Java	Java	Java
클래스 수(개)	772	1,612	1,084
라인 수(LOC)	82,099	122,054	117,370
메소드 수(개)	7,297	10,239	7,412

JHotDraw[12]는 자바 기반의 드로잉 프로그램이고 JEdit은 텍스트 편집 프로그램[13], Argo UML은 UML 모델링 도구[14]로서 각기 다른 도메인과 규모를 가지는 시스템이다. 이 시스템을 대상으로, 제시되는 방안을 통해 식별된 긴 매개변수 리스트 중 아래의 조건을 만족하는 긴 매개변수 리스트를 추출하였다.

1) 이미 선정한 메소드의 매개변수의 식별자와 동일한 식별자의 매개변수를 가지는 메소드는 선정에서 제외한다. 그 이유는, 동일한 이름의 매개변수를 사용하는 메소드는 이전과 비슷한 매개변수의 군집 형태를 가질 수 있기 때문이다.
2) Getter/Setter, 생성자는 선정에서 제외한다. 이는 클래스의 변수(Attribute)만을 다루는 메소드이기 때문이다.

추출된 긴 매개변수 리스트를 가지는 메소드들은 아래의 군집화 품질지표를 이용하여 군집화 후의 품질이 측정되었다.

$$CQ(m)=\underset{p_i\in P_m, p_k\in P_m}{\arg max} SS(p_i,p_k)$$

위의 수식을 이용하여, 모든 메소드 m에 대해 m이 가지는 매개변수의 집합 P_m내의 매개변수 쌍 중에 높은 의미적 유사도를 가지는 메소드 순으로 우선순위화 하고 프로젝트 별로 상위 10개씩 추출하였다. 추출된 30개의 메소드는 아래와 같이 선정된 전문가 그룹이 군집화 한다. 전문가 그룹은 자바 개발 실무경험이 있는 세 명(석사 학위를 가지는 10년 이상의 개발자 X 2명, 학사 학위를 가지는 3년 이상의 개발자 X 1명)으로 구성되어 있다. 전문가 그룹은 주어진 메소드의 내용을 파악하여 의미적으로 유사한 매개변수를 군집화 하였고, 전문가 그룹이 군집화한 결과와 제시되는 방안을 통해 자동으로 군집화된 결과가 일치하는 정도를 아래의 $FMeasure$를 기반으로 평가하였다.

$FMeasure$의 정의를 위해 M_{LPL}을 긴 매개변수 리스트를 가지는 메소드의 집합이라 칭하고, M_{LPL}의 원소인 m은 여러 개의 유사한 매개변수의 집합 C_i로 군집화 될 때, 집합 C_i를 원소로 가지는 C_i의 멱집합을 $Clst_m=\{c_0,c_1,\ldots,c_i\}$ 이라고 하자. 그러면 메소드 m에 대한 $FMeasure(m)$은 아래와 같이 정의된다.

- $P(c_i) = \frac{|c_i \cap c'_i|}{|c_i|}$
- $R(c_i) = \frac{|c_i \cap c'_i|}{|c'_i|}$
- $FMeasure(c_i) = 2 * \frac{P(c_i) * R(c_i)}{P(c_i) + R(c_i)}$
- $FMeasure(m) = \frac{\sum_{c_i \in Clst_m} FMeasure(c_i)}{|Clst_m|}$

위의 정의에서 c'_i는 전문가가 군집화한 매개변수를 말한다.

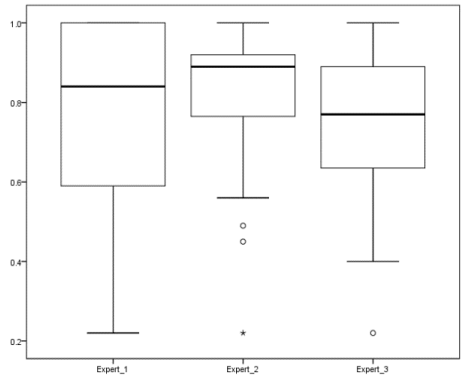
4.2 평가 시행

본격적인 평가에 앞서, 평가에 대한 기본적인 내용을 설명하고, 추출된 30개의 긴 매개변수 리스트를 배포하였다. 전문가 그룹은 긴 매개변수 리스트를 가지는 메소드 내에 있는 여러 정보를 토대로 매개변수를 군집화하고, 자신이 판단한 매개변수의 군집 형태를 그림으로 표시하여 제출한 한다.

한 전문가가 다른 전문가와 평가에 대하여 정보를 공유할 경우, 해당 전문가 간에 유사한 형태의 결과가 발생할 수 있기 때문에, 이를 방지하기 위해 개별적으로 평가를 진행한다. 또한 제공한 평가항목의 내용을 충분히 숙지하고 평가에 응할 수 있도록 하루의 제출기한을 두어 긴 매개변수리스트를 검토하도록 한다.

4.3 평가 결과 분석

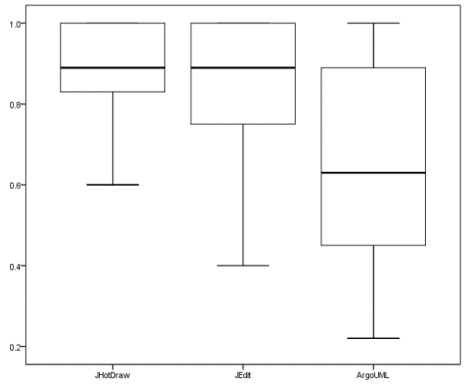
본 실험에서는 긴 매개변수 리스트의 군집화를 위해 임계치를 적용하였다 (τ = 0.5). 자동으로 군집화된 긴 매개변수의 리스트는 아래의 <표 4>와 같다. 표 4에 나타난 매개변수 군집화 결과와 전문가 그룹이 수작업으로 군집화한 결과를 비교하였다. 먼저 3명의 전문가 모두가 공통적으로 30개의 긴 매개변수 리스트 중 6개는 군집화가 필요 없다 판단하였다. 나머지 24개의 매개변수 리스트를 자동으로 군집화한 결과와 각 전문가가 수작업으로 군집화 한 결과와의 일치율을 나타내는 F-Measure의 분포는 아래와 같다.



[그림 7] 전문가 별 평가 결과

각각 세 명의 전문가가 군집화한 결과는 자동으로 군집화된 결과와 비교하여 평균적으로 0.77, 0.83, 0.74의 F-Measure로 측정되었다. 한편 전문가별로 추출한 샘플의 F-Measure가 차이가 있기 때문에 F-Measure의 유의수준 0.05 에서 통계적인 신뢰구간을 추정하였다. 추정된 F-Measure의 신뢰구간은 0.72~0.82로 추정되었는데, 이는 추출된 샘플에서 측정된 F-Measure가 평균 0.78을 보였지만 결과의 평균과 편차를 고려하였을 때, 그 평균이 95%의 확률로 0.72~0.82 구간 내에 존재한다는 것을 나타낸다.

아래의 그림은 프로젝트 별 F-Measure의 분포를 보여준다.



[그림 8] 프로젝트 별 평가 결과

#	프로젝트	메소드	군집결과
1	JH	createRect	[doc], [attributes], [x, y, width, height], [rx, ry]
2	JH	cap	[p1, p2], [radius]
3	JH	encodeBytes	[source], [off, len], [options]
4	JH	createCircle	[doc], [attributes], [cx, cy, r]
5	JH	reparameterize	[d, bezCurve, first, last], [u]
6	JH	draw	[g], [f], [p1, p2]
7	JH	createColorWheelChooser	[sys], [angularIndex, radialIndex, verticalIndex, flipX, flipY], [type]
8	JH	groupFigures	[view, group], [figures]
9	AG	connect	[fromPort, toPort], [edgeType]
10	AG	buildConnection	[graphModel], [edgeType], [sourceFig, destFig]
11	AG	getPointOnPerimeter	[rect], [direction], [xOff, yOff]
12	AG	fireTreeNodesChanged	[source], [path, childIndices, children]
13	AG	startElement	[uri, localname, qname, attributes]
14	AG	setBounds	[xInc, yInc, w], [concurrency]
15	JE	hbAssignCodes	[code], [minLen, maxLen, alphaSize], [length]
16	JE	paintValidLine	[gfx, y], [screenLine, physicalLine], [start, end]
17	JE	parseStyle	[str, defaultFgColor, family, size], [color]
18	JE	getWriteEncodingErrorMessage	[encodingName, encoding], [line], [lineIndex]
19	JE	fireContentRemoved	[startLine, numLines], [length], [offset]
20	JE	extendSelection	[offset], [end, extraStartVirt, extraEndVirt]
21	JE	prepareBackupFile	[path], [backups, backupPrefix, backupSuffix, backupTimeDistance, backupDirectory]
22	JE	nextMatch	[text, firstTime, reverse], [start, end]
23	JE	doFontSubstitution	[subst, start, end], [mainFont, text]
24	JE	paintScreenLineRange	[textArea], [firstLine, lastLine], [gfx, y, lineHeight]

<표 4> 긴 매개변수 리스트와 군집결과

세 프로젝트의 평균 F-Measure는 각각 0.83, 0.83, 0.69로서 JHotDraw와 JEdit이 더 높은 결과를 보였다. 프로젝트 별 F-Measure의 편차가 통계적으로 무의미하고, 프로젝트간의 차이가 없다는 것을 증명하기 위해 유의 수준을 0.05로 하여 대응 표본 T 검정을 시행 하였다. 검정 결과는 아래의 <표 5>와 같다.

표 5 프로젝트 별 대응 표본 T검정 결과

	대응차					유의 확률
	평균	표준 편차	표준 오차	차이의 95% 신뢰구간		
				하한	상한	
JH-JE	0.04	0.16	0.04	-0.04	0.11	0.295
AG-JE	-0.16	0.40	0.09	-0.34	0.02	0.077
JH-AG	0.14	0.34	0.07	0.01	0.28	0.038

위의 검정결과에서 보는 바와 같이 JHotDraw와 JEdit 그리고 ArgoUML과 JEdit의 평균의 차이는 각각 0.04, 0.16의 차이가 나지만 유의 확률이 미리 설정된 0.05보다 크기 때문에 통계적으로 차이를 인정할 수 없으며, 그러므로 두 쌍의 프로젝트 간의 평균의 차이는 의미가 없다고 할 수 있다. 그러나 JHotDraw와 ArgoUML의 경우 유의 확률이 0.05미만으로 차이가 있다고 할 수 있다. 이러한 차이는 두 프로젝트의 품질과 관계가 있다고 할 수 있다. 왜냐하면 자동으로 군집화하기 위해 사용한 매개변수의 식별자간의 의미적 유사도는 메소드 내에 선언된 변수가 많이 등장할수록 높은 값을 가지도록 설계되어 있지만, 그 결과가 전문가들이 판단하기에는 의미적 유사도가 적다고 판단하였기 때문이다.

결과적으로 이는 ArgoUML의 메소드들이 JHotDraw의 메소드들 보다 의미적인 응집력이 떨어진다고 할 수 있다. 표 3에서 보는 바와 같이 ArgoUML의 경우 제시되는 방안을 통해 식별된 10개의 긴 매개변수 리스트 중 4개(40%) 가 전문가들의 판단으로는 군집화할 수 없는 매개변수로 판별되어 잘못 식별된 것으로 평가되었는데, 이 또한 ArgoUML의 품질이 상대적으로 떨어지기 때문일 것으로 판단된다. 우리는 이러한 차이를 검증해 보기 위해 두 프로젝트 내에 사용된 텍스트를 이용한 의미적 응집력 지표인 C3[15] 지표를 이용하여 두 프로젝트의 품질을 비교하였다. 측정결과 JHotDraw의 C3는 약 0.66으로 측정된 반면 ArgoUML은 0.53으로 측정되었는데, 이러한 사실은 프로젝트의 품질이 낮을수록 본 제안방법의 정확도도 떨어진다는 사실을 뒷받침해준다.

5. 결 론

본 논문은 소프트웨어의 유지보수성을 떨어뜨리는 요소 중 하나인 나쁜 냄새의 한 종류인 긴 매개변수 리스트 문제를 해결하는 방안으로 매개변수간의 의미적 유사도를 이용한 군집화 방법을 제안했다. 본 논문에서 제안하는 방법은 세 개의 오픈소스를 대상으로 평가되었다. 평가는 제시되는 방안을 통해 식별한 긴 매개변수 리스트 30개를 대상으로 세 명의 전문가 그룹이 수작업으로 군집화하고, 이렇게 군집화한 결과와 제시되는 방안을 통해 자동으로 군집화한 결과를 비교하였다. 비교결과 평균 0.78정도의 정확도를 보였고, 통계적으로 F-Measure의 평균의 신뢰구간은 0.72~0.82내에 존재하는 것으로 추정되었다. 또한 프로젝트간의 F-Measure를 비교한 결과 ArgoUML이 가장 정확도가 떨어지는 것으로 나타났다. 텍스트 기반의 클래스 응집도 지표를 기반으로 ArgoUML의 품질을 측정해 본 결과 ArgoUML의 품질이 다른 시스템에 비해서 떨어지기 때문인 것으로 나타났다. 따라서 본 논문에서 제안되는 방안은 메소드의 의미적인 응집도가 너무 낮으면 정확도가 떨어지는 한계를 가지고 있다고 할 수 있다. 향후 연구로서 메소드의 의미적인 응집도를 고려하여 정확도를 향상시키기 위한 연구를 진행해 보고자 한다.

참 고 문 헌

[1] Y. Ren, T. Xing, X. Chen, X. Chai, “Research on Software Maintenance Cost of Influence Factor Analysis and Estimation Method”, 2011 3rd International Workshop on Intelligent Systems and Applications, pp.1-4, May 2011

[2] M. Fowler, “Refactoring: Improving the Design of Existing Code”, Addison Wesley, 1999

[3] C. Zhao, J. Kong, K. Zhang, “Program Behavior Discovery and Verification: a Graph Grammar Approach”, IEEE Transactions on Software Engineering, Vol.32, Issue.3, pp.431-448, May-June 2010

[4] M. Arora, Dr. S. S. Sarangdevot, Vikram Singh Rathore, J. Deegwal , S. Arora, “Refactoring, Way for Software Maintenance”, IJCSI International Journal of Computer Science Issues, Vol.8, Issue.2, March 2011

[5] Kataoka. Y, Imai. T, Andou. H, Fukaya T, “A Quantitative Evaluation of Maintainability Enhance ment by Refactoring”, Proceedings of the International Conference on Software Maintenance, pp.576-585, 2002

[6] Eva van Emden, Leon Moonen, “Java Quality Assurance by Detecting Code Smells”, Proceedings of the Ninth Working Conference on Reverse Engineering, pp.97-106, 2002

[7] Meananeatra. P, “Identifying Refactoring Sequences for Improving Software Maintainability”, Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering, pp.406-409, Sept 2012

[8] O. Maqbool, H.A. Babri, “Hierarchical Clustering for Software Architecture Recovery”, IEEE Transactions on Software Engineering, Vol.33, Issue.11, pp.759-780, Nov 2007
[9] O. Maqbool, H.A. Babri, “The Weighted Combined Algorithm: A Linkage Algorithm for Software Clustering”, Proceedings of the Eighth European Conference on Software Maintenance and Reengineering, pp.15-24, March 2004
[10] Robert C.Martin, “Clean Code: A Handbook of Agile Software Craftsmanship”, Prentice Hall, 2008
[11] T. Bayes, R. Price, “An Essay towards solving a Problem in the Doctrine of Chances. By the late Rev. Mr. Bayes, communicated by Mr. Price, in a letter to John Canton, M. A. and F. R. S.”, 1763
[12] JHotDraw, http://www.jhotdraw.org/
[13] JEdit, http://www.jedit.org/index.php
[14] ArgoUML, http://argouml.tigris.org/
[15] Marcus. A, Poshyvanyk. D, “The conceptual cohesion of classes”, Proceedings of the 21st IEEE International Conference on Software Maintenance, pp.133-142, Sept 2005

저자 소개

함 동 화

2012년 명지대학교 컴퓨터공학과(학사)
2013년~ 서강대학교 컴퓨터공학과 석사과정
관심분야는 소프트웨어 유지보수, 소프트웨어 품질

이 준 하

2007년 단국대학교 컴퓨터과학과(학사)
2010년 서강대학교 컴퓨터공학과(석사)
2010년~ 서강대학교 컴퓨터공학과 박사과정
관심분야는 소프트웨어 유지보수, 소프트웨어 품질

박 수 진

1995년 경북대학교 컴퓨터공학과(학사)
2000년 서강대학교 정보통신대학원(석사)
2008년 서강대학교 컴퓨터공학과(박사)
1995년~1999년 (주)LG-CNS 근무
2000년~2002년 한국래쇼날 소프트웨어 선임 컨설턴트
2010년~현재 서강대학교 기술경영전문대학원 조교수
관심분야는 소프트웨어 재사용, 요구공학, 소프트웨어 아키텍처, 적응형 소프트웨어

박 수 용

1986년 서강대학교 컴퓨터과학(학사)
1988년 Florida State University, 컴퓨터정보과학(석사)
1995년 George Mason University, 정보기술(박사)
1998년~서강대학교 컴퓨터공학과 교수
관심분야는 소프트웨어 요구사항, 자가적응형 소프트웨어, 소프트웨어 품질

회원 가입 안내 및 회비 납부 요령

한국정보과학회 소프트웨어공학소사이어티는 회원 여러분에게 유익한 정보를 제공해 드리기 위하여 보다 충실한 내용의 논문지 발간 배포, 그리고 국제·국내 학술발표회 및 초청강연회와 단기강좌 등의 여러 가지 사업들을 추진하고 있습니다.

소프트웨어공학 소사이어티의 가입을 통해 정보 및 기술 교류, 그리고 인적 네트워크의 구성에 참여하시기를 기대합니다. 회원 가입을 위하여 아래의 회비 안내를 참고하시어 회비를 납부하시고, 다음 쪽의 입회원서를 작성하시어 아래 소프트웨어공학소사이어티 주소로 보내주시거나 팩스 또는 이메일을 통해 보내어 주시기 바랍니다.

한국정보과학회 소프트웨어공학소사이어티 연락처는 아래와 같습니다.

◈ **소프트웨어공학소사이어티**

주 소 : (우) 110-743 서울특별시 종로구 홍지문 2길 20, 상명대학교 소프트웨어대학관 G511 한국정보과학회 소프트웨어공학소사이어티 한혁수

전 화 : 02-2287-5033

팩 스 : 02-2287-0049

전자우편 : hshan@smu.ac.kr(한혁수 교수), jungwony@ajou.ac.kr(이정원 교수)

홈페이지 : http://www.sigse-kiss.or.kr/

◈ **회비 안내**

회원구분	•학생회원 : IT 분야 학과 또는 관심 있는 학생 •정회원, 종신회원 : IT 분야 종사자
가입비	학생회원, 정회원 : 20,000원, 종신회원 : 200,000원
년회비	학생회원, 정회원 : 20,000원

- 회비납부 방법

(1) 무통장입금 또는 계좌이체 후 입회원서 발송

: 계좌 번호 : 제일은행 150-20-358028 (이병정)

(2) 소사이어티 주관 학술행사 개최시, 행사장 당일 가입 및 납부 가능

개인회원용 입회원서

회원구분	학생회원 ( ) 정회원( ) 종신회원( )				
성명	한글		생년월일		
	영문				
연락처	직장전화		휴대전화		
	e-mail				
주소	직장명/부서		직급		
	직장주소	(우)			
학력	학사	년 월 - 년 월	대학교	과	
	석사	년 월 - 년 월	대학원	과	
	박사	년 월 - 년 월	대학원	과	
관심분야					

본인은 한국정보과학회 소프트웨어공학소사이어티의 취지에 찬성하여 회원으로 가입하고자 이에 입회원서를 제출합니다.

년 월 일

신 청 인: (인)

한국정보과학회 소프트웨어공학 소사이어티 회장 귀하

논문지 논문 모집 (Call for Papers)

한국정보과학회 소프트웨어공학 소사이어티에서는 매년 4회에 걸쳐 '소프트웨어공학 소사이어티 논문지'를 발간하고 있습니다. 이 논문지에는 소프트웨어공학 전반에 걸친 연구논문과 산업계 논문을 게재해 오고 있습니다. 다음과 같은 소프트웨어공학 주제에 관련된 논문을 모집하고 있으니 학계와 산업계의 여러분들의 적극적인 논문투고를 바랍니다.

◈ 논문 주제

- ○ 소프트웨어 설계 및 아키텍처
- ○ 소프트웨어 재사용 및 프로덕트라인
- ○ 요구공학
- ○ 소프트웨어 품질 및 테스트
- ○ 관리 및 프로세스
- ○ 소프트웨어 정형 기법
- ○ 서비스기반 소프트웨어 개발
- ○ 임베디드, 모바일, 웹기반 소프트웨어 개발
- ○ 기타 소프트웨어 응용 (국방, 자동차, 조선 등의 분야)

◈ 논문심사

- ○ 투고된 논문은 편집위원회에서 심사 선정하며, 필요 시 외부 심사위원을 위촉하여 심사를 합니다. 제출된 논문은 반환하지 않습니다.
- ○ 심사료 및 게재료: 없음

◈ 논문 제출

- ○ 소프트웨어공학 소사이어티의 논문지 투고 양식(http://www.sigse-kiss.or.kr/)을 사용하며, 논문의 분량은 10장으로 제한합니다.
- ○ 논문지 투고규정에 따라 작성된 심사용 논문파일은 온라인투고시스템을 통하여 투고하시기 바랍니다.

◈ 문의처 (편집위원회)

- ○ 편집이사 : 강성원 교수 (KAIST, 042-350-3512, sungwon.kang@kaist.ac.kr)
- ○ 편집이사 : 윤회진 교수 (협성대학교, 031-299-0841, hjyoon@uhs.ac.kr)
- ○ 편집이사 : 이관우 교수 (한성대학교, 02-760-5864, kwlee@hansung.ac.kr)
- ○ 편집이사 : 채흥석 교수 (부산대학교, 051-510-3517, hschae@pusan.ac.kr)
- ○ 편집이사 : 김문주 교수 (KAIST, 042-350-3543, moonzoo@cs.kaist.ac.kr)
- ○ 편집위원 : 김정아 교수 (관동대학교, 033-649-7801, clara@kwandong.ac.kr)
- ○ 편집위원 : 김현수 교수 (충남대학교, 042-821-6657, hskim401@cnu.ac.kr)
- ○ 편집위원 : 이우진 교수 (경북대학교, 053-950-6378, woojin@mail.knu.ac.kr)
- ○ 편집위원 : 박수진 교수 (서강대학교, psjdream@sogang.ac.kr)
- ○ 편집위원 : 이지현 교수 (대전대학교, jihyun30@dju.ac.kr)
- ○ 편집위원 : 최종무 교수 (단국대학교, choijm@dankook.ac.kr)
- ○ 편집위원 : 김태호 박사 (ETRI, taehokim@etri.re.kr)

투 고 요 령

1. 소프트웨어공학소사이어티 논문지에 실리는 원고는 주제 논문, 일반 논문, 산업체 기고 등으로 구분하며 다음과 같은 분야에 대하여 모집한다.
   가. 소프트웨어공학 및 그 응용분야에 대한 연구결과
   나. 강좌 및 관련 교육사항 소개 (목적, 과정, 일정, 대상, 특징)
   다. 소프트웨어 도구 및 방법론 소개 (가격, 특징, 종류, 적용사례)
   라. 소프트웨어 산업에 대한 학계, 업계의 주요 관심사
   마. 기타 관련 사항

2. 투고자는 원칙적으로 본 소사이어티의 회원으로 한다. 다만 공동 또는 초청 기고자는 예외로 한다.

3. 논문은 원칙적으로 한글로 작성한다.

4. 원고는 한글(hwp), 워드(MS Word), PDF 형식 중 하나를 택하여 A4용지에 작성하며, 그림과 표를 포함하여 10쪽 이내로 한다.

5. 논문 내용에 직접 관련이 있는 문헌에 대해서는 이들 문헌에 관련이 있는 본문 중에 참고문헌 번호를 쓰고 그 문헌을 참고문헌 난에 인용 순서대로 기술한다. 참고문헌은 학술지의 경우 저자, 제목, 학술지명, 권, 호, 쪽수, 발행 연도의 순서로, 단행본은 저자, 서명, 쪽수, 발행처, 발행 연도의 순서로 기술한다.

   [1]Cole, R., "Parallel Merge Sort", SIAM Journal of Computing, vol.17, No.4, pp.770-785, 1988.
   [2]김수형, 강명호, 조형재, 송주석. "안전하고 효율적인 침입자 역추적 시스템", 정보과학회논문지, 제25권, 제10호, pp.1123-1131, 1998

6. 논문은 소프트웨어공학 소사이어티(http://www.sigse-kiss.or.kr/)의 온라인 투고 시스템을 통해 제출한다.

7. 논문투고신청서를 반드시 작성하여 이메일(hjyoon@uhs.ac.kr)로 제출한다.

7. 원고 접수는 수시로 하며, 접수일은 온라인 접수일로 한다.

8. 기타 자세한 사항은 한국정보과학회 논문지 투고 요령을 따른다.

한국정보과학회 소프트웨어공학소사이어티 임원명단

구분		성 명	소속기관명	E-Mail
회 장		한 혁 수	상명대학교	hshan@smu.ac.kr
부회장 (기획)		권 기 현	경기대학교	khkwon@kyonggi.ac.kr
부회장 (편집)		강 성 원	KAIST	sungwon.kang@kaist.ac.kr
부회장 (학술)		홍 장 의	충북대학교	jehong@chungbuk.ac.kr
부회장 (조직)		이 병 걸	서울여자대학교	byongl@swu.ac.kr
부회장 (협력)		전 진 옥	비트컴퓨터	jojeon@bit.co.kr
운영위원회	총무이사	이 정 원	아주대학교	jungwony@ajou.ac.kr
		유 준 범	건국대학교	jbyoo@konkuk.ac.kr
	기획이사	이 병 정	서울시립대학교	bjlee@uos.ac.kr
		서 주 영	아주대학교	jyseo@ajou.ac.kr
	조직이사	백 종 문	KAIST	jbaik@kaist.ac.kr
		김 영 철	홍익대학교	bob@hongik.ac.kr
	학술이사	인 호	고려대학교	hoh_in@korea.ac.kr
		고 인 영	KAIST	iko@kaist.ac.kr
	편집이사	윤 회 진	협성대학교	hjyoon@uhs.ac.kr
		이 관 우	한성대학교	kwlee@hansung.ac.kr
		채 흥 석	부산대학교	hschae@pusan.ac.kr
		김 문 주	KAIST	moonzoo@cs.kaist.ac.kr
	홍보이사	조 은 숙	서일대학교	escho@seoil.ac.kr
		이 찬 근	중앙대학교	cglee@cau.ac.kr
		박 용 범	단국대학교	ybpark@dankook.ac.kr
	협력이사	이 세 영	NIPA	sarahlee230@gmail.com
감사		차 성 덕	고려대학교	scha@korea.ac.kr
		이 상 은	NIPA	selee@nipa.kr
자문위원회		강 교 철	포항공과대학교	kck@postech.ac.kr
		권 용 래	KAIST	kwon@cs.kaist.ac.kr
		성 기 수	KISTI 고문	Kss13@truefriend.com
		배 두 환	KAIST	bae@se.kaist.ac.kr
		신 규 상	ETRI	gsshin@etri.re.kr
		양 승 민	숭실대학교	smyang@ssu.ac.kr
		우 치 수	서울대학교	wuchisu@selab.snu.ac.kr
		이 경 환	중앙대학교	kwlee@object.cau.ac.kr
		이 단 형	KAIST	danlee@cs.kaist.ac.kr
		정 기 원	숭실대학교	chong@comp.ssu.ac.kr
		박 수 용	서강대학교	sypark@sogang.ac.kr
		황 선 명	대전대학교	sunhwang@dju.kr
		전 진 옥	비트컴퓨터	jojeon@bit.co.kr
		김 수 동	숭실대학교	sdkim777@gmail.com
		이 긍 해	항공대학교	khlee@kau.ac.kr
		최 병 주	이화여자대학교	bjchoi@ewha.ac.kr

구분	성 명	소속기관명	E-Mail
이사	김 정 아	관동대학교	clara@kwandong.ac.kr
	남 영 광	연세대학교	yknam@yonsei.ac.kr
	박 수 진	서강대학교	psjdream@sogang.ac.kr
	염 근 혁	부산대학교	yeom@pusan.ac.kr
	오 재 원	카톨릭대학교	jwoh@catholic.ac.kr
	윤 희 병	국방대학원	hbyoon37@hanmail.net
	이 우 진	경북대학교	woojin@knu.ac.kr
	이 은 석	성균관대학교	leees@skku.edu
	이 석 원	아주대학교	leesw@ajou.ac.kr
	이 은 주	경북대학교	ejlee@knu.ac.kr
	전 태 웅	고려대학교	jeon@korea.ac.kr
	정 인 상	한성대학교	insang@hansung.ac.kr
	최 승 훈	덕성여자대학교	csh@duksung.ac.kr
	최 종 무	단국대학교	choijm@dankook.ac.kr
	최 호 진	KAIST	hojinc@kaist.ac.kr
	현 창 문	탐라대학교	cmhyun@tnu.ac.kr
	계 승 교	삼성SDS	seankae@samsung.com
	권 경 룡	국방기술품질원	ka-ja17@hanmail.net
	권 원 일	STA컨설팅	wonil@softwaretesting.co.kr
	김 상 기	현대자동차	sangkikim@hyundai-motor.com
	김 진 태	SEEG	jtkim@swexpertgroup.com
	민 경 오	LG전자	davidmin@lge.com
	민 상 윤	솔루션링크	sang@sol-link.com
	박 복 남	핸디피엠지	pbnkhb@hitel.net
	박 찬 규	국방SW산학연합회	milspark@hotmail.com
	배 현 섭	슈어소프트테크	hsbae@suresofttech.com
	손 세 창	인천공항공사	scsohn@airport.kr
	손 진 규	삼성탈레스	Jinkyu.son@samsung.com
	신 석 규	SW시험인증센터	skshin@tta.or.kr
	양 상 욱	KAI	sangyang@koreaaero.com
	유 영 수	현대엠앤소프트	ysyoo@hyundai-mnsoft.com
	윤 태 권	한국SW기술진흥협회	tkyune@empal.com
	윤 형 진	케피코	Hyoungjin.yoon@kefico.co.kr
	이 근	삼성전자	gskeun.lee@samsung.com
	이 성 남	방위사업청	dapalee@korea.kr
	이 우 복	삼성전자	woobok.yi@samsung.com
	이 장 수	한국원자력연구원	jslee@kaeri.re.kr
	장 주 수	모아소프트	jsjang@moasoftware.co.kr
	정 연 대	N3SOFT	ydchung@n3soft.co.kr
	조 병 인	국방과학연구소	chobyun@dreamwiz.com
	조 상 윤	다한테크	sycho@dahan.co.kr

소프트웨어공학소사이어티 논문지 제26권 제4호 (통권 100호)

발 행 일 ॥ 2013년 12월 31일
발 행 인 ॥ 한혁수
편 집 인 ॥ 강성원
발 행 처 ॥ 사단법인 한국정보과학회 소프트웨어공학소사이어티
연 락 처 ॥ 서울특별시 종로구 홍지문 2길 20, 상명대학교 소프트웨어 대학관 G511
한국정보과학회 소프트웨어공학소사이어티 한혁수
전 화 : 02-2287-5033, 팩 스 : 02-2287-0049
전자우편 : hshan@smu.ac.kr(한혁수 교수), jungwony@ajou.ac.kr(이정원 교수)
홈페이지 : http://www.sigse-kiss.or.kr/
인 쇄 처 ॥ (주)참기획 (전화 : 042-861-6380, 팩스 : 042-861-6381)