韓國IT서비스學會誌
第17卷 第3號
2018年 9月, pp.171-189

Journal of Information Technology Services
https://doi.org/10.9716/KITS.2018.17.3.171

사회연결망 분석과 자료포락분석 기법을 이용한 소프트웨어 함수 우선순위 분석 연구

허상무* · 김우제**

Priority Analysis for Software Functions Using Social Network Analysis and DEA(Data Envelopment Analysis)

Sang Moo Huh* · Woo Je Kim**

■ Abstract ■

To remove software defects and improve performance of software, many developers perform code inspections and use static analysis tools. A code inspection is an activity that is performed manually to detect software defects in the developed source. However, there is no clear criterion which source codes are inspected. A static analysis tool can automatically detect software defects by analyzing the source codes without running the source codes. However, it has disadvantage that analyzes only the codes in the functions without analyzing the relations among source functions. The functions in the source codes are interconnected and formed a social network. Functions that occupy critical locations in a network can be important enough to affect the overall quality. Whereas, a static analysis tool merely suggests which functions were called several times. In this study, the core functions will be elicited by using social network analysis and DEA (Data Envelopment Analysis) for CUBRID open database sources. In addition, we will suggest clear criteria for selecting the target sources for code inspection and will suggest ways to find core functions to minimize defects and improve performance

Keyword : Software Defect, Social Network Analysis, DEA(Data Envelopment Analysis), Code Inspection, Static Analysis Tool, Cubrid Database Software

Submitted : April 18, 2018 1st Revision : May 15, 2018 Accepted : June 14, 2018

* 서울과학기술대학교 IT정책대학원 산업정보시스템 박사
** 교신저자, 서울과학기술대학교 글로벌융합산업공학과 교수

1. 서 론

소프트웨어 개발 조직에서는 결함을 제거하기 위하여 개발 초기에는 코드 인스펙션을 수행하고, 개발 중에는 버그를 수정하며, 개발 후에는 테스트를 통하여 결함을 제거한다. 개발 초기에 코드 인스펙션을 수행하여 결함을 제거하는 것이, 개발 비용을 절감하고 개발 기간도 단축시킬 수 있는 가장 좋은 방법으로 알려져 있다(Rico, 2002). 그렇지만, 코드 인스펙션을 수행할 때, 어느 소스를 선정해야 하는지에 대한 명확한 기준은 없다. 근래 IT 시스템은 규모가 점점 커지고 복잡해지므로, 사람이 육안으로 하는 코드 인스펙션을 수행하기가 어려워지고 있다. 이런 문제를 해결하기 위하여, 정적분석도구를 이용하여 결함이 발생할 가능성이 높은 코드를 제시한다. 소프트웨어 소스의 함수들은 서로 호출하는 연결망 구조를 형성하고 있고, 중요한 위치를 점유하고 있는 함수가 있을 것이다. 만약, 결함이나 성능에 관련된 중요한 함수를 찾아낼 수 있다면, 그 함수들을 집중적으로 관리하여 결함을 제거하거나 성능을 개선시키면, 적은 비용과 노력으로 상당한 효과를 거둘 수 있을 것이다. 하지만, 코드 인스펙션이나 정적분석도구에서는 소스 함수의 연결망 구조를 분석하지는 않는다. 정적분석도구는 함수가 몇 번 호출(Fan-in)되고, 다른 함수를 몇 번 호출(Fan-out)하는 지에 대하여 분석할 수 있다. 그렇지만, 이 정보는 호출이 많이 된 함수는 중요한 함수이고, 호출이 적게 된 함수는 중요하지 않은 함수라는 잘못된 정보를 전달할 수 있다.

사회연결망 분석은 복잡하게 연결된 네트워크 구조에 대하여 여러 가지 지표를 이용하여 중요한 노드와 링크를 도출할 수 있다. 소프트웨어의 함수도 네트워크 적으로 형성되어 있으므로, 사회연결망 지표를 이용하면 원하는 중요한 함수를 도출할 수 있을 것이다. 그렇지만 함수의 연결망 구조를 파악할 수 없을 정도로 소스가 방대하다면, 어느 사회연결망 지표로 분석해야 하는 지 알 수 없는 딜레마에 빠지게 된다.

자료포락분석은 복수의 입력요소와 산출요소를 이용하여 의사결정단위들의 상대적인 효율성을 측정하는 기법이다. 사회연결망 기법의 여러 지표를 이용하여 도출된 핵심 함수에 대하여 자료포락분석 기법을 이용하면, 함수간의 상대적인 효율성을 측정하여 최종적인 핵심함수를 도출할 수 있을 것이다.

이런 방법으로 도출된 함수는 코드 인스펙션을 수할 때, 소스 함수를 선정하는 기준으로 사용할 수 있을 것이며, 정적분석도구에 사회연결망 분석 지표와 자료포락분석기법을 내장하면, 명확한 기준으로 핵심함수를 제시할 수 있을 것이다. 핵심함수를 제시할 수 있다면, 함수에 내재된 결함을 집중적으로 제거하여 실행 결함을 감소시키고, 성능을 개선하면 전체적인 성능이 향상될 것이다. 하지만, 선행연구를 탐색한 결과, 소스 코드에 대한 핵심함수를 도출하는 연구나 사회연결망 기법을 이용하여 핵심 함수를 도출하는 연구는 미흡한 것으로 파악되었다. 이에 본 연구에서는 공개 소프트웨어 소스를 대상으로 사회연결망 분석기법의 지표를 이용하여 핵심 함수를 도출하고 그 의미를 분석하도록 한다.

본 연구의 구성은 제 2장에서는 선행연구와 이론적 배경을 소개하고, 제 3장에서는 연구 설계방법을 살펴보고 제 4장에서는 실험 결과를 분석한다. 마지막으로 제 5장에서는 결론과 향후 추가 연구과제에 대하여 논의한다.

2. 선행연구 고찰 및 이론적 배경

2.1 정적분석도구 및 선행연구 고찰

정적분석도구는 소프트웨어의 소스를 분석하여 결함이 발생할 가능성이 높은 코드를 자동으로 도출하는 데 사용되고 있다. 더 나아가, 소스 구조를 파악할 수 있도록, 함수들 간의 연관 관계를 시각화하는 도구도 있다. 그렇지만, 정적분석도구에서 제공하는 함수 간의 연결 정보는, 함수가 몇 번 호출되는 지에 대한 정보만을 제시하고 있다.

사회연결망 기법을 이용한 선행 연구로는, 건설 분야에서는 부적합 보고서를 수집하여 결함에 대한 근본 원인과 직접원인을 수집하여 사회연결망 분석을 수행한 결과, 8가지의 근본원인 중 2가지를 제거하면 결함의 90%를 예방할 수 있다는 연구가 있다(Van Den Brink and Han, 2015). 정보시스템 분야에서는 4개의 공개 소프트웨어 커뮤니티에서 보고된 결함에 대하여, 사회연결망 분석을 통하여, 결함의 분포 및 주요 결함을 도출한 연구가 있다(Zanetti et al., 2013). 또한, Windows 2003서버를 배포한 후 6개월 동안 발생한 결함을 수집하여, 사회연결망 분석을 통하여 결함이 존재하는 DLL 파일에 대하여 결함의 의존상태를 연구한 논문이 있다(Nagappan and Ball, 2007). 또한, DLL 파일에 대한 결함의 의존상태를 분석한 후, 결함을 예측하여 효과적으로 자원을 할당하는 연구가 있다(Zimmermann and Nagappan, 2008). 이와 같이 건설 분야에서 주요 결함에 대한 연구와, 정보시스템 분야에서 공개소프트웨어 결함의 분포와 주요 결함을 도출한 연구와, DLL 파일들 간의 결함의 의존상태를 분석하여 결함을 예측하는 연구가 있는 것으로 파악되었다. 하지만 이 연구들은 소스 코드 내의 함수들 간의 연관성을 분석하는 연구는 아니다.

소스코드를 이용한 논문으로는 소스 코드 내의 제어흐름, 데이터 흐름 등을 연구한 논문이 상당수 존재하였다. 1970년대의 제어흐름에 대한 연구(Allen, 1970)부터 MacCabe 회전복잡도(McCabe, 1976; Hall and Preiser, 1984) 논문, 소스 프로그램 함수에 대하여 연관 관계를 그래프로 표출하는 연구(Ferrante et al., 1987), 비즈니스 흐름을 기반으로 소스 코드의 제어 흐름을 분석하여 결함을 신속하게 발견할 수 있다는 연구가 있다(Vanhatalo et al., 2007). 이러한 유형의 연구는 과거부터 지금까지 수많은 연구가 진행된 것으로 파악되었고, 상당히 의미가 있는 연구이지만, 이 연구의 단점은 프로그램 소스 코드 흐름과 데이터 흐름만을 분석하였고, 함수 간의 연관관계를 분석하여 핵심함수를 도출하는 연구는 아니다.

소스 프로그램 함수의 외부 흐름을 분석하여 복잡도를 산정한 연구가 있다. 이 연구는 함수로 입력되는 호출 수와 다른 함수를 호출하는 횟수를 이용하여 어느 함수가 결함에 취약한 지 산정하였다(Henry and Kafura, 1981; Singer et al., 2010). 하지만 이 연구에서 이용한 호출 지표는 이미 정적 분석도구에서 사용하고 있는 연결정도 지표이며, 사회연결망의 중심성(Centrality) 지표 중에 내향연결정도(In-degree)와 외향연결정도(Out-degree) 지표로서, 사회연결망 관점에서는 매우 초보적인 수준의 연구로서, 사회연결망 기법을 이용한 연구라고 하기에는 미흡하다고 할 수 있다.

소스 코드의 함수를 이용한 다른 연구로는 소스 가시화 연구가 있다. 정보시스템에 대한 용이한 개발과 유지보수를 위하여, 소스의 구조를 그래프로 가시화해 주는 연구가 있다(Lommerse et al., 2005). 또한, 소스 함수의 호출 정도를 이용하여 소스를 자동으로 클러스터링하는 연구도 있다(Mancoridis et al., 1998). 대규모 소스인 경우에, 함수들 간의 관계를 그래프를 표시하는 데 너무 많은 시간이 소요되므로 소스 간의 의존성을 분석하여 그래프 표출 속도를 개선한 연구도 있다(Pinzger et al., 2008). 함수의 연결 구조 가시화 기능은 소스의 연결을 육안으로 식별할 수 있는 유용한 기능으로 판단되지만, 이 연구들은 사회연결망 기법을 이용하여 핵심 함수를 도출하는 연구는 아니다.

이상과 같이 선행연구를 살펴본 결과, 사회연결망 분석을 통하여 핵심 결함을 도출한 연구가 있고, 소스코드 자체를 이용하여 제어흐름과 데이터흐름을 연구하는 논문이 있었으며, 복잡해지는 소스 코드를 좀 더 용이하게 개발 및 유지보수 할 수 있도록 소스의 연결을 그래프로 가시화해주는 연구가 있었다. 그리고 함수의 호출정도를 이용하여 결함에 취약한 함수를 도출한 연구가 있다. 그렇지만, 소스 코드의 핵심함수를 도출하는 연구나 함수의 사회연결망을 분석하여 핵심 함수를 도출하여 결함을 제거하고 성능을 개선하면, 전체 실행 결함을 줄

이고 성능을 향상시킬 수 있다는 취지의 연구는 미흡한 것으로 파악되었다. 이에 본 연구에서는 사회연결망 기법을 이용하여 큐브리드 공개 데이터베이스 소프트웨어의 핵심 함수를 도출하는 연구를 수행하고자 한다.

2.2 관련 기법의 고찰

2.2.1 사회연결망분석 기법

사회연결망(Social Network)은 노드(Node)와 노드들이 연결된 링크(Link)로 구성된다. 사회연결망 분석은 연결망의 노드와 링크 구조를 그래프 이론으로 분석하여 연결망의 특성을 파악하는 기법이다. 사회연결망 내에서 중요한 위치를 점유하고 있는 노드들을 도출할 수 있다면, 그 노드들을 집중적으로 관리하여 전체 연결망의 효과를 증대시킬 수 있을 것이다(Wikipedia, 2018).

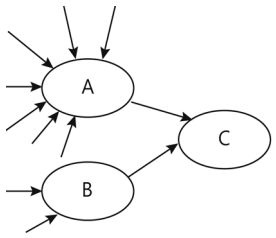
소프트웨어의 소스 함수들은 서로 호출하며 연결망을 형성하고 있고, 중요한 위치를 점유하고 있는 함수가 있을 것이다. 사회연결망 분석 지표를 이용하여 중요한 위치를 점유한 핵심함수를 도출할 수 있다면, 결함을 집중적으로 제거하고, 수행속도를 개선시키면, 전체 실행결함은 감소될 것이며, 전체 성능은 개선될 것이다. 이에 사회연결망분석 기법을 이용하여 핵심함수를 도출하고자 한다. 사회연결망 기법은 <Table 1>과 같이, 사회연결망의 구조 분석과 특성을 분석할 수 있다.

구조 분석의 연결성(Connection) 지표를 살펴보면, 연결정도, 밀도, 포괄성, 최단경로거리 등을 분석할 수 있다. 중심성 지표는 지역 중심성을 분석할 수 있는 연결정도(Degree)중심성과 전체 중심성을 분석할 수 있는 인접(Closeness) 중심성, 그리고 노드와 노드 간의 연결을 분석할 수 있는 사이(Between) 중심성 지표가 있다. 관계성(Cohesion)지표는 연결망의 분포와 등위적 관계성을 분석할 수 있다.

본 연구는 함수들 간의 관계를 이용하여 빈번하게 실행되는 핵심함수를 도출하는 것이므로 세 가지 중심성 지표 중에 핵심 노드를 도출할 수 있는

〈Table 1〉 The Analysis Scope of Social Network

Category	Meaning	Indicators
Structure anaysis	Connection	Degree mean
		Density
		Inclusiveness
		Geodesic distance
		Redundancy
	Centrality	Degree centrality
		Closeness centrality
		Between centrality
Characteristic analysis	Cohesion	Component
		Clique
		Structure equivalence



〈Figure 1〉 Example of Function's Connection

노드 기반 지표를 적용하고자 한다. 연결정도 중심성 지표는 지역 중심성을 측정하는 가장 단순한 방법으로서, 노드가 얼마나 많이 연결되어 있는 지를 측정하는 지표이며, 연결된 정도가 많으면 핵심노드가 될 가능성이 있다. 노드 v_i에 대한 연결정도 Cd는

$$Cd(v_i) = d_i \tag{1}$$

이며, d_i는 노드 v_i에 연결된 횟수를 의미한다. 연결정도는 내향연결정도와 외향연결정도 그리고 두 개를 합한 연결정도가 있다(Zafarani et al., 2014)

$$Cd(v_i) = d_i(in) \quad \text{(명성)} \tag{2}$$

$$Cd(v_i) = d_i(out) \quad \text{(사교)} \tag{3}$$

$$Cd(v_i) = d_i(in) + d_i(out) \tag{4}$$

연결 중심성은 <Figure 1>과 같이 함수들이 연결되었을 때, 함수 A는 7개의 내향연결정도와 1개의

외향연결정도를 가지고 있고, 함수 C는 2개의 내향연결정도를 가지고 있다. 함수 A는 7번 연결되어 있으므로 다른 함수보다 빈번하게 실행될 수 있으므로 핵심 함수일 가능성이 높다. 그러므로 첫 번째 분석지표로 연결정도 중심성 지표를 선정한다.

그렇지만, 함수 A가 실행되면 함수 C도 실행될 가능성이 있으므로 함수 C가 중요하지 않은 함수라고 단정할 수는 없을 것이다. 그러므로 피링크 된 함수 C도 자주 사용되는 핵심 함수일 가능성이 있다. 이것을 위세 중심성(Eigenvector Centrality)이라고 한다. 하지만, 위세중심성 지표는 비순환 방향성 그래프와 같은 특별한 경우에 중심성이 0이 나올 수 있다는 단점이 있다. 이를 방지하기 위하여 Katz는 모든 노드 중심성에 특정 상수 값을 더하는 방식을 제안하였다. 하지만, Katz 중심성은 노드의 중요성이, 연결되어 있는 다른 노드로 전파되는 특성이 있다. 다시 말하면, 한 노드가 중요하게 도출되면, 그 노드와 연결된 노드들도 전부 같이 중요도가 높아지는 단점이 있다(Zafarani et al., 2014). 구글의 페이지랭크 알고리즘은 Katz의 단점을 보완한 것으로서, 노드의 영향력이 다른 노드로 전파될 때, 외부로 향하는 모든 간선의 수로 나누어 영향력이 지나치게 전파되는 것을 방지했다. 페이지랭크 수식과 변수에 대한 설명은 식 (5)와 같고, 본 연구에서는 위세중심성을 대표하는 페이지 랭크를 두 번째 분석지표로 선정하였다.

$$Cp(v_i) = \frac{(1-d)}{n} + d\sum_{i=1}^{n}\frac{PR(T_i)}{C(T_i)} \quad (5)$$

$Cp(V)$: PageRank value of Node V_i
T_i : Node that points to the V_i node
$PR(T_i)$: PagerRank value of Node T_i
$C(T_i)$: Number of outgoing links from node T_i
d : Damping Factor(Value 0.5 was applied in this study)

피링크에 중점을 두는 위세 중심성과는 다르게, 노드 간의 링크 관계를 분석한 HITS(Hypertext Induced Topic Selection)라는 지표도 있다. 중요한 노드로부터 많은 링크를 받는 노드(Authority)는 중요한 노드로 간주된다. 그리고 중요한 노드에 링크를 많이 연결하는 노드(Hub)도 중요하다고 간주된다. 링크 구조를 양면으로 분석하여 우수한 참조관계를 이용하여 중요한 노드를 도출하는 식은 식 (6), 식 (7)과 같으며(Kim et al., 2003), 세 번째 분석지표로 선정하였다.

$$Hub[n] = \sum_{\forall n', n-point-n'} Auth[n'] \quad (6)$$

$$Auth[n] = \sum_{\forall n', n-point-n'} Hub[n'] \quad (7)$$

지금까지의 중심성 척도는 함수가 연결된 것을 이용하였지만, 직접적인 연결만으로 중요도를 도출하는 것은 전체 연결망에서 그 노드의 영향력을 파악하기 어렵다. 그러므로 직접적인 연결과 간접적인 연결을 고려하여 중요한 노드를 도출해야 한다(Kim and Kim, 2016; Zafarani, 2014). 인접 중심성 지표는 전체 연결망에서의 경로 거리의 합을 측정하여 가장 중심에 있는 노드를 도출한다. 가장 중심에 있는 노드는 빈번하게 실행될 가능성이 있으므로, 인접 중심성 지표를 네 번째 분석지표로 선정하였다. 사이 중심성 지표는 함수가 다른 함수에 도달하기 위하여 경유하는 함수를 도출한다. 다른 함수에 의해 많이 경유되는 함수는 빈번하게 호출될 가능성이 있으므로 사이 중심성 지표를 다섯 번째 분석지표로 선정하였다. 이외에도 경로 흐름과 중요 경로를 분석할 수 있으나, 빈번하게 실행되는 함수를 도출하는 본 연구의 목표에 부합되지 않아서 제외하였다. 본 연구를 위하여 선정된 지표는 <Table 2>와 같이 세 가지 중심성의 다섯 가지 지표로 분석하고자 한다.

소규모의 프로그램을 제작한 개발자들은 자신의 코드를 잘 알기 때문에, 적합한 사회연결망 지표를 취사선택하여 분석하면 될 것이다. 하지만, 소스가 방대해지면, 소스 함수의 사회연결망 구조가 어떻게 형성되어 있는 지 아무도 알 수 없을 것이며, 어느 지표를 적용해서 분석해야 하는 지 알 수 없을 것이다. 이럴 경우에는 지표 전체를 이용하여

〈Table 2〉 Research Scope for Node-base Indicators

Centrality	Indicator	Explanation
Degree	Degree	Analyze nodes with high connectivity.
	PageRank	This is a representative indicator for analyzing the eigenvector.
	HITS	Analyze nodes with excellent reference relationships.
Closeness	Closeness	Analyze the most central node in a total network.
Between	Node Between-ness	Analyze high degree of broker nodes.

통합적으로 중요한 핵심함수를 산정하는 것이 바람직한 방법일 것이다. 이에, 자료포락분석기법을 이용하여 도출된 지표 전체 값을 통합하여 핵심함수를 도출하는 방법에 대하여 살펴보도록 한다.

2.2.2 자료포락분석

1978년에 처음 소개된 자료포락분석기법(DEA : Data Envelopment Analysis)은 비교할 수 있는 DMU(Decision Making Units, 의사결정단위)에 대한 상대적인 효율성을 평가하는 기법이다. 자료포락분석 기법은 입력물과 산출물에 있어 비교가 어려운 경우에 상대적인 성과를 측정하기 위하여 사용할 수 있다(Choi and Kim, 2007). DEA를 통한 효율성 산정은 식 (8)과 같다.

$$\text{효율성} = \frac{\text{가중치를 적용한 산출물의 합}}{\text{가중치를 적용한 입력물의 합}} \quad (8)$$

$$Max \qquad E_k = \frac{\sum_{r=1}^{s} y_{kr} u_{kr}}{\sum_{i=1}^{m} x_{ki} v_{ki}} \quad (9)$$

$$Subject\ to \quad E_k = \frac{\sum_{r=1}^{s} y_{jr} u_{kr}}{\sum_{i=1}^{m} x_{ji} v_{ki}} \le 1 \quad \text{for each unit j} \quad (10)$$

$v_{ki} \ge \varepsilon,\ i = 1, 2, \cdots, m$

$\mu_k \ge \varepsilon,\ i = 1, 2, \cdots, s$

E_k = DMUk 효율성

u_{kr} = DMUk 산출물 r에 대한 가중치(r = 1, 2, ⋯, s)

y_{kr} = DMUk 산출물 r의 값(r = 1, 2, ⋯, s)

v_{ki} = DMUk 투입요소 i 가중치(i = 1, 2, ⋯, m)

x_{kj} = DMUk의 투입요소 I 값(i = 1, 2, ⋯, m)

ε : 아주 작은 값, $0 \le E_k \ge 1$

$v_{ki} > 0$, $u_{kr} > 0$ & Ek = 1이면 효율적이고

$Ek < 1$이면 비효율적임.

의사결정단위가 n개이고 i개의 입력물과 j개의 산출물이 있다면, 의사결정단위의 상대적 효율성은 CCR 모형으로 구할 수 있고 식 (9), 식 (10)과 같다. 자료포락분석 분석 유형은 규모수익(RTS, Return to Scale)에 따라 CCR Efficiency 모형과 BCC Efficiency 모형으로 분류한다. CCR 모형은 1978년에 Charnes, Cooper and Rhodes에 의해 개발되었고(Charnes, 1978), 입력과 산출의 관계가 규모에 따라 변하는 가변적인 수익을 산정할 수 있는 모형이고, 1984년에 Banker, Charnes and Cooper에 의해 개발된(Banker, 1984) BCC 모형은 입력과 산출의 관계가 규모와 관계없이 일정한 비율로 수익을 산정할 수 있는 모형이다(Seo and Ahn, 2016). 자료포락분석 기법의 계산 방법에는 Standard Efficiency 계산 방법과 Super Efficiency 계산 방법이 존재한다. Standard Efficiency 계산방법은 입력물에 대한 산출물의 효율성이 100%를 초과하지 못한다. 반면에, Super Efficiency 계산방법은 100%를 초과하여 최대한으로 효율을 계산할 수 있다(Adler et al., 2002; Cooper et al., 2004). 본 연구에서는 CCR 모형과 Super Efficiency 계산법을 적용하여 핵심 함수를 그룹 별로 도출하는 방법을 적용하였다.

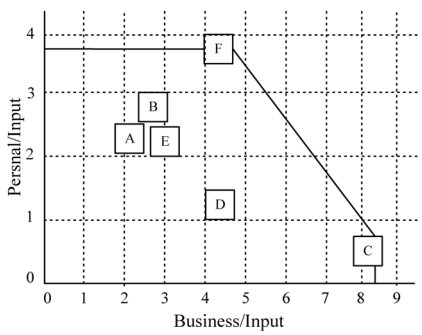
그룹 별로 가장 좋은 DMU를 도출하는 방법은 사례를 통하여 설명하고자 한다. <Table 3>은 은행 영업사원이 Input만큼의 경비를 사용하였고 기업예금(Business Trans)과 개인예금(Personal Trans)을 수주한 실적이다. 이 자료를 이용하여 영업을 가장 잘하는 영업사원 그룹과 두 번째로 잘하는 영업사원 그룹을 도출하는 방법을 살펴본다. DEA Frontier

〈Table 3〉 The Sale of Sales Man

Sales man	Input	Output	
		Business Trans	Personal Trans
A	10	23	25
B	10	30	30
C	20	165	10
D	22	65	50
E	30	125	35
F	12	55	45

〈Table 4〉 The Score of First Sales Man Group

Sales man	Score	Efficient
A	66.7%	-
B	80.0%	-
C	180.0%	○
D	62.6%	-
E	62.2%	-
F	750.0%	○



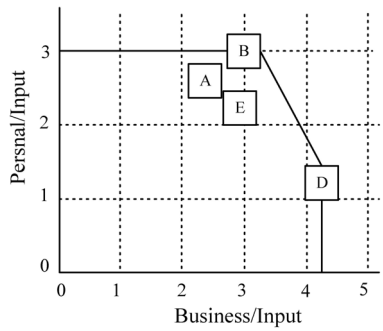
〈Figure 2〉 The Efficient Frontier Diagram of First Sales Man Group

도구의 CCR 모형과 Super-Efficiency 계산법으로 <Table 3>을 계산하면 <Table 4>와 <Figure 2>와 같은 Efficient Frontier 도표가 도출된다. <Table 4>를 보면 100%가 넘는 영업사원은 C와 F이며, <Figure 2>와 같이, 다른 영업사원을 포락하고 있으며, 영업을 제일 잘하는 그룹으로 분석되었다.

두 번째로 영업을 잘하는 영업사원을 도출하기 위하여 C, F를 제외하고 다시 계산하였고, <Table 5>와 <Figure 3>과 같은 결과가 도출되었다.

〈Table 5〉 The Score of Second Sales Man Group

Sales man	Score	Efficient
A	83.3%	-
B	132.0%	○
D	138.9%	○
E	89.6%	-



〈Figure 3〉 The Efficient Frontier Diagram of Second Sales Man Group

<Table 5>에서 100%가 넘는 영업사원은 B와 D이며, <Figure 3>과 같이 다른 영업사원을 포락하고 있으며, 두 번째로 영업을 잘하는 그룹으로 분석되었다

자료포락분석 계산을 두 번 수행한 결과를 정리하면 <Table 6>과 같다. 자료포락분석 기법으로 처음 계산했을 때, 영업사원 D는 62.6%로 저조한 결과였지만, 첫 번째 그룹을 제외하고 계산하니, 두 번째로 영업을 잘하는 그룹에 포함되었다. D는 기업 실적이 좋아서 선정된 것으로 파악되었다.

〈Table 6〉 The DEA Summary of Sales Man

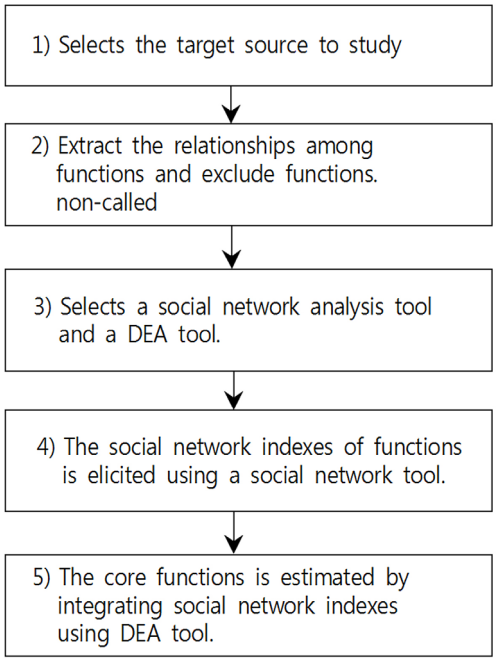
Sales man	Input	Output		Group 1	Group 2
		Business Trans	Personal Trans	Sore	Score
F	12	125	35	750.0%	-
C	20	55	45	180.0%	-
D	22	65	50	62.6%	138.9%
B	10	30	30	80.0%	132.0%

3. 연구 설계 방법

3.1 연구과제에 대한 목표설정

3.1.1 연구를 위한 사회연결망 기법 적용방법

연구 절차는 <Figure 4>와 같이, 첫 번째, 연구할 소프트웨어 소스를 선정한다. 두 번째, 선정된 소스에서 함수들 간의 연결 경로를 도출한다. 세 번째, 사회연결망 분석과 자료포락분석을 위한 도구를 선정한다. 네 번째, 사회연결망분석도구를 이용하여 각 함수들에 대한 사회연결망 지표 값을 도출한다. 마지막으로 자료포락분석도구를 이용하여 도출된 사회연결망지표를 통합하여 핵심 함수를 도출한다.



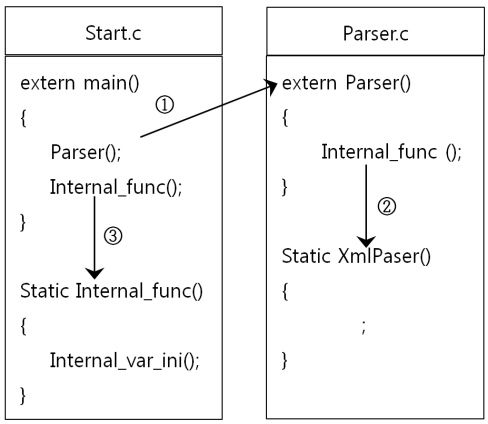
〈Figure 4〉 Research Procedures

3.3 연구 소스 및 도구 선정

연구의 첫 번째 단계는, 연구를 위한 소스 선정 단계로서, 본 연구에서는 공개 소프트웨어 큐브리드 데이터베이스 Ver 10.0.0.136 소스를 선정하였다. 대상 소스는 공개용이고 공공기관, 회사 등에서 많이 사용하고 있고, 지속적으로 성능향상을 하고 있으므로 연구에 적합하다고 판단하였다.

3.4 소스 코드의 함수들 간의 관계 도출

연구의 두 번째 단계로서, 프로그램 소스에서 함수들 간의 연결을 도출한 후 호출되지 않는 함수를 제외시키는 단계이다. 큐브리드 데이터베이스 소스에 대하여 함수 연결 관계를 분석한 결과, 호출되지 않는 함수들이 존재하였다. 이런 함수들은 결함과 성능에는 관련이 없기 때문에 연구 대상에서 제외시켰다. 큐브리드 데이터베이스 C언어 소스 파일은 약 630여 개가 존재하였고, 파일 사이에서 함수들이 호출될 수 있도록 extern으로 정의한 글로벌 함수와 파일 내부에서만 사용할 수 있도록 정의한 static 함수로 코딩되어 있었다.



〈Figure 5〉 The Function Call among C Source Files

소스에서 함수를 도출하는 방법을 <Figure 5>로 설명하면, 소스 파일은 2개(Start.c, Parser.c)로 구성되어 있고, Start.c의 main 함수 내에서 <Figure 5>의 ①처럼 Parser.c의 Paser() 함수를 호출하고, Paser.c는 <Figure 5>의 ②처럼 자기 자신 파일 내의 Internal_func 함수를 호출하도록 코딩되어 있다고 가정한다. 또한, Start.c는 <Figure 5>의 ③처럼 자기가 속한 파일 내의 Internal_func 함수를

호출한다. static으로 선언된 Internal_func() 함수는 Start.c와 Parser.c에 전부 존재하지만, 파일 내부에서만 사용되므로 컴파일이나 실행하는 데 전혀 문제가 없다. 하지만, 이것을 구분하지 않고 그냥 도출하면 Internal_func 함수가 한 개만 도출되므로, 이 함수가 많이 사용되는 결과가 도출될 것이다. 이것은 연구진이 원하는 결과가 아니므로 Internal_func 함수 이름 앞에 파일명을 붙여서 유일하게 한 후에 도출하였다. Start.c에서는 main, Start.Internal_func가 도출되고, Parser.c에서는 Parser, Parser.Internal_func 함수가 도출되도록 하였다.

4. 연구 수행 및 결과

4.1 사회연결망 지표 적용 범위

본 연구를 위하여 <Table 3>과 같이 선정된 연결정도지표, 페이지 랭크지표, HITS 지표, 인접중심성지표 및 사이 중심성지표를 이용하여 순서적으로 분석한다.

4.2 연구 수행 및 결과

4.2.1 사회연결망분석을 통한 핵심함수 도출

첫 번째, 연결정도지표로서, 연결된 정도가 높으면 핵심함수가 될 가능성이 높다. NetMiner의 연결정도지표로 분석한 화면은 <Figure 6>과 같고, 내향연결정도 값이 큰 순서대로 정렬한 결과, <Table 7>과 같은 결과가 도출되었다.

		1	2
		In-Degree	Out-Degree
1	CreateMiniDump	2	2
2	enwar_bindir_file	7	1
3	ux_database_shutdown	5	1
4	MemcatImproved	1	1
5	ReallocImproved	1	0
6	SHA1Input	1	1
7	SHA1ProcessMessageBlock	2	0
8	SHA1PadMessage	1	1
9	SHA1Result	1	1

〈Figure 6〉 In-Degree and Out-Degree Result Screen Using NetMiner Tool

<Table 7>에서 보듯이 er_set 함수의 내향연결정도는 1,709번 연결되었고 맨 밑에 있는 error_manager. er_set_internal 함수는 3번 연결되었다. 연결정도 관점으로는 er_set은 많이 연결되었으므로 중요한 함수이고, err_set_internal 함수는 중요한 함수가 아니라고 잘못된 정보를 제공할 수 있다. 사회연결망분석에서는 연결정도중심성 지표를 제공하고 있으며, NetMiner로 분석한 결과, <Table 8>과 같은 결과가 도출되었다. <Table 7>과 <Table 8>의 값은 다르지만 함수 순서는 동일하다. 연결 중심성은 노드들이 단순히 연결된 횟수만을 대상으로 한다는 단점이 있다.

〈Table 7〉 The Prioritized In-Degree and Out-Degree for CUBRID Database Source Functions

Functions	In-Degree	Out-Degree
er_set	1,709	1
er_errid	844	1
pr_clear_value	304	5
or_unpack_int	237	0
parser_free_tree	200	1
or_pack_int	191	0
pgbuf_check_page_ptype	185	1
pgbuf_set_dirty	161	4
pt_has_error	149	1
:	:	:
error_manager.er_set_internal	3	6

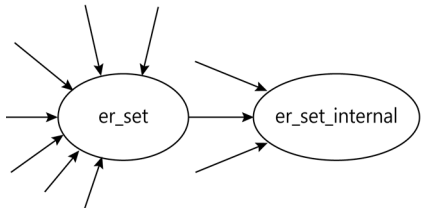
〈Table 8〉 The Prioritized In-Degree and Out-Degree Centrality for CUBRID Database Source Functions

Functions	In-Degree	Out-Degree
er_set	0.158387	0.000093
er_errid	0.078221	0.000093
pr_clear_value	0.028174	0.000463
or_unpack_int	0.021965	0
parser_free_tree	0.018536	0.000093
or_pack_int	0.017702	0
pgbuf_check_page_ptype	0.017146	0.000093
pgbuf_set_dirty	0.014921	0.000371
pt_has_error	0.013809	0.000093

두 번째, 위세중심성의 대표적인 지표로는 구글의 페이지 랭크 지표로서, <Figure 1>에서 예시했듯이, 함수 C는 함수 A가 실행되면, 함수 C도 실행될 가능성이 있으므로 중요하지 않은 함수라고 단정할 수는 없을 것이다. 하지만 어느 정도 확률로 실행될지는 알 수 없다. 페이지 랭크지표에서는 이것을 댐핑계수라고 하며, 기본적으로 0.85의 값을 사용하고 있다. 본 연구에서는 특정 함수가 실행되면 그 함수에 연결된 다른 함수가 실행될 확률을 50%라고 가정하여, 댐핑계수를 0.5로 적용하였다. NetMiner의 페이지랭크 지표로 분석하여 값이 큰 순으로 정렬한 결과, <Table 9>와 같은 결과가 도출되었다. er_set_internal 함수에 대한 연결정도 중심성 지표에서는 <Table 7>에서는 중요하지 않은 함수라고 제시되었지만, 페이지 랭크 지표에서는 두 번째로 중요한 함수라고 제시하고 있었다. 이 결과에 대하여 error_manager.c 소스를 분석한 결과, er_set_internal 함수는 <Figure 7>과 같이 er_et 함수에 연결되어 있고, 동일한 파일 내에서만 3번 호출되고 있었다. 페이지 랭크에서 er_set_internal 함수가 중요한 함수라고 도출된 이유는, 중요한 er_set 함수가 실행되면 그 함수에 연결된 er_set_internal 함수도 실행될 가능성이 있기 때문에 중요하다는 의미인 것이다. 이와 같이 구글 페이지 랭크는 피링크에 중점을 두어 분석하는 것이 특징이다.

〈Table 9〉 The Prioritized PageRank Centrality for CUBRID Database Source Functions

Functions	PageRank
er_set	0.020918
error_manager.er_set_internal	0.011542
thread_get_thread_entry_info	0.00735
er_errid	0.005943
error_manager.er_get_er_entry	0.004466
parse_tree_cl.pt_walk_private	0.003147
parser_walk_tree	0.002859
pr_clear_value	0.002676
pgbuf_check_page_ptype	0.002263



〈Figure 7〉 The er__set and er__set__internal Connection Graph

세 번째, HITS라는 지표는 사회연결망을 양면으로 분석하여 우수한 참조관계를 찾아내는 지표로서, NetMiner의 HITS 지표로 분석하여, Authority 값이 큰 순으로 정렬한 결과, <Table 10>과 같은 함수들이 도출되었다. HITS 지표에서 노드의 Authority 점수가 높으면 Hub 점수가 높다는 것이고, Hub 점수가 높다는 것은 노드의 링크가 많다는 것이므로 실행될 가능성이 높다는 것을 의미하므로 핵심함수 일 가능성이 높다는 것으로 이해할 수 있다.

〈Table 10〉 The HITS for CUBRID Source Functions

Functions	Authority	Hub
do_get_serial_obj_id	0.006886	0.032888
pgbuf_get_vpid_ptr	0.006845	0.000062
es_get_type	0.006784	0.000000
db_datetime_decode	0.006781	0.000301
obj_set	0.006768	0.000004
sm_downcase_name	0.006742	0.000154
tp_domain_resolve_value	0.006675	0.001205
mht_put	0.006665	0.000000

지금까지 살펴본 중심성 척도는 연결된 것을 기준으로 중요 함수를 도출하였다. 하지만, 연결된 것만으로 중요도를 도출하는 것은 전체 연결망에서 그 노드의 영향력을 파악하기 어렵다. 그러므로 간접적인 연결도 고려하여 중요한 노드를 도출해야 한다.

네 번째, 인접 중심성지표는 두 노드를 연결하는 최단거리인 경로거리를 측정하여 합산한 지표이다. 인접 중심성이 높을수록 최단경로거리의 합은 작아지고 전체 연결망의 중심에 가까워지는 것을 의미

한다. NetMiner의 인접중심성 지표로 분석한 결과 <Table 11>과 같이 도출되었다. 앞서 분석한 연결정도지표는 지역 연결망 측면에서의 중요한 함수를 도출하였다면, <Table 11>의 함수들은 전체 연결망 측면에서의 중요한 함수라고 이해할 수 있다.

〈Table 11〉 The Prioritized Closeness Centrality for CUBRID Database source functions

Functions	In-Closeness	Out-Closeness
er_set	0.226960	0.001224
error_manager.er_set_internal	0.169888	0.001568
thread_get_thread_entry_info	0.162651	0.000000
error_manager.er_get_er_entry	0.155422	0.000093
er_errid	0.152087	0.000124
error_manager.er_start	0.136439	0.001239
error_manager.er_stop_on_error	0.133413	0.000212
error_manager.er_call_stack_dump_on_error	0.133216	0.000612

다섯 번째, 사이 중심성은 연결망 내에서 노드가 다른 점들 사이에 위치는 정도를 측정한다. 특정 노드가 노드와 노드사이의 최단거리를 연결하는 최단경로에 위치하면 할수록, 사이 중심성은 높아지며, 다른 노드들 사이의 중계자(브로커)역할이 증가한다. Netminer로 분석한 결과 <Table 12>와 같이 도출되었고, 도출된 함수들은 전체 연결망이 원활하게 흐를 수 있도록 하는 함수들로 이해할 수 있다.

〈Table 12〉 The prioritized Betweenness Centrality for CUBRID Database source functions

Functions	Betweeness Centrality
trigger_manager.run_user_triggers	0.050830
tr_check_abort_triggers	0.050066
trigger_manager.execute_activity	0.049814
pt_exec_trigger_stmt	0.047791
do_scope	0.047395
tran_unilaterally_abort	0.033008
ws_abort_transaction	0.031575
work_space.ws_make_mop	0.031507
ws_mop	0.030435

이상과 같이 여러 가지의 사회연결망 지표를 이용하여 핵심 함수들을 도출하였다. 데이터베이스는 실제로 사용되는 상황에 따라 중요 함수가 다르게 나올 수 있다. 예를 들면, 데이터의 용량이 적을 경우에는 데이터의 무결성을 중요시하여 정규화 방식으로 모델링하지만, 데이터가 대용량이고 빈번하게 서비스할 경우에는 속도를 중요시하여 비정규화 방식으로 모델링한다. 또한, 데이터와 데이터를 연결하여 처리하는 참조 무결성을 적용한 방식, 속도를 중요시하여 참조 무결성을 적용하지 않고 데이터를 분리하여 처리하는 경우, 트리거가 적용된 경우와 트리거가 적용되지 않은 경우, 인덱스가 적은 경우와 인덱스가 많이 있는 경우 등, 동일한 큐브리드 데이터베이스라고 하더라도 사용하는 상황에 따라 중요한 함수가 다를 수 있다. 그러므로 모든 상황에 대한 핵심함수를 도출하려면, 지금까지 도출된 사회연결망 지표 전체를 통합하여 핵심함수를 도출하는 것이 가장 근접한 방법일 것이다. 이에, 도출된 지표 전체를 통합하는 자료포락분석기법을 적용하여 핵심함수를 도출하고자 한다.

4.2.1 자료포락분석을 통한 핵심함수 도출

자료포락분석 기법을 적용하기 전에 사회연결망 기법으로 도출한 지표 값에 대하여 네 가지 유형의 사전처리를 수행하였다. 첫 번째, 도출된 결과 값들은 값이 클수록 핵심함수가 될 가능성이 높으므로 식 (8)의 분자 항목에 해당된다. 반면에, 식 (8)의 '가중치를 적용한 입력물의 합'에 해당하는 분모 항목이 없기 때문에, 신규로 입력항목(Input)을 생성하여 모든 함수에 대한 값을 1로 부여한 후 DEA 도구의 입력물로 설정하였다. 두 번째, 도출된 지표 값이 소수점 6째 자리로 도출되어, DEA 계산이 수행되지 않았다. 그래서 전부 소수점 이상으로 값을 변환하였다. 세 번째, 도출된 지표 값이 0인 경우에는 DEA 도구가 계산할 수 없으므로 0을 아주 작은 값(very small value)으로 치환하였고, <Table 13>과 같은 값으로 변환되었다.

〈Table 13〉 The Social Network Index Values of Functions to be input in DEA Frontier Tools

Functions	Input	In-Degree Centrality	Out-Degree Centrality	PageRank Centrality	Hub	Authority	In-Closeness	Out-Closeness	Node Betweenness Centrality
boot_register_client	1	93	927	14	23,126.00	719.00	12,879	30,127	4
boot_reset_db_parm	1	93	93	15	very small valu	485.00	165	18,727	very small valu
boot_restart_client	1	185	5,468	24	31,652.00	469.00	13,441	50,519	5,948
boot_restart_from_backup	1	very small value	93	14	20,596.00	very small valu	very small valu	1,026	very small valu
boot_server_die_or_changed	1	93	278	38	218.00	674.00	27,346	35,980	8,205
boot_server_status	1	185	93	56	20,596.00	968.00	25,152	1,026	very small valu
boot_shutdown_client	1	278	834	31	134.00	1459.00	12,917	43,344	538
boot_shutdown_server	1	small value	93	14	20,596.00	very small valu	very small valu	1,026	very small valu

〈Table 14〉 The List of Core Functions Calculated by DEA Frontier Tool

Functions	Line	Group 1	Group 2	Group 3	Group 4	In-Degree	Out-Degree	Page-Rank	Hub	Auth-ority	In-Closeness	Out-Closeness	Node Betweenness
		Score	Score	Score	Score								
pr_clear_value	87	785.4%	-	-	-	28,174	463	68,983	894	304	2,174	91,337	721
fetch.fetch_peek_arith	3,249	268.5%	-	-	-	93	16,219	23,195	39,762	24,014	28	533	31,051
trigger_manager.run_user_triggers	33	133.9%	-	-	-	278	371	30,313	44,268	50,830	37	654	28,720
do_scope	36	107.6%	-	-	-	278	278	22,274	62,306	47,395	16	1,633	74
do_execute_statement	329	105.2%	-	-	-	185	5,931	20,982	69,535	24,136	19	773	30,777
pt_exec_trigger_stmt	115	101.1%	-	-	-	185	1,112	23,586	56,348	47,791	21	977	10,328
db_vdb.db_execute_and_keep_statement_local	339	100.5%	-	-	-	278	3,336	16,249	64,687	6,155	38	669	33,929
object_domain.tp_value_cast_internal	2,800	83.8%	177.5%	-	-	185	10,195	29,480	37,114	17,424	267	476	20,913
error_manager.er_get_er_entry	11	91.9%	147.6%	-	-	1,112	93	155,422	93	11	8,515	73	265
do_statement	325	99.8%	133.6%	-	-	278	5,468	20,982	68,946	20,211	20	996	30,744
er_clear	19	84.3%	132.4%	-	-	11,399	278	84,779	1,046	168	665	53,309	267
db_value_domain_init	183	93.4%	111.4%	-	-	10,380	93	46,389	1,026	57	1,060	29,928	20,596
ws_mop	70	91.3%	111.3%	-	-	4,449	463	43,171	33,638	30,435	268	13,763	20,810
spage_get_record	15	98.0%	110.8%	-	-	12,326	463	35,176	1,335	171	624	24,019	21,342
pt_to_regu_variable	1,599	99.2%	110.0%	-	-	1,854	4,727	18,245	49,672	4,903	82	3,342	35,604
locator_cl.locator_lock	140	97.4%	107.6%	-	-	741	1,483	34,798	40,940	29,108	124	2,419	30,899

〈Table 14〉 The List of Core Functions Calculated by DEA Frontier Tool(Continue)

Functions	Line	Group 1	Group 2	Group 3	Group 4	In-Degree	Out-Degree	Page-Rank	Hub	Auth-ority	In-Closeness	Out-Closeness	Node Betweenness
		Score	Score	Score	Score								
au_drop_user	221	98.0%	102.5%	-	-	185	2,966	17,416	55,196	1,759	31	196	36,889
fetch_peek_dbval	338	87.2%	102.4%	-	-	3,336	927	25,096	37,624	29,762	178	11,807	23,473
ws_abort_transaction	14	92.2%	102.0%	-	-	185	278	33,798	38,433	31,575	60	469	28,700
do_alter_serial	467	96.7%	101.5%	-	-	185	2,595	19,641	44,153	1,419	14	1398	36,623
do_insert_template	512	95.8%	100.7%	-	-	93	3,892	17,899	52,921	2,198	15	18	36,470
error_manager.er_start	115	82.5%	94.5%	121.4%	-	278	834	136,439	1,239	1,763	2,778	18	456
heap_file.heap_mvcc_lock_and_get_object_version	530	90.6%	96.6%	121.1%	-	649	1,483	29,124	33,423	28,059	117	691	29,794
fileio_get_volume_label	50	87.8%	95.5%	117.8%	-	4,819	93	46,007	1,026	1	758	25,063	20,596
boot_restart_client	470	87.8%	97.2%	116.0%	-	185	5,468	13,441	50,519	5,948	24	469	31,652
error_manager.er_stop_on_error	11	92.2%	99.9%	115.8%	-	185	185	133,413	212	82	2,585	14	8,074
cas_execute.dbval_to_net_buf	574	62.4%	75.3%	110.9%	-	649	6,487	701	33,601	270	29	371	5,008
locator_flush_class	37	87.2%	99.1%	108.8%	-	2,410	741	24,219	40,362	8,332	66	10,868	29,077
tran_unilaterally_abort	17	84.4%	95.2%	108.4%	-	463	278	31,258	41,447	33,008	47	2,007	20,633
boot_initialize_client	312	91.5%	97.2%	107.3%	-	93	4,634	93	59,165	49	18	2	32,520
db_execute_statement_local	16	85.9%	87.1%	106.3%	-	1,019	371	17,143	57,589	5,797	55	3,018	21,195
query_executor.qexec_execute_insert	682	91.8%	97.8%	104.1%	-	185	4,449	13,789	51,139	647	16	704	34,667
sm_delete_class_mop	237	88.2%	89.3%	103.0%	-	556	3,151	652	60,128	406	42	2,239	11,879
tran_abort_only_client	24	82.9%	91.8%	103.0%	-	1,112	463	33,552	38,479	28,365	49	6,462	20,684
set_get_element	16	74.9%	82.3%	102.8%	-	6,024	185	37,422	31,995	3,238	325	20,445	31
mht_create	51	92.0%	97.7%	102.7%	-	1,854	371	83,716	1,099	201	141	5,769	20,671
do_get_serial_obj_id	46	92.8%	98.2%	102.6%	-	927	834	23,267	33,615	579	27	6,886	32,888
work_space.ws_make_mop	46	82.3%	95.5%	102.3%	-	463	185	37,966	35,828	31,507	81	2,157	20,606
sm_add_index	351	89.5%	92.4%	101.9%	-	649	2,595	17,477	52,817	2,616	35	2,247	31,181
do_prepare_update	225	90.4%	93.9%	101.5%	-	93	2,039	19,734	50,667	3,300	17	185	33,552
classobj_make_class_constraints	330	92.2%	96.7%	101.3%	-	1,205	1,576	19,808	30,379	470	47	5,228	33,318
execute_schema.do_create_partition	449	91.2%	96.2%	101.2%	-	278	2,317	297	56,039	87	20	966	33,546
execute_statement.insert_subquery_results	325	92.0%	95.7%	101.0%	-	93	3,151	16,913	48,757	4	14	829	35,050
jsp_cl.jsp_add_stored_procedure	164	92.8%	96.6%	101.0%	-	93	1,946	18,460	44,470	128	16	667	35,426
pt_to_update_xasl	575	90.6%	94.1%	100.9%	-	185	3,429	18,555	50,732	1,480	28	763	33,570
locator_sr.locator_check_primary_key_update	258	92.9%	96.7%	100.8%	-	185	2,224	17,660	39,280	1,047	16	575	35,557
do_update_maxvalue_of_auto_increment_serial	147	93.6%	97.2%	100.7%	-	93	2,039	124	40,470	2	14	755	35,730
network_cl.set_server_error	40	89.0%	96.0%	100.6%	-	1,576	371	30,021	33,711	8,203	95	8,567	29,453
execute_schema.create_or_drop_index_helper	201	87.3%	89.1%	100.6%	-	278	1,483	18,466	53,759	1,285	27	583	30,022
authenticate.au_get_new_auth	127	90.8%	92.7%	100.6%	-	185	1,668	15,658	53,579	297	21	375	33,058

〈Table 14〉 The List of Core Functions Calculated by DEA Frontier Tool(Continue)

Functions	Line	Group 1 Score	Group 2 Score	Group 3 Score	Group 4 Score	In-Degree	Out-Degree	Page-Rank	Hub	Auth-ority	In-Closeness	Out-Closeness	Node Betweenness
pt_value_to_db	124	74.8%	81.6%	100.4%	-	4,541	1,019	21,669	41,216	5,143	122	14,608	1,105
authenticate.fetch_class	114	67.4%	70.3%	100.4%	-	278	649	37,930	35,776	11,552	1,006	949	8,971
locator_sr.locator_check_primary_key_delete	303	93.0%	96.7%	100.4%	-	93	2,410	16,700	39,449	144	16	275	35,648
error_manager.er_event_final	32	93.9%	99.0%	100.1%	-	93	185	94,837	1,136	5	54	6	20,977
db_vdb.do_recompile_and_execute_prepared_statement	46	84.1%	84.7%	100.0%	-	93	556	15,430	58,386	155	15	771	8,224
do_create_serial	338	92.2%	95.9%	100.0%	-	185	1,946	19,642	42,260	567	14	1,398	34,918
cas_execute.netval_to_dbval	508	60.5%	71.4%	95.5%	133.5%	93	6,024	212	34,984	64	21	156	6,127
pgbuf_set_dirty	25	54.3%	72.0%	94.6%	127.1%	14,921	371	28,645	1,131	62	512	32,331	866
query_executor.qexec_execute_mainblock_internal	741	84.0%	90.2%	99.2%	121.8%	93	4,263	14,438	48,971	9,344	16	490	30,715
sm_partitioned_class_type	114	75.6%	88.4%	92.9%	109.7%	2,966	463	23,008	32,312	254	82	15,260	22,643
er_msg	17	66.2%	76.6%	94.1%	109.6%	4,912	185	88,280	899	96	735	20,840	1,213
set_size	17	72.4%	79.8%	97.6%	107.4%	6,024	185	37,482	31,470	3,248	299	17,617	32
csql.csql_do_session_cmd	549	78.4%	79.1%	92.0%	105.7%	93	4,171	124	54,539	19	15	1	9,914
authenticate.update_cache	118	86.4%	93.2%	99.6%	105.6%	278	1,112	30,283	37,048	14,379	53	1,490	30,447
sm_add_constraint	78	82.4%	83.8%	97.9%	105.1%	741	741	17,468	55,648	720	35	2,783	8,598
locator_is_class	47	73.9%	81.8%	98.5%	104.1%	3,707	371	40,904	35,410	16,016	541	7,237	8,445
pt_report_to_ersys	32	87.7%	80.6%	83.3%	103.9%	2,780	185	27,701	1,039	6	74	10,798	28,654
do_prepare_merge	315	89.5%	94.6%	99.3%	103.9%	93	2,502	19,734	45,202	1,520	17	185	34,312
execute_schema.do_change_att_schema_only	263	88.9%	92.4%	97.2%	103.8%	93	1,946	93	51,651	9	14	730	33,243
jsp_cl.jsp_do_call_stored_procedure	100	86.0%	94.5%	99.2%	103.7%	185	741	19,308	38,412	11,914	22	1,199	32,587
au_find_user	85	78.4%	91.2%	97.5%	103.4%	741	1,112	18,470	46,768	2,119	76	4,151	30,728
update_logical_result	79	64.4%	71.2%	82.4%	103.3%	278	185	26,602	35,349	24,805	27	1,400	554
jsp_call_stored_procedure	85	86.7%	91.9%	99.8%	103.2%	93	1,019	19,957	47,246	9,445	17	652	32,153
do_check_rows_for_null	93	87.1%	89.2%	97.7%	102.9%	185	1,297	15,719	52,793	816	20	949	30,674
lock_manager.lock_internal_perform_lock_object	596	64.8%	91.7%	97.8%	102.8%	834	2,595	26,935	25,697	4,360	64	3,096	30,711
execute_schema.do_redistribute_partitions_data	92	85.5%	89.0%	92.9%	102.6%	371	927	386	55,147	50	24	1,071	29,527
do_create_auto_increment_serial	194	92.1%	95.3%	99.1%	102.6%	185	1,576	299	42,170	23	16	1,443	34,850
obj_get_att	59	84.0%	90.5%	98.9%	101.8%	371	834	27,844	41,676	14,529	81	733	29,547
db_compile_statement_local	145	86.2%	88.6%	96.9%	101.8%	834	1,483	16,474	47,926	2,758	59	1,756	31,282
authenticate.get_grants	142	89.5%	94.2%	98.4%	101.7%	741	1,390	25,499	37,885	223	25	3,037	32,268
execute_statement.do_create_serial_internal	118	90.8%	94.0%	97.8%	101.7%	185	1,112	18,481	37,437	43	15	1,585	34,295
xbtree_load_index	434	88.7%	92.3%	96.9%	101.6%	93	3,058	15,060	36,431	1,868	15	708	33,832
heap_get_class_name_alloc_if_diff	35	82.9%	90.5%	95.4%	101.4%	185	741	24,419	29,695	13,218	134	265	30,533
log_end_system_op	197	74.2%	79.8%	92.5%	101.3%	5,005	927	30,427	6,512	3,439	152	14,937	20,833

자료포락분석은 DEA Frontier Ver 4.0.10 도구를 이용하였고, <Table 3>부터 <Table 6>까지와 같은 방식으로 진행하였으며, 제일 중요한 함수 그룹부터 네 번째로 중요한 함수 그룹까지 도출하였다. 도출된 함수 결과의 정확성을 향상하기 위하여 두 가지 관점으로 결과를 정리하였다. 첫 번째, 소스가 10줄 이하인 함수는 결과에서 제외시켰다. 소스가 한 줄인 함수를 확인한 결과, 'return(Function Name(Parameter));'으로 코딩되어 있었고, 함수를 연결하는 정도의 기능이므로 결과에서 제외시켰다. 또한, 10줄 이하 소스들을 확인한 결과, 결함제거나 성능향상에 관계가 없는 코드이어서 전부 제외시켰다. 두 번째, 도출된 핵심 함수가 아주 작은 값을 가지는 지표가 존재하는 경우에는 이 지표 관점으로는 전혀 중요하지 않은 함수이므로, 결함제거나 성능향상에 도움이 되지 않을 수 있으므로 결과에서 제외시켰고, <Table 14>과 같이 네 가지 그룹의 핵심함수들이 도출되었다. <Table 14>의 도출된 함수들은 모든 지표를 통합하여 도출한 함수로서 코드 인스펙션을 수행하기에 적합한 함수로 보이며, 이 함수를 검토하여 결함을 제거하고, 성능을 향상시키면, 전체 실행 결함을 줄이고 전체 성능을 개선할 수 있을 것으로 기대된다.

5. 결 론

5.1 연구의 제안

다양한 분야에서 사용하고 있는 소프트웨어는 결함이 내재되어 있고, 결함을 완벽하게 제거하는 것은 불가능한 것으로 알려져 있다. 소프트웨어에 내재한 결함을 제거하기 위하여 개발 초기에는 코드 인스펙션을 수행하고, 개발 말기에는 테스트를 수행하여 결함을 제거하고 있다. 개발초기에 코드 인스펙션을 수행하는 것이 결함을 제거하는 데 가장 효과적인 방법으로 알려져 있다. 하지만 코드 인스펙션을 어느 소스를 선정해야 하는 지에 대한 명확한 기준은 없다. 근래 IT 시스템은 점점 대형화, 복잡화되고 있는 실정이다. 대형화된 시스템의 결함을 용이하게 검출하기 위하여 정적분석 도구를 사용하고 있다. 정적분석도구는 프로그램 소스를 점검하여 결함 가능성이 있는 코드를 제시한다. 프로그램 소스는 함수들끼리 서로 호출하며 연결되어 있으므로 이 함수 연결망을 분석하면 중요 함수를 도출할 수 있을 것이다. 정적분석도구에서는 함수가 연결된 횟수정도만을 제시하고 있다. 이 지표는 많이 연결된 함수는 중요한 함수이고 적게 연결된 함수는 중요하지 않은 함수로 인식시킬 수 있다. 하지만, 사회연결망 지표에는 그것이외에 여러 지표들이 존재하며, 소스 코드 특성에 부합하는 지표로 핵심 함수를 제시한다면, 코드 인스펙션을 수행 할 때, 소스를 선정하는 명확한 기준으로 사용할 수 있을 것이다. 정적분석도구에 사회연결망 기법과 자료포락분석기법을 내장하여 핵심 함수를 자동으로 제시한다면, 그 함수들을 집중적으로 관리하여 결함을 제거하고 성능을 개선하여, 전체적인 실행 결함을 낮추고 성능을 개선시킬 수 있을 것이다.

5.2 연구의 시사점

본 연구는 노드 사회연결망 지표 중에 중심성 지표를 이용하여 큐브리드 데이터베이스의 중요한 함수를 도출하였다. 직접적인 연결정도를 측정하는 연결정도 중심성과 페이지랭크 지표, HITS 지표 등을 이용하였고, 간접적인 연결까지 고려하는 인접중앙성과 노드와 노드를 연결해주는 사이중앙성 관점으로 핵심 함수를 도출하였다. 또한, 자료포락분석기법을 이용하여 도출된 지표 전체를 통합하여 핵심함수를 도출하였다.

본 연구의 학문적인 의의는, 첫째, 기존에는 코드 인스펙션을 수행하기 위하여 어느 소스를 선정해야 하는 지에 대한 명확한 기준이 없었다. 하지만, 사회연결망 기법과 자료포락분석기법을 이용하여 핵심함수에 대한 명확한 수치를 제시하고 있으므로 코드 인스펙션을 위한 소스를 선정할 수 있는 기준으로 사용할 수 있을 것이다. 둘째, 프로그램 소스

에 대하여 사회연결망 기법과 DEA 기법을 이용하여 모든 함수를 비교하여 핵심 함수를 탐색하였다는 점이다. 그러므로 프로그램 소스에서 중요한 함수를 도출할 수 있는 학문적 기준을 제시하였고 소프트웨어 공학에서 결함제거 및 성능향상을 위한 방안으로 사용할 수 있는 이론적 근거를 제시했다는 점이다. 셋째, 소프트웨어 유형별로 어떤 종류의 사회연결망 지표가 가장 적합한지 연구할 수 있는 기반을 마련했다고 할 수 있다.

산업적인 활용 방안으로는 첫째, 정적분석도구에 사회연결망 기법과 자료포락분석기법을 내장하여 중요한 핵심 함수를 자동으로 제시할 수 있을 것이다. 둘째, 프로그램 소스에 대하여 핵심함수를 제시할 수 있으므로, 핵심함수를 집중적으로 테스트하여 결함을 제거하면 전체적으로 실행 결함을 감소시키는 효과를 얻을 수 있다. 셋째, 핵심 함수에 대한 실행 성능을 개선시키면, 전체적인 시스템 성능이 향상될 것으로 판단한다. 넷째, 방대한 규모의 소프트웨어에 대하여 적은 비용과 노력으로 정말로 중요한 핵심함수를 찾을 수 있는 있는 후보 핵심함수를 제시했다는 점이다.

5.3 연구의 한계점

본 연구는 공개 데이터베이스 큐브리드 소스에 대하여 사회연결망 기법으로 핵심 함수를 탐색한 연구이지만, 노드 기반으로만 핵심함수만을 도출하였고, 경로관점으로 어느 경로가 핵심 경로인지 대해서는 도출하기 않았다는 점이 아쉬운 점이다. 또한, 핵심 함수의 소스를 실제로 개선하지 않았으므로 어느 정도 결함 제거와 성능 개선이 되는지 알 수 없다는 것이 한계점이라고 할 수 있다.

5.4 향후 연구 방향

향후 연구로는 경로 관점으로 주요 경로를 도출하여 그 경로에 소속된 함수들을 분석하는 연구가 있으며, 큐브리드 데이터베이스나 실제 정보시스템에 대하여 도출된 핵심 함수에 대하여 집중적으로 결함으로 제거하고 성능을 개선하였을 때, 어느 정도로 실행 결함이 감소하고, 어느 정도로 성능이 개선되는 지 분석하는 연구가 있다.

References

Adler, N., L. Friedman, and Z. Sinuany-Stern, “Review of ranking methods in the data envelopment analysis context”, *European Journal of Operational Research*, Vol.140, No.2, 2002, 249-265.

Allen, F.E., “Control flow analysis”, *In ACM Sigplan Notices*, Vol.5, No.7, 1970, 1-19.

Banker, R.D., A. Charnes, and W.W. Cooper, “Some models for estimating technical and scale inefficiencies in Data Envelopment Analysis”, *Management Science*, Vol.30, No.9, 1984, 1078-1092.

Charnes, A., W.W. Cooper, and E. Rhodes, “Measuring the efficiency of decision making units”, *European Journal of Operational Research*, Vol.2, No.6, 1978, 429-444.

Choi, M.S. and W.J. Kim, “A Study on an Evaluation Method for LCD TV products using Hybrid AHP/DEA Model”, Dept. of Information and Industrial Engineering Graduate School of Industry and Engineering Seoul National University of Technology, 2007.
(최민수, 김우제, “AHP/DEA 혼합모형을 이용한 LCD TV 평가방법에 관한 연구”, 서울산업대학교 산업대학원 정보산업공학과, 2007.)

Cooper, W.W., L.M. Seiford, and J. Zhu, “Data envelopment analysis. In Handbook on data envelopment analysis”, Springer, Boston, MA, 2004, 1-39.

Ferrante, J., K.J. Ottenstein, and J.D. Warren, “The program dependence graph and It’s use in optimization”, *ACM Transactions on*

Programming Languages and Systems (TOPLAS), Vol.9, No.3, 1987, 319-349.

Hall, N.R. and S. Preiser, "Combined network complexity measures", *IBM Journal of Research and Development*, Vol.28, No.1, 1984, 15-27.

Henry, S. and D. Kafura, "Software structure metrics based on information flow", *IEEE transactions on Software Engineering*, No. 5, 1981, 510-518.

Kim, B.H., S.Y. Han, and Y.C. Kim, "Design of Advanced HITS Algorithm by Suitability for Importance-Evaluation of Web-Documents", *The Journal of Society for e-Business Studies*, Vol.8, No.2, 2003, 23-31.

(김분희, 한상용, 김영찬, "웹 문서 중요도 평가를 위한 적합도 향상 HITS 알고리즘 설계", *한국전자거래학회지*, 제8권, 제2호, 2003, 23-31.)

Kim, Y.H. and Y.J. Kim, "Social Network Analysis", PARKYOUNGSA, 2016.

(김용학, 김영진, "사회연결망 분석", 박영사, 2016.)

Lommerse, G., F. Nossin, L. Voinea, and A. Telea, "The visual code navigator : An interactive toolset for source code investigation", *In Information Visualization*, INFOVIS IEEE Symposium, 2005, 24-31.

Mancoridis, S., B.S. Mitchell, C. Rorres, Y. Chen, and E.R. Gansner, "Using automatic clustering to produce high-level system organizations of source code", *In Program Comprehension*, IWPC 1998, Proceedings, 6th International Workshop, 1998, 45-52.

McCabe, T.J., "A complexity measure", *IEEE Transactions on Software Engineering*, No.4, 1976, 308-320.

Nagappan, N. and T. Ball, "Using software dependencies and churn metrics to predict field failures : An empirical case study", *In Empirical Software Engineering and Measurement*, ESEM First International Symposium on, IEEE, 2007, 364-373.

Pinzger, M., K. Graefenhain, P. Knab, and H.C. Gall, "A tool for visual understanding of source code dependencies", *In Program Comprehension*, ICPC 2008, The 16th IEEE International Conference on, IEEE, 2008, 254-259.

Rico, D.F., "How to estimate ROI for inspections, PSP sm, TSP sm, SW-CMM ISO 9000, and CMMI sm", *The DOD Software Tech News*, Vol.5, No.4, 2002, 23-31.

Seo, K.S. and H.M. Ahn, "Urban railway train operation efficiency studies using DEA", Autumn Conference & Annual Meeting of the The Korean Society For Railway, 2016, 456-449.

(서경수, 안현미, "DEA(자료포락분석)를 이용한 도시철도 전동차 운영효율 연구", 한국철도학회 추계학술대회논문집, 2016, 446-459.)

Singer, J., C. Tjortjis, and M. Ward, "Using software metrics to evaluate static single assignment form in GCC", University of Ioannina-Greece, University of Western Macedonia-Greece, 2010.

Van Den Brink, C. and S.W. Han, "Application of social network analysis for analyzing the relationships between root and direct causes of defects", *Modern Applied Science*, Vol.9, No.12, 2015, 12-20.

Vanhatalo, J., H. Völzer, and F. Leymann, "Faster and more focused control-flow analysis for business process models through SESE decomposition", *International Conference on Service-Oriented Computing*, Springer, Berlin, Heidelberg, 2007, 43-55.

Wikipedia, "Social Network Analysis", 2018,

Available at https://en.wikipedia.org/wiki/Social_network_analysis (Accessed June 12, 2018.)

Zafarani, R., M.A. Abbasi, and H. Liu, "Social media mining: an introduction", Cambridge University Press, 2014.

Zanetti, M.S., I. Scholtes, C.J. Tessone, and F. Schweitzer, "Categorizing bugs with social networks : A case study on four open source software communities", In Proceedings of the 2013 International Conference on Software Engineering, *IEEE Press*, 2013, 1032-1041.

Zimmermann, T. and N. Nagappan, "Predicting defects using network analysis on dependency graphs", *Software Engineering*, 2008, ICSE'08, ACM/IEEE 30th International Conference on, IEEE, 2008, 531-540.

About the Authors

Sang Moo Huh (norhuh@empal.com)

Sang Moo Huh received the Master degree in Industrial and Information Systems Engineering in 2012 at Seoul National University of Science & Technology. He is a Ph. D. Candidate in Industrial and Information Systems Engineering at Seoul National University of Science & Technology. His current research interests include software engineering, software quality, software defect, IT service, etc.

Woo Je Kim (wjkim@seoultech.ac.kr)

Professor Woo Je Kim received the Bachelor degree and Master degree and Doctor degree in Industrial Engineering from Seoul National University. He has been working for Department of Industrial and Systems Engineering, Seoul National University of Science & Technology. His current research interests include optimization, IT service, Software Engineering, etc.