제26권 제3호(통권 99호) 2013년 9월 ISSN 1738-6322

소프트웨어공학소사이어티 논문지

Journal of Software Engineering Society

VOLUME 26, NUMBER 3, September 2013

사단법인 한국정보과학회
소프트웨어공학소사이어티

Journal of Software Engineering Society

VOLUME 26, NUMBER 3, September 2013

Korean Institute of Information Scientists and Engineers
Software Engineering Society

안드로이드 디바이스를 위한 에너지 효율적 컨텍스트 모니터링 기법†

(Energy-Efficient Context Monitoring Methods for Android Devices)

김문권‡ (Moon Kwon Kim) 이재유§ (Jae Yoo Lee) 김수동¶ (Soo Dong Kim)

요 약 스마트 디바이스의 보급과 함께 컨텍스트 인지 애플리케이션에 대한 수요가 증가하는 추세이다. 하지만, 센서를 통한 컨텍스트의 수집은 추가적인 배터리 소비가 요구된다. 이것은 제한된 배터리 수명을 가진 모바일 디바이스에서 다수의 컨텍스트 인지 애플리케이션을 구동하는데 큰 제약 사항이 된다. 그러므로 에너지 효율적인 컨텍스트 모니터링 기법이 요구된다. 본 논문에서는 안드로이드 디바이스에서 가능한 네 가지 컨텍스트 모니터링 기법을 제시하고, 각 기법 간의 에너지 소모량을 분석하여 적합한 적용 지침을 제안한다. 본 논문에서 제안된 기법을 컨텍스트 모니터링 애플리케이션에 적용하면 컨텍스트 모니터링에 소비되는 에너지 소모량을 효과적으로 감소시킬 수 있다. 제안된 기법을 검증하기 위해 적용하여 사용자 행동 인지 애플리케이션을 개발하여 각 기법에 대한 에너지 소모량을 정량적으로 비교한다.

키워드 모바일 컴퓨팅, 컨텍스트 모니터링, 에너지 효율, 안드로이드

Abstract Along with increasing supplies of smart devices, a proliferation of context-aware applications is came. However, acquiring contexts through sensors requires considerable energy consumption. It has became big constraints on running many context-aware applications in mobile devices having limited battery capacity. Hence, energy-efficient methods for monitoring contexts are highly required. In this paper, we propose four context monitoring methods, analyse energy consumption in each method, and provide guidelines for applying the methods. It is effective to decrease energy consumption for monitoring contexts with applying the methods. To assess the proposed methods, we implement an application that is aware of a user's motion and show quantitative comparison between each of the methods.

Key words Mobile Computing, Context Monitoring, Energy Efficiency, Android

† 본 연구는 산업통상자원부의 지원을 받는 로봇산업원천기술개발 사업의 연구결과로 수행되었음.
‡ 학생회원 : 숭실대학교 컴퓨터학과
mkdmkk@gmail.com
§ 학생회원 : 숭실대학교 컴퓨터학과
jaeyoo1981@gmail.com (Corresponding Author)
¶ 종신회원 : 숭실대학교 컴퓨터학부 교수
sdkim777@gmail.com

1. 서 론

센싱 능력을 보유한 스마트 디바이스 보급의 증가로 인해 사용자 컨텍스트를 기반으로한 개인화된 애플리케이션에 대한 수요가 증가하는 추세이다[1]. 컨텍스트는 어떠한 상황을 특징지을 수 있는 정보로서 컨텍스트 인지 애플리케이션은 스마트 디바이스의 센서로부터 수집되는 미가공

데이터를 이용하여 특정 상황의 변화에 동적으로 반응할 수 있어 전통적인 애플리케이션과는 차별화된 특징을 제공한다.

스마트 디바이스의 컨텍스트 수집은 디바이스에 장착된 다양한 하드웨어 센서를 활용한다. 따라서, 컨텍스트 인지 애플리케이션의 실행은 센서를 구동하는데 필요한 추가적인 에너지 소모가 요구된다[2][3][4]. <표 1>은 세 가지 안드로이드 디바이스의 배터리 용량, 가용센서의 수, 모든 가용 센서를 통해 컨텍스트를 한번 측청했을 때의 배터리 소모량, 배터리가 방전될 때까지의 소요 시간을 보여준다. 본 연구의 실험을 통해 도출한 이 결과에 따르면 배터리 수명이 제한적인 스마트 디바이스에서 지속적인 컨텍스트 모니터링을 유지하기 힘들다.

<표 1> 스마트 디바이스의 센싱 배터리 소모량

측정치 \ 디바이스	Nexus 7	Galaxy S3	Galaxy Note 1
배터리 용량(mAh)	3950	2100	2500
가용센서 수	12	10	9
배터리 소모량(mA)	1.76×10^{-10}	1.01×10^{-10}	0.96×10^{-10}
배터리 방전 시간	1시간 44분	1시간 36분	1시간 56분

본 논문에서는 안드로이드 디바이스에서 에너지 효율성을 고려한 컨텍스트 모니터링 기법과 각 기법의 적용 지침을 제안한다. 컨텍스트 모니터링을 필요로하는 애플리케이션은 각 기법의 적용 지침을 기반으로 적합한 모니터링 기법을 선정하고 적용할 수 있다.

2. 관련 연구

Wissen[5]의 연구에서는 에너지 효율성이 높은 컨텍스트 모니터링을 위한 표현 기반(Expression-Based) 프레임워크를 제시한다. 이 프레임워크는 클라이언트 애플리케이션이 개별적으로 컨텍스트 모니터링 하지 않고 특정 표현을 만족하는 상황에서 이벤트를 받는 방식을 제공한다. 이 연구에서는 상황을 판단하여 이벤트를 받는 방식만을 고려하여 다양한 도메인에 적용이 어려울 수 있으며, 에너지 효율성에 대한 정량적 검증이 이루어지지 않았다.

Kramer[6]의 연구에서는 여러 컨텍스트 획득 방법을 제공하는 컨텍스트 엔진을 제시한다. 이 엔진의 주 목적은 컨텍스트 획득 및 제공으로, 여러 클라이언트 애플리케이션에서 이 엔진을 이용하여 여러 종류의 컨텍스트를 포함한 컨텍스트 집합을 획득할 수 있도록 하는 것이다. 이 연구는 클라이언트 애플리케이션이 개별적으로 컨텍스트 모니터링을 수행해야 하는 하는 것을 컨텍스트 엔진에서 컨텍스트 모니터링을 전담하여 효율성을 높일 것으로 기대되지만 이에 대한 정량적인 검증이 이루어지지 않았으며, 각 컨텍스트 획득 방법 적용에 대한 적합한 상황을 제시하고 있지 않다.

Taleb[7]의 연구에서는 에너지 효율성, 센서의 정확도를 고려하여 컨텍스트 모니터링에 사용될 센서를 선택하는 알고리즘을 제시한다. 이 연구에서는 각 센서마다 최소 에너지 소모량, 정확도를 고려하여 센서의 효율성 메트릭을 제시하고 센서 선택 알고리즘에서 이 메트릭을 이용한다. 본 연구에서 제시하는 메트릭은 컨텍스트 획득에 대한 에너지 소모량만을 고려하여 컨텍스트를 처리하는 에너지 소모량, 컨텍스트를 관리하는 에너지 소모량, 컨텍스트 애플리케이션의 개수 등, 컨텍스트 모니터링의 에너지 효율성에 영향을 주는 여러 요소들을 고려하지 않고 있다.

기존 연구에서는 컨텍스트 모니터링 기법에 대한 고려와 각 기법별 정량적인 에너지 효율성 검증이 부족하고, 컨텍스트 모니터링의 다양한 기법에 대한 장단점, 적합한 적용 상황, 적용 지침에 대한 고려가 부족하다. 그러므로 본 연구

에서는 컨텍스트 모니터링 기법의 분류 기준과 그 분류 기준으로부터 컨텍스트 모니터링 기법 분류를 도출하고, 각 분류에서 에너지 효율성을 측정하기 위한 메트릭을 제시한다. 또한 각 기법을 구현하고 제시한 메트릭을 이용하여 각 기법의 에너지 효율성을 비교 및 검증한다.

3. 컨텍스트 모니터링 기법의 분류 기준 및 에너지 효율성 메트릭

본 장에서는 안드로이드의 특성을 고려하여 컨텍스트 모니터링 기법을 제안 및 분류하고 각 기법의 에너지 효율성 측정을 위한 메트릭을 제시한다.

3.1 컨텍스트 모니터링 기법의 분류 기준

본 절에서는 안드로이드 디바이스에서 가능한 컨텍스트 모니터링 기법의 분류 기준을 제시하고, 제시된 기준에 따른 네 가지 컨텍스트 모니터링 기법을 제안한다.

분류 기준 1. 컨텍스트 획득 방식(Acquisition Method): 컨텍스트 모니터링을 위한 메시지 방식은 푸싱(Pushing)과 풀링(Pulling)으로 구분된다. 푸싱은 컨텍스트 이벤트를 감지하여 비동기적으로 컨텍스트를 획득하는 방식이다. 따라서 컨텍스트의 변화에 즉각적인 반응이 가능하지만 클라이언트 애플리케이션의 필요에 따른 능동적인 컨텍스트 획득이 어렵고, 지속적인 컨텍스트 이벤트 감지로 인해 에너지 소모가 크다. 반면에, 풀링 방식은 클라이언트 애플리케이션에서 능동적인 컨텍스트의 획득이 가능하다. 안드로이드 서비스 빌딩 블록은 컨텍스트를 수집 및 관리하고, 다수의 클라이언트 애플리케이션으로 풀링 방식의 컨텍스트 모니터링 기능을 제공하기에 알맞다. 컨텍스트의 수집 및 관리를 위한 공통적인 기능성을 재사용 가능한 서비스를 통해 제공하여 클라이언트 애플리케이션에서 중복 컨텍스트 수집 기능을 제거한다[8][9].

분류 기준 2. 획득하고자 하는 컨텍스트의 시점(Timeframe): 컨텍스트의 시점에는 현재, 최근, 과거가 있다. 현재(Current) 컨텍스트는 수집된 가장 최근의 컨텍스트이다. 최근(Recent) 컨텍스트는 일정 기간 이전부터 현재까지 수집된 컨텍스트이다. 과거(Past) 컨텍스트는 일정 기간 이전의 컨텍스트를 의미한다.

제시된 두 가지 분류 기준을 기반으로 모두 여섯 가지의 컨텍스트 모니터링 기법을 도출할 수 있지만 푸싱 방식은 이벤트 발생에 따라 현재 컨텍스트를 갱신하는 방식으로 최신 컨텍스트 또는 과거 컨텍스트에 대하여 적용하는 것은 에너지 효율성의 측면에서 효과적이지 않다.

기법 1은 안드로이드에서의 기본 컨텍스트 수집 방식으로, 시스템으로부터 푸싱 방식으로 현재 미가공 컨텍스트를 수집한다. 기법 2는 서비스를 이용하여 현재 미가공 컨텍스트를 클라이언트에게 풀링 방식으로 제공하며, 기법 3은 최근 컨텍스트를 기법 2와 같은 방식으로 클라이언트에게 제공한다. 기법 4는 과거 미가공 컨텍스트를 콘텐트 프로바이더를 이용하여 클라이언트에게 제공한다.

3.2 에너지 소모량 메트릭

컨텍스트 모니터링을 위한 에너지 소모량(Energy Consumption, ENG)은 컨텍스트 수집(Acquisition, ACQ), 컨텍스트 획득(Retrieval, RET), 획득한 컨텍스트 관리(Managmenet, MAN), 획득한 컨텍스트 처리(Processing, PRO)에 필요한 에너지 소모량으로 이루어진다. 컨텍스트 모니터링을 위한 1분간의 에너지 소모량은 아래와 같이 정의되며 단위는 mAh이다. 아래 수식의 인자는 <표 2>와 같다.

$$ENG = (ACQ + MAN) \times S + \sum_{i=1}^{N} ACQ \times C_i + RET \times R_i + PRO_i \times (C_i + R_i)$$

<표 2> 메트릭 *ENG*의 인자

인자	설명
N	클라이언트 수
S	모니터링 서비스에서의 컨텍스트 수집 수
C	클라이언트에서의 컨텍스트 수집 수
R	서비스로부터 획득한 컨텍스트 집합의 수
ACQ	컨텍스트를 수집 비용
MAN	모니터링 서비스가 수집한 컨텍스트 관리 비용
RET	서비스로부터의 컨텍스트 집합 획득 비용
PRO	클라이언트에서 획득한 컨텍스트/이벤트 처리 비용

각 기법의 에너지 사용량을 측정하기 위한 ENG에서 고려되지 않는 인자를 소거하여 두 가지의 변형 메트릭, ENGa와 ENGb를 제시한다.

기법 1에서는 서비스를 사용하지 않고 각 클라이언트 애플리케이션에서 시스템으로부터 컨텍스트를 수집하기 때문에 S, R, E, MAN, RET를 소거하여 다음과 같은 메트릭 ENGa를 에너지 사용량을 구하는데 사용할 수 있다.

$$ENG^a = \sum_{i=1}^{N} (ACQ + PRO_i) \times C_i$$

기법 2, 기법 3, 기법 4에서는 서비스를 이용하여 시스템으로부터 컨텍스트를 수집하고 관리하고 클라이언트 애플리케이션이 서비스로부터 컨텍스트를 풀링 방식으로 획득하기 때문에 C, E를 소거하여 다음과 같은 메트릭 ENGb를 에너지 사용량을 구하는데 사용할 수 있다.

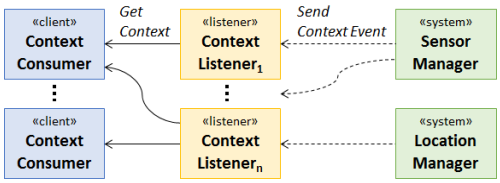
$$ENG^b = (ACQ + MAN) \times \sum_{i=1}^{N} (RET + PRO_i) \times R_i$$

4. 에너지 효율성을 고려한 컨텍스트 모니터링 기법의 설계 및 적용 지침

본 장에서는 각 기법에 대한 특징, 장단점을 소개하고 각 기법을 위한 설계 모델을 제시한다.

4.1 기본 컨텍스트 모니터링 기법

안드로이드에서 제공하는 기본 컨텍스트 모니터링 기법을 이용하여 가용한 하드웨어 센서 및 소프트웨어 센서를 파악하고 센서 이벤트 리스너와 로케이션 리스너를 컨텍스트 매니저에 등록하고 컨텍스트 이벤트를 받는다. 본 기법은 컨텍스트 획득을 필요로하는 모든 클라이언트 애플리케이션에서 위의 동일한 절차를 수행해야 한다. [그림 1]은 안드로이드에서 제공하는 기본 컨텍스트 모니터링 기법의 구조를 보여준다.



[그림 1] 기본 컨텍스트 모니터링 구조

기법 1은 메트릭 ENGa의 결과 값이 작게 나오는 상황에 적합하다. 즉, 컨텍스트 수집 에너지 소모량과 컨텍스트 처리 에너지 소모량을 줄여야 한다.

클라이언트 애플리케이션의 수(N): 클라이언트 애플리케이션 별로 컨텍스트 이벤트를 처리해야 한다. 컨텍스트 이벤트는 다량으로 발생되기 때문에, 컨텍스트 애플리케이션의 수가 증가함에 따라 컨텍스트 수집 비용과 컨텍스트 처리 비용이 증가한다. 그러므로 컨텍스트를 획득 및 처리하는 클라이언트 애플리케이션의 수가 적어야 효율적이다.

컨텍스트 수집 주기: 기법 1에서 컨텍스트 수집 에너지 소모량을 줄이는 방법은 컨텍스트 이벤트 주기를 길게 하여서 컨텍스트 수집을 적게 하는

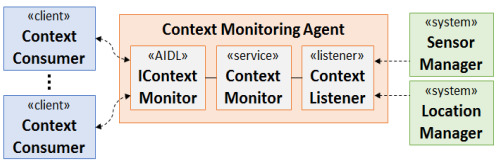
것이다. 센서 이벤트 리스너와 로케이션 리스너를 안드로이드 시스템에 등록할 시점에 이벤트 발생 주기를 정할 수 있다.

컨텍스트 처리 복잡도: 다량의 컨텍스트 이벤트를 처리하기 때문에 컨텍스트 처리의 복잡도가 크면 컨텍스트 처리 비용 크게 증가한다.

4.2 현재 컨텍스트 모니터링 기법

안드로이드에서는 Background 프로세싱을 위해 서비스(Service)와 프로세스 간의 통신을 위한 Android Interface Definition Language(AIDL) [10]을 제공한다. 기법 2는 Background 에서 컨텍스트를 수집 및 관리하기 위한 서비스인 컨텍스트 모니터(ContextMonitor)와 컨텍스트 모니터가 수집한 컨텍스트를 클라이언트에게 제공하기 위한 AIDL 인터페이스인 컨텍스트 모니터 인터페이스(IContextMonitor)를 활용한다.

컨텍스트 모니터가 센서 매니저와 로케이션 매니저에 컨텍스트 리스너를 등록하여 지속적으로 컨텍스트를 획득하고, 최신 컨텍스트를 메모리상에 유지한다. 그러므로 클라이언트 애플리케이션은 컨텍스트 획득을 위해 개별적으로 안드로이드 센서 매니저에 센서 이벤트 리스너를 등록할 필요 없이, 컨텍스트 모니터 인터페이스를 통해 컨텍스트를 필요에 따라 획득한다. [그림 2]는 현재 컨텍스트 모니터링 기법의 구조를 보여준다.



[그림 2] 현재 컨텍스트 모니터링 기법의 구조

기법 2에서는 클라이언트 애플리케이션이 컨텍스트를 풀링 방식으로 획득 및 처리하기 때문에, 기법 1에서 푸싱 방식으로 발생하는 오버헤드를 줄인다. 이 방식은 클라이언트 애플리케이션의 수가 증가함에 따라 더욱 효율적이다. 하지만 기법 2는 최신 컨텍스트만을 유지하기 때문에, 클라이언트 애플리케이션이 축적된 컨텍스트를 요구하는 경우 적합하지 않다. 또한, 서비스로부터 컨텍스트를 획득하는 주기가 짧을 경우, 서비스로부터 컨텍스트를 획득하기 위한 오버헤드로 인해 본 기법을 적용하는 것이 효율적이지 않을 수 있다. 즉, ENGb를 구하기 위한 모든 인자값이 주어졌을 때, 다음과 같은 조건이면 본 기법을 적용하는 것이 효율적이다.

$$\frac{\sum_{i=1}^{N}(RET+PRO_i)\times R_i}{\frac{\sum_{i=1}^{N}PRO_i}{N}\times S}<1$$

이 수식에서 분모 부분은 기법 1을 적용했을 때의 예측 에너지 소모량이다. 본 기법의 컨텍스트 관리 비용(MAN)은 에너지 효율성에 큰 영향을 미치지 않기 때문에 고려하지 않는다.

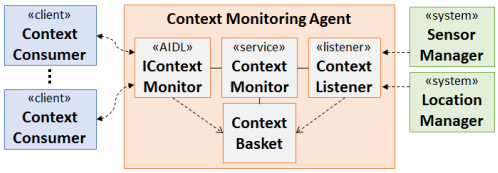
컨텍스트 획득 횟수: 클라이언트 애플리케이션이 필요로 하는 컨텍스트 획득 주기는 안드로이드 시스템이 기본적으로 제공하는 컨텍스트 이벤트 주기보다 길다. 이러한 애플리케이션에서 기법 1을 적용하는 것은 필요치 않게 많은 컨텍스트 처리 작업을 수행해야하므로 효율적이지 않다.

클라이언트 애플리케이션의 수(N): 클라이언트 애플리케이션의 수(N)가 복수개이면 기법 1보다 기법 2가 효율적이다. 기법 2는 컨텍스트 모니터링을 복수개의 클라이언트 애플리케이션에서 개별적으로 수행하지 않고 서비스가 전담하면 중복적인 작업을 제거하여 에너지 효율성 관점에서 좋다.

컨텍스트 처리 복잡도(PRO): 컨텍스트 처리 작업의 복잡도가 높을수록 위의 수식이 0에 더 가까워지기 때문에 본 기법의 적용이 기법 1을 적용하는 것에 비해 효과적이다.

4.3 최근 컨텍스트 모니터링 기법

컨텍스트 바스켓(ContextBasket)은 최근 컨텍스트를 관리하는 클래스로 복수개의 컨텍스트를 메모리 상에서 저장하기 위한 큐(Queue)를 가진다. 기법 3은 컨텍스트 바스켓을 이용하여 컨텍스트 리스너를 통해 수집된 최근 컨텍스트를 유지한다. 클라이언트 애플리케이션은 최신 컨텍스트 뿐만 아니라 축적된 최근의 컨텍스트를 컨텍스트 모니터 인터페이스를 통해 획득할 수 있다. [그림 3]은 최근 컨텍스트 모니터링 기법의 구조를 보여준다.



[그림 3] 최근 컨텍스트 모니터링 기법의 구조

기법 3에서 컨텍스트 모니터는 축적된 최근 컨텍스트 집합을 클라이언트 애플리케이션에게 제공함으로써 클라이언트 애플리케이션이 연속적인 데이터를 가공해야 하는 패턴 인식[11]과 같은 프로세싱을 효율적으로 수행할 수 있도록 한다. 즉, 클라이언트 애플리케이션에서 컨텍스트를 요청하는 횟수를 줄일 수 있는 기회가 많아진다. 하지만 짧은 시간에 다량의 데이터가 발생하는 컨텍스트 모니터링의 특성에 의해 컨텍스트 바스켓 관리 오버헤드가 발생한다. 즉, ENGb를 구하기 위한 모든 인자값이 주어졌을 때, 다음과 같은 조건이면 본 기법을 적용하는 것이 효율적이다.

$$\frac{MAN\times S+\sum_{i=1}^{N}(RET+PRO_i)\times R_i}{\sum_{i=1}^{N}(RET\times\alpha+PRO_i)\times R_i\times\beta}<1$$

이 수식에서 분모 부분은 기법 2를 적용했을 때의 예측 에너지 소모량이다.

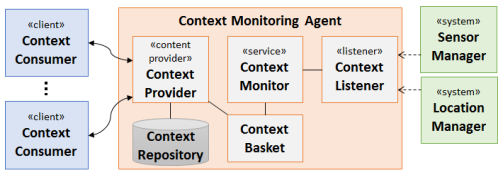
컨텍스트 획득 오버헤드(α): 기법 3에서의 컨텍스트 획득 오버헤드(RET)와 기법 2에서의 컨텍스트 획득 오버헤드(RET$\times\alpha$)는 차이가 크지 않다. 그러므로 α를 구하지 못할 때는 1로 적용해도 효율성 비교에 큰 영향을 주지 않는다.

컨텍스트 획득수 비율(β): 일반적으로, 기법 2에서의 컨텍스트 획득 횟수(R$\times\beta$)에 비해 본 기법의 컨텍스트 획득 횟수(R)가 적다. 왜냐하면 기법 2에서는 최신 컨텍스트만을 제공하고 연속적인 컨텍스트가 필요한 클라이언트 애플리케이션이 짧은 주기로 서비스로부터 컨텍스트를 획득하기 때문이다. 즉, 기법 2와 기법 3의 컨텍스트 획득수 비율인 β가 클수록 본 기법의 효율성은 더욱 높아진다.

클라이언트 애플리케이션의 수(N): 클라이언트 애플리케이션의 수(N)가 많아지면 기법 2을 사용한 예상 컨텍스트 획득수가 가증되므로(R$\times\alpha\times$N) 기법 3의 적용이 더욱 효과적이다.

4.4 과거 컨텍스트 모니터링 기법

기법 3에서 제공하는 최신 컨텍스트 이전에 발생한 과거 컨텍스트를 획득하기 위한 기법이다. 기법 3에서 수집된 최신 컨텍스트는 Context Basket에 저장되고, 지속적으로 갱신된다. Context Basket에서 제외되는 컨텍스트는 과거 컨텍스트로 취급되며, 안드로이드 콘텐트 프로바이더(Content Provider) 형태의 컨텍스트 프로바이더(Context Provider)를 통해 SQLite 데이터베이스의 컨텍스트 저장소(Context Repository)에 저장된다. [그림 4]는 과거 컨텍스트 모니터링 기법의 구조를 보여준다.



[그림 4] 과거 컨텍스트 모니터링 기법의 구조

새로운 컨텍스트 이벤트가 발생하면, 컨텍스트 모니터는 새로운 컨텍스트를 Context Basket에 추가한다. 저장된 컨텍스트의 수가 정해진 Context Basket의 크기를 초과하게 되면, 저장된 순서에 따라 가장 먼저 수집된 컨텍스트가 Context Basket에서 제거되고, 컨텍스트 프로바이더에게 전달된다.

기법 4는 축적되는 콘텐트 프로바이더를 통해 과거 현재까지의 컨텍스트 집합을 클라이언트 애플리케이션에게 제공함으로써 클라이언트 애플리케이션이 컨텍스트의 집합으로부터 사용자의 패턴 분석이나 새로운 가치를 창출하기 위한 컨텍스트 처리과정을 효율적으로 수행할 수 있도록 한다. 따라서, 클라이언트 애플리케이션은 특정 기간의 컨텍스트 집합을 컨텍스트 모니터링 에이전트 서비스로부터 획득하여 각 애플리케이션의 목적에 따라 가공하는 기능만을 수행한다. 짧은 주기로 현재 혹은 최근 컨텍스트를 획득해야 하는 기법 2와 3에 비하여 필요한 시점에 컨텍스트를 획득하는 기법 4의 컨텍스트 요청 횟수가 보다 적어진다. 따라서, ENGb를 구하기 위한 모든 인자 값이 주어졌을 때, 다음과 같은 조건이면 본 기법을 적용하는 것이 효율적이다.

$$\frac{MAN\times S+\sum_{i=1}^{N}(RET+PRO_i)\times R_i}{MAN\times\alpha\times S+\sum_{i=1}^{N}(RET\times\beta+PRO_i)\times R_i\times\gamma}<1$$

이 수식에서 분모 부분은 기법 3를 적용했을 때의 예측 에너지 소모량이다.

컨텍스트 관리 오버헤드(α): 기법 3에 비해 기법 4는 수집한 컨텍스트를 콘텐트 프로바이더를 이용하여 데이터베이스에 저장하기 때문에 컨텍스트 관리 비용이 크다. 즉 α는 1보다 작은 값을 가진다.

컨텍스트 획득 오버헤드(β): 기법 4에서는 클라이언트 애플리케이션이 컨텍스트 획득을 위해 콘텐트 프로바이더를 이용하여 데이터베이스에 연결하기 때문에 컨텍스트 획득 비용이 기법 3에 비해 크다. 즉 β는 1보다 작은 값을 가진다.

컨텍스트 획득수(γ): 앞서 언급한 컨텍스트 관리 오버헤드와 컨텍스트 획득 오버헤드를 감당하면서도 본 기법이 효율적이려면 컨텍스트의 획득 수가 기법 3에 비해 상당히 적어야 한다. 즉 γ이 큰 값을 가져서 기법 4의 컨텍스트 획득 비용과 컨텍스트 처리 비용을 줄여야 한다. 이는 기법 4의 경우 주기적인 컨텍스트 획득에는 큰 효율을 얻을 수 없음을 의미한다.

5. 실험 및 평가

본 장에서는 컨텍스트 모니터링 기법 별로 에너지 소모량 메트릭을 이용하여 에너지 소모량을 측정하고 비교한다. 기법 별 적용 지침을 기반으로 본 실험시나리오에 가장 적합한 컨텍스트 모니터링 기법을 확인한다.

5.1 실험 환경 및 시나리오

본 실험은 Nexus 7에서 수행되었으며, 에너지 소모량을 구하기 위해, 1%의 배터리가 소모되는 동안의 시간을 측정하여 1분동안의 배터리 소모량(ENG)을 유도한다. Nexus 7의 전체 배터리량은 3,950mAh이고, 1%의 배터리량은 39.5mAh이다. 실험 환경을 동일하게 하기 위해 본 실험과 무관한 애플리케이션은 모두 종료하고 화면 밝기를 최소로 유지한다.

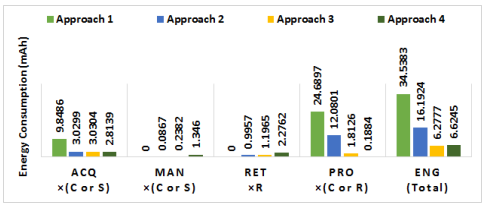
본 실험에서는 획득한 컨텍스트를 이용하여 사용자의 행동을 인지한다. 이를 위해 분류(Classification) 알고리즘인 J48을 사용하고 학습 데이터로 [12]에서 적용한 사용자 행동 데이터를 사용하였다. 기법 1은 클라이언트에서 지속적으로 모니터링한다. 기법 2에서는 0.1초의 주기로 서비스로부터 최신 컨텍스트를 획득하고 기법 3에서는

1초의 주기로 최근 10개의 컨텍스트를 획득하고 기법 4에서는 10초의 주기로 100개의 컨텍스트를 획득한다.

5.2 컨텍스트 모니터링 기법 별 실험 결과

본 실험은 각 기법에서 세 개의 동일한 클라이언트 애플리케이션(N=3)이 컨텍스트를 획득하고, 5회의 실험을 통해 얻은 평균값을 사용한다.

실험 결과를 통해 기법 별로 각 세부 에너지 소모 비용과 전체 에너지 소모 비용을 [그림 5]와 같이 비교한다. 각 에너지 소모량 항목의 막대 그래프는 5개의 막대를 가지며, 각 막대는 순서대로 기법 1, 기법 2, 기법 3, 기법 4를 의미한다.



[그림 5] 시나리오의 기법 별 에너지 소모량 (N=3)

컨텍스트 수집 에너지 소모량(ACQ)은 기법 1의 값이 가장 높으며, 기법 2, 기법 3, 기법 4는 서로 비슷한 값을 가진다. 이는 각각의 클라이언트 애플리케이션에서 컨텍스트를 수집하는 기법 1의 방식과 그 이외 기법의 방식인 서비스에서 컨텍스트 수집을 전담하는 것과의 차이를 보여준다. 컨텍스트 관리 에너지 소모량(MAN)의 경우 기법 2에서는 매우 적은 양의 에너지 소모를 보이며, 기법 3에서도 값이 크지 않지만, 기법 4에서는 비교적 큰 에너지 소모량을 보인다. 컨텍스트 획득 에너지 소모량(RET)은 기법 2에서 가장 적은 값을 가지며 기법 3에서는 기법 2보다는 값이 크지만 큰 차이를 보이진 않는다. 그러나 기법 4에서는 콘텐트 프로바이더를 사용하기 때문에 비교적 큰 에너지 소모량을 보인다. 컨텍스트 처리 에너지 소모량(PRO)은 기법 4에서 가장 적은 값을 가진다. 이는 한번에 대량의 컨텍스트를 획득하여 컨텍스트 처리 횟수를 줄이기 때문이다. 반면에 기법 1에서는 컨텍스트를 클라이언트 애플리케이션에서 수집하여 처리하기 때문에 컨텍스트 처리 횟수가 많다. 기법 2의 컨텍스트 처리 횟수가 기법 3의 컨텍스트 처리 횟수보다 많기 때문에 에너지 소모량 역시 기법 2에서 많다.

본 실험의 시나리오는 최근의 컨텍스트를 기반으로 사용자의 행동을 판단하는 것이다. 행동 판단은 일련의 복수 가속도 컨텍스트를 통해 이루어진다. 그러므로 한번에 복수개의 컨텍스트를 획득 함으로써 컨텍스트 획득수를 줄이는 기법 3이 적합하다. 이 결과 기반으로 4장에서 제시한 기법 3과 기법 2의 비교 메트릭을 적용(α =1, β =10)하면 아래와 같이 결과 값이 1보다 작다.

$$\frac{MAN\times S+\sum_{i=1}^{N}(RET+PRO_i)\times R_i}{\sum_{i=1}^{N}(RET\times\alpha+PRO_i)\times R_i\times\beta}=\frac{0.2382+1.1965+1.8126}{(1.1965+1.8126)\times 10}=0.1044$$

그러므로 본 시나리오에서는 기법 3을 사용하는 것이 기법 2를 사용하는 것보다 에너지 효율성 측면에서 더 적합하다.

6. 결 론

스마트 디바이스의 보급과 함께 사용자의 상황을 동적으로 반영할 수 있는 컨텍스트 인지 애플리케이션에 대한 수요가 증가하는 추세이다. 스마트 디바이스의 제한된 배터리 수명은 센서를 이용한 컨텍스트 모니터링의 지속적인 수행 측면에서 주요 제약사항이다. 본 논문에서는 컨텍스트 모니터링을 위한 에너지 소모를 줄이기 위해, 안드로이드의 특성을 고려한 네 가지 컨텍스트 모니터링 기법을 제안하였다. 제안된 네 가지 컨텍스트 모니터링 기법을 적용하여 각 기법 간의 에너지

소모량을 분석하고, 적합한 적용 지침을 제안하였다. 사용자의 행동을 인지하기 위한 애플리케이션을 각 기법 별로 구현하고 실제로 에너지 소모량을 측정해서 제안한 모니터링 기법 및 적용 지침을 검증하였다. 본 논문에서 제안된 기법을 적용 지침을 기반으로 컨텍스트 모니터링 애플리케이션에 적용하면 보다 에너지 효율적인 컨텍스트 모니터링이 가능하다.

참 고 문 헌

[1] MarketsAndMarkets, Context Aware Computing Market-Global Advancements, Emerging Applications, Worldwide Forecasts and Analysis (2013-2018), March 2013.

[2] Kjaergaard, M.B., "Location-based services on mobile phones: minimizing power consumption," Pervasive Computring, IEEE, Vol. 11, No. 1, pp. 67-73, January 2012.

[3] Priyantha, B., Lymberopoulos, D., Liu, J., "LittleRock: Enabling Energy-Efficient Continuous Sensing on Mobile Phones," Pervasive Computing, IEEE, Vol. 10, No. 2, pp. 12-15, April 2011.

[4] S. K. Datta, C. Bonnet, and N. Nikaein, "Android power management: Current and future trends," 2012 The First IEEE Workshop on Enabling Technologies for Smartphone and Internet of Things (ETSIoT2012), pp. 48–53, 2012.

[5] B. Van Wissen, N. Palmer, and R. Kemp, "Context Droid: an expression-based context framework for Android," In Proceedings of PhoneSense, 2010.

[6] D. Kramer, A. Kocurova, S. Oussena, T. Clark, and P. Komisarczuk, "An extensible, self contained, layered approach to context acquisition," In Proceedings of the Third International Workshop on Middleware for Pervasive Mobile and Embedded Computing, p. 6, Dec. 2011.

[7] S. Taleb, N. Abbas, H. Hajj, and Z. Dawy, "On sensor selection in mobile devices based on energy, application accuracy, and context metrics," In Proceedings of Third International Conference on Communications and Information Technology (ICCIT), 2013, pp. 12–16.

[8] G. Cardone, A. Cirri, A. Corradi, L. Foschini, and D. Maio, "MSF: An Efficient Mobile Phone Sensing Framework," International Journal of Distributed Sensor Networks, vol. 2013, Mar. 2013.

[9] Y. Wang, J. Lin, M. Annavaram, Q. A. Jacobson, J. Hong, B. Krishnamachari, and N. Sadeh, "A framework of energy efficient mobile sensing for automatic user state recognition," In Proceedings of the 7th international conference on Mobile systems, applications, and services, pp. 179–192, Jun. 2009.

[10] AIDL, Android Developers, http://developer.android.com/guide/components/aidl.html

[11] Sergios Theodoridis and Konstantinos Koutroumbas, Pattern Recognition, 4thEd.,AcademicPress,2008.

[12] J. R. Kwapisz, G. M. Weiss, and S. A. Moore, "Activity recognition using cell phone accelerometers," ACM SIGKDD Explorations Newsletter, vol. 12, no. 2, pp. 74–82, Mar. 2011.E79-A, No. 9, pp. 1338-1353, Sep., 1996.

저자 소개

김 문 권

2012년 숭실대학교 컴퓨터학과 학사
2013년 숭실대학교 컴퓨터학과 석사
2013년~현재 숭실대학교 컴퓨터학과 박사과정
관심분야는 소프트웨어공학, 모바일 컴퓨팅, 사물인터넷, 컨텍스트 추론

이 재 유

2007년 홍익대학교 컴퓨터학과 학사
2009년 숭실대학교 컴퓨터학과 석사
2009년~현재 숭실대학교 컴퓨터학과 박사과정
관심분야는 소프트웨어공학, 모바일 컴퓨팅, 사물인터넷, 컨텍스트 인지 서비스

김 수 동

1984년 미국 노스웨스트 미주리 주립대학 컴퓨터학과 학사
1988년 미국 아이오와 주립대학 컴퓨터학과 석사
1991년 미국 아이오와 주립대학 컴퓨터학과 박사
1991년~1993년 한국통신 연구개발단 선임연구원
1994년~1995년 현대전자 소프트웨어연구소 책임연구원
1995년~현재 숭실대학교 컴퓨터학과 교수
관심분야는 서비스 지향 아키텍처, 클라우드 컴퓨팅, 모바일 서비스, 객체지향 S/W공학, 컴포넌트 기반 개발, 소프트웨어 아키텍처

소프트웨어 모듈성을 정량적으로 측정하는 방법

(A method for quantitative measurement of software moduleness)

정필수[+] (Pilsu Jung) 안종선[++] (Jongsun Ahn) 박태현[+++] (Teahyun Park) 강성원[++++] (Sungwon Kang)
은나래[§] (Narae Eun) 고상원[¶] (Sangwon Ko)

요 약 소프트웨어의 품질을 평가하기 위해, 구성 모듈의 품질을 측정하는 것이 중요하다. 특히, 현업에서는 품질이 높은 모듈을 재사용하고 품질이 낮은 모듈을 개선하는 활동이 중요하기 때문에 모듈의 품질을 정량적으로 측정할 필요가 있다. 본 논문에서는 모듈이 지녀야 할 속성으로서 모듈성(moduleness)을 정의하고 모듈성을 구성하는 여러 가지 품질 속성과 품질 측면 및 측정 지표를 정의한다. 그리고 이들을 활용하여 모듈성을 정량적으로 측정하는 방법을 제안한다. 사례 연구에서는 C, C++, Java기반의 오픈소스 모듈들을 대상으로 모듈성을 정량적으로 측정하고 그 측정 결과를 검증함으로써 제안 방법의 효용성을 보인다.

키워드 소프트웨어 모듈성, 소프트웨어 품질 측정, 가중치 결정 방법, AHP

Abstract We need to measure the quality of each module to assess software quality. Especially, in industry, it is important to measure quality of modules quantitatively in order to improve bad modules and reuse good modules. In this paper, we define moduleness as characteristic that modules should be possessed and propose a method to measure software moduleness quantitatively. A case study conducted in this paper shows the usefulness of the method by measuring and evaluating moduleness of modules based on open source software developed by C, C++ and Java languages.

Key words Software moduleness, Software quality measurement, weighting method, Analytic hierarchy process

1. 서 론

소프트웨어 활용분야가 점점 넓어지면서 소프트웨어 품질은 중요한 이슈가 되었다[1]. 그 이유는 소프트웨어의 규모와 복잡도는 점점 증가하고 소프트웨어가 적용되는 도메인은 시간이 지남에 따라 다양해고 있기 때문이다. 현업에서는 레거시 소프트웨어에서 품질이 좋은 모듈을 재사용하여, 적은 비용으로 품질이 좋은 소프트웨어를 생산하기를 원한다. 또한, 품질이 나쁜 모듈들을 개선하여 해당 모듈이 사용된 소프트웨어의 전체 품질을 향상시키기를 원한다. 이를 위해, 모듈의 품질을 측정하여 평가함으로써 모듈의 현 수준을 평가할 수 있는 연구가 필요하다.

[+] 학생회원 : 한국과학기술원 전산학과 psjung@kaist.ac.kr
[++] 학생회원 : 한국과학기술원 전산학과 jsahn@kaist.ac.kr
[+++] 학생회원 : 한국과학기술원 전산학과 danapark@kaist.ac.kr
[++++] 종신회원 : 한국과학기술원 전산학과 교수 sungwon.kang@kaist.ac.kr
[§] 비회원 : LG전자 소프트웨어 아키텍처팀 narae.eun@lge.com
[¶] 비회원 : LG전자 소프트웨어 아키텍처팀 sangwon.ko@lge.com

소프트웨어 품질(software quality)이란 소프트웨어에 요구되는 품질 속성(quality attribute)들을 잘 반영하는 정도를 의미한다[1]. 품질 속성은 소프트웨어가 이해 관계자들의 요구를 얼마나 잘 반영하는지 나타내기 위해 사용되는 측정가능한 소프트웨어 특성이며[2], 이는 여러 가지 관련 품질 측면(quality concern)으로 구성된다. 품질 측면은 소프트웨어 품질 속성을 명세, 측정, 평가하기 위한 수단이다[3].

과거에 소프트웨어 품질을 정량적으로 측정하는 연구들이 많이 수행되었다[4,5,6]. 이들은 소프트웨어 품질을 여러 가지 품질 측면들로 나누고 각 품질 측면들을 측정하는 방법을 제시한다. 그러나 품질 측면들을 종합하여 소프트웨어의 수준을 평가하는 방법을 제시하지 않았기 때문에 현업에서 이를 활용하기 어렵다는 한계가 있다. 모듈로 구성된 소프트웨어의 모듈화 수준을 정량적으로 측정하기 위한 연구[7,8,9]도 수행되었다. 이들은 모듈화 수준을 평가하기 위한 품질 측면을 정의하고 그 측정 방법을 제안한다. 그러나 이들도 품질 측면들을 종합하여 평가하는 방법은 제시하지 않거나[8,9], 각 품질 측면들에 대한 가중치를 고려하지 않고 명확한 근거 없이 동일한 비중으로 더하여 계산하는 데 한계점을 지닌다[7]. 따라서 기존 연구들의 제안 방법을 활용하여 소프트웨어 모듈의 품질을 정량적으로 측정하기 어렵다.

본 논문은 소프트웨어 모듈성(moduleness)을 구성하는 여러 가지 품질 속성들과 품질 측면들을 정의하고 이들을 정량적으로 측정하는 방법을 제안한다. 제안 방법을 통해, 모듈의 현 수준을 객관적으로 파악할 수 있고, 객관적 수치를 활용하여 해당 모듈을 새로운 소프트웨어 개발에 재사용할지 또는 새롭게 개발할지를 결정하는 기준을 세울 수 있다. 또한, 재사용 가능하다면 추가적인 개선이 필요한지의 여부를 정량적 수치를 활용하여 결정할 수 있다.

본 논문의 구성은 다음과 같다. 제2절에서는 소프트웨어 품질을 측정하는 기존 연구들을 소개한다. 제3절에서는 모듈성을 정량적으로 측정하는 방법을 제시하고 제4절에서는 오픈소스를 활용한 사례 연구를 통하여 제안 방법의 효용성을 보인다. 끝으로, 제5절에서는 제안 방법의 기여와 향후 연구 방향에 대해 논의한다.

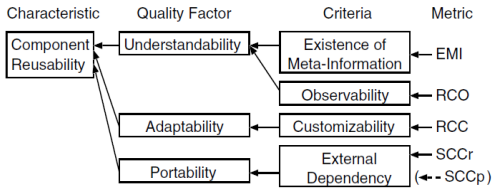
2. 소프트웨어 품질 평가에 관한 기존 연구

과거에 소프트웨어의 품질을 평가하기 위해 품질 속성을 구성하는 여러 가지 품질 측면들을 정의하고 이들을 정량적으로 측정하는 연구들이 수행되어 왔다. 과거 연구들은 객체지향 설계 품질을 측정하기 위한 연구, 소프트웨어 모듈화 수준을 측정하는 연구, ISO/IEC 9126 표준 또는 ISO/IEC 25010 표준에서 정의한 품질 속성을 측정하는 연구로 분류할 수 있다.

객체지향 설계 품질을 측정하는 연구는 Li et al.[10]의 연구와 Bansiya et al.[11] 연구가 있다. Li et al.[10]은 객체지향 소프트웨어 설계품질을 측정하기 위해 기존에 연구된 여러 메트릭을 분석하고 새로운 메트릭들을 추가하여 두 개의 상용 프로그램으로부터 효용성을 검증한다. Bansiya et al.[11]은 객체지향 설계에서 여러 가지 품질 속성을 측정하기 위한 계층 모델을 제안한다. 계층 모델은 품질 속성(quality attribute) 계층, 객체지향 설계 특성(design property) 계층, 객체지향 메트릭 계층, 객체지향 컴포넌트 계층으로 구성된다. 그리고 품질 속성과 설계 특성, 메트릭들을 정의하여 여러 가지 메트릭들의 측정값을 합산함으로써 품질 속성을 측정하는 방법을 제시한다. 그러나 측정값을 종합하여 평가하는 방법을 제시하고 있지 않기 때문에 소프트웨어의 객체지향 설계 품질이 어느 수준을 지니고 있는지 평가하기 어렵다.

소프트웨어 모듈화 수준을 측정하는 연구는 Sarkar et al.[8,9]의 연구와 Bangare et al.[4]의 연구 등이 있다. Sarkar et al.[8,9]은 모듈화 수준을 높이기 위한 여덟 가지 모듈 설계 원칙을 제안하고 각 설계 원칙을 정량적으로 측정할 수 있는 메트릭을 정의한다. 그리고 각 메트릭들을 실험을 통해 효용성을 검증한다. Bangare et al.[4]은 소프트웨어 유지보수성을 높이고 개발자들의 부담을 줄이기 위한 목적으로, 객체지향 모듈화 수준 측정 방법을 세 단계로 제안한다. 이를 수행하기 위해 네 가지 메트릭을 정의하고 각 메트릭을 측정하기 위한 알고리즘을 제안한다. 그리고 각 메트릭들을 실험을 통해 효용성을 검증한다. 두 연구는 제안한 메트릭을 종합하여 모듈화 수준을 평가하는 방법을 제시하고 있지 않기 때문에 제안한 메트릭 측정값을 통해 모듈화 수준을 주관적으로 판단해야 된다는 단점을 지닌다.

ISO/IEC 9126, 25010 표준에서 정의한 소프트웨어 품질 속성을 측정하는 연구는 상당히 많이 수행되었다. Washizaki et al.[6]은 컴포넌트 기반 개발(Component-based Development)에서 소프트웨어 컴포넌트의 재사용성(reusability)을 측정하기 위한 메트릭들을 제안한다. 컴포넌트는 경우에 따라 목적 코드(object code)만 사용자에게 공개되고 소스코드는 숨겨지는 경우가 많기 때문에 그들은 소스코드를 분석하지 않고 컴포넌트의 재사용성을 측정하기 위한 모델을 [그림 1]과 같이 제안한다. 이들은 재사용성을 세 가지 품질 요소(quality factor)와 네 가지 기준으로 나누어 각각에 대한 메트릭을 정의한다. Subramanian et al.[12]은 ISO/IEC 25010 표준에서 정의한 소프트웨어 적응성(adaptability)을 측정하기 위한 메트릭을 제안한다. 적응성을 측정하기 위해 컴포넌트와 커넥터가 각각 다른 컴포넌트와 커넥터에 적응 가능한 상태인지 판별하기 위한 아키텍처 수준의 적응성 메트릭을 정의하고 이를 이용하여 소프트웨어 수준의 적응성 메트릭을 정의한다. 이와 달리, Liu et al.[13]은 이해관계자들에 의해 발생되는 변경 요구사항에 대해 얼마나 영향을 많이 받는지 정량적으로 측정하여 소프트웨어 아키텍처의 적응성을 측정한다. 변경 요구사항에 대한 영향을 측정하기 위해서 발생 가능한 여러 가지 시나리오를 정의하고 각 시나리오에 대해 컴포넌트와 커넥터의 변경 정도를 측정한다. 세 연구에서 정의한 각 메트릭은 소프트웨어 품질에 영향을 주는 정도가 다를 수 있기 때문에 각 메트릭의 가중치를 결정할 필요가 있다. 그러나 이를 고려하지 않고 있다는 점에서 한계를 지닌다.



[그림 1] 컴포넌트 재사용성의 품질 속성 모델[6]

3. 모듈성을 정량적으로 측정하는 방법

본 절에서는 본 연구에서 다루는 모듈과 모듈성을 3.1절에서 정의하고 모듈성 품질 모델을 3.2절에서 제시한다. 이를 활용하여 3.3절에서는 모듈성을 정량적으로 측정하는 방법을 소개한다. 그리고 모듈성이 올바로 측정되었는지 검증하는 방법을 3.4절에서 소개한다.

3.1 모듈과 모듈성의 정의

기존의 많은 연구에서 모듈의 특성에 대해 언급하고 있다. Clements 등은 모듈을 관련된 행위, 지식, 의사결정 등을 제공하는 소프트웨어 구현 단위[14]라고 표현한다. Briand 등과 Sarkar등은 모듈은 시스템 내의 일부 코드 조각, 자료구조, 또는 부분 시스템 등으로 정의될 수 있다고 말하며[8,15], Rozanski등은 시스템을 개발하기 위해 고려

될 수 있는 기본 단위라고 말한다[16]. Hofmeister 등은 모듈의 내부 데이터와 함수들은 캡슐화 되어야 하며, 외부에 서비스를 제공하기 위해서 인터페이스를 정의해야 한다고 말한다[17]. 이들의 주장을 바탕으로 모듈을 다음과 같이 정의할 수 있다.

모듈의 정의: 기능을 제공하기 위한 관련된 함수와 클래스들의 집합으로 캡슐화된 소프트웨어 구현 단위

모듈의 정의에 따라, 모듈은 함수와 클래스들이 모여 이뤄진 파일 또는 디렉토리 단위로도 구성될 수 있다. 모듈이 가지는 관련된 여러 기능들은 함수와 클래스의 집합으로 구현될 수 있는데, 이들은 캡슐화되어 인터페이스를 통해서만 외부로부터 접근 가능해야 한다.

새로운 소프트웨어 개발할 때, 기존 모듈들을 활용하기 위해서 기존 모듈들이 모듈로서 자격을 갖추고 있는지 평가할 필요가 있다. 그 이유는 좋은 모듈을 재사용하면 그만큼 좋은 소프트웨어를 산출할 수 있기 때문이다. 즉, 잘 구성된 모듈을 재사용하면 소프트웨어 개발 비용을 줄일 수 있고 그 소프트웨어는 유지보수가 용이하며 결함 발생률이 낮다. 본 논문에서는 모듈이 얼마나 잘 구성되었는지 품질을 나타내는 척도로서 모듈성을 다음과 같이 정의한다.

모듈성(moduleness)의 정의: 모듈외부로부터 정보를 숨기고 다른 모듈과 독립적으로 기능을 수행함으로써 모듈의 기능을 여러 시스템에서 재사용하기 용이한 정도

3.2 모듈성의 품질 모델

소프트웨어 모듈성을 정량적으로 측정하기 위해, 먼저 모듈성을 구성하는 품질 속성과 품질 측면을 정의할 필요가 있다. 많은 연구들에서 모듈과 관련하여 여러 가지 품질 속성들을 제시하고 있지만, 이들을 모두 고려하는 것은 많은 노력이 필요하고 복잡하기 때문에 사용하기 어렵다. 또한, 모듈성과 상대적으로 관련이 적은 속성들까지 모두 고려하기 보다 관련이 깊은 속성들을 신중히 고려하는 것이 효과적이다. 본 논문에서는 소프트웨어 모듈에 관해 기술하는 서적[14,16,17,18,19]과 논문[20,21]들을 조사하였다. 그 결과, 여러 연구에서 여러 번 언급되는 모듈의 특성들을 다음 다섯 가지로 정리할 수 있다.

특성 1. 모듈은 관련된 기능들을 제공한다[14,16,17].
특성 2. 모듈은 외부로부터 내부 정보를 노출시키지 않는다[17,18].
특성 3. 모듈은 재사용이 용이해야 한다[16,20].
특성 4. 모듈은 이해하기 쉬워야 한다[19,21].
특성 5. 모듈이 지닌 기능을 시스템에 맞게 잘 적응할 수 있는 능력을 가져야 한다[20].

위 다섯 가지 특성을 바탕으로 MOOD (Model for Object-Orient Design)와 ISO/ IEC9126, ISO/ IEC25010 표준에서 정의한 품질 속성들을 참고하여 모듈이 갖춰야 할 품질 속성들로서 캡슐화(encapsulation)(특성 1, 2에 의해), 맞춤성(customizability)(특성 3, 5에 의해), 이해용이성(understandability)(특성 3, 4에 의해), 그리고 코드의 품질 수준을 평가하기 위해 코드 품질(code quality)을 정의한다. 각 품질 속성의 정의는 <표 1>과 같다.

<표 1>에서 정의한 네 개의 품질 속성들을 측정하기 위해 코드 수준에서 측정할 수 있는 품질 측면들을 정의할 필요가 있다. 본 논문에서는 CK(Chidamber & Kemerer) metrics[22]와 QMOOD [11]에서 제시하는 품질 측면들을 참고하여 모듈성의 네 품질 속성들과의 연관성을 분석한다.

모듈의 캡슐화는 정의와 같이 내부 정보를 숨기는 정도를 나타내는 정보 은닉(information hiding)이 갖춰져야 한다.

모듈의 맞춤성을 높여 다른 시스템에 쉽게 적용하기 위해 다른 모듈과의 의존성을 제거해야 한다.

그 이유는 새로운 시스템에 적용하기 위한 해당 모듈의 수정이 다른 모듈의 수정을 요구하여 추가적인 노력이 들기 때문이다. 또한, 모듈의 맞춤성을 높이기 위해 다른 시스템에 재사용이 용이해야 한다. 객체지향의 주요 특성인 상속과 다형성은 시스템의 재사용성을 높여 준다[23]. 따라서 모듈에 상속과 다형성의 원리를 적용하여 새로운 시스템에 적용하기 유용하게 만들 수 있다. 결론적으로, 맞춤성을 결정하는 품질 측면으로 결합도, 상속성, 다형성을 선정한다.

<표 1> 모듈성을 구성하는 품질 속성

품질 속성	정의
캡슐화	공개가 불필요한 모듈의 데이터와 함수를 숨겨 외부에서 보이지 않게 하는 정도
맞춤성	모듈이 새로운 시스템에 적용되기 위해 수정하기 용이한 정도
이해 용이성	사용자가 모듈 또는 소스코드의 구성과 의미를 파악하기 용이한 정도
코드 품질	코드가 잠재적 결함 발생률을 낮추고 유지보수를 용이하게 해줄 수 있는 정도

모듈의 기능을 쉽게 이해되도록 구성하기 위해서는 모듈을 구성하는 함수의 크기가 너무 크거나 내부 구현이 복잡하지 않아야 한다. 또한, 한 모듈의 기능과 관련된 데이터와 그 구현들끼리 잘 집적되어야 해당 모듈의 기능을 빠르게 이해하는데 도움이 된다. 그리고 모듈의 상속 깊이(inheritance depth)가 커서 추상화 수준이 높으면 해당 모듈을 이해하기 위해 여러 부모 클래스를 모두 분석해야 하므로 이해하기 어려워진다. 따라서 이해 용이성을 결정하는 품질 측면으로 평균 함수(메소드) 라인 수, 복잡도, 응집도, 추상성을 선정한다.

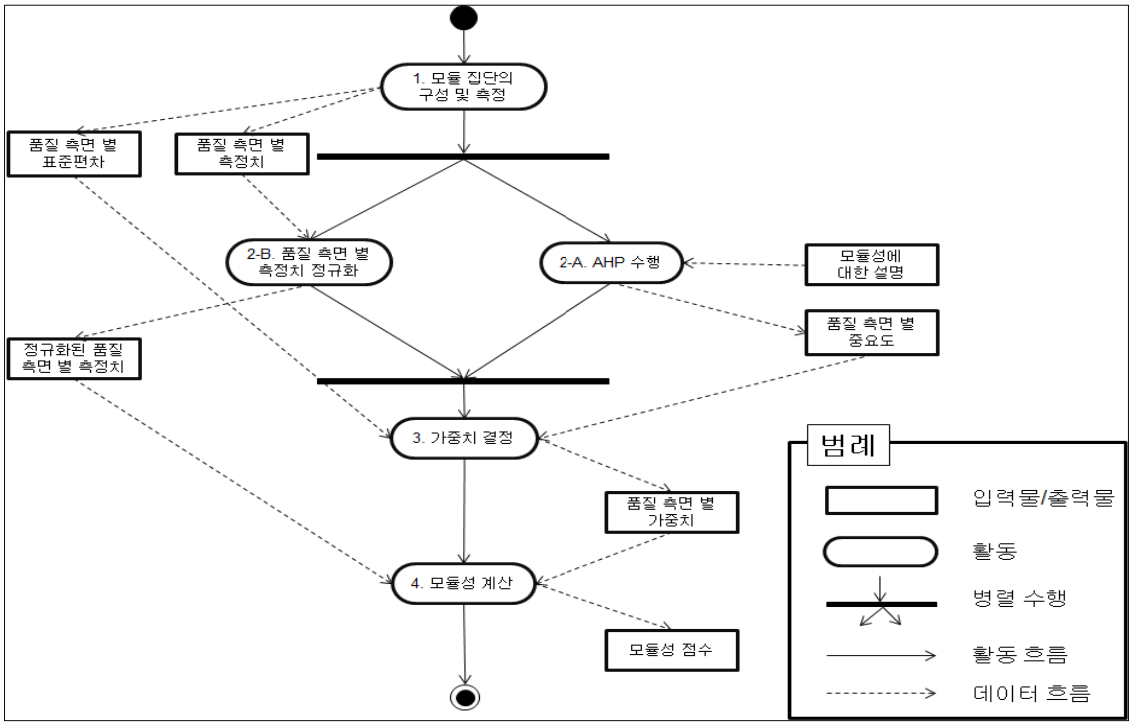
모듈을 구성하는 각 클래스(파일)의 라인 수가 많으면 그만큼 많은 기능들이 한 클래스에 집적되어 유지보수성, 재사용성 등 모듈 전체의 코드 품질을 저하시킨다. 또한, 모듈 내에 동일한 기능을 수행하는 코드가 중복으로 구현되어 있으면 코드 수정 시, 잇따라 수정되어야 할 코드의 미수정으로 인한 결함 발생률을 높인다. 따라서 코드 품질을 결정하는 품질 측면으로 평균 클래스(파일) 라인 수와 중복코드 비율을 선정한다.

<표 2>는 각 품질 속성을 구성하는 품질 측면과 측정 지표를 나타낸다. 품질 측면은 해당 속성에 대해 긍정적인 상관관계를 가질 수도 있고 (+로 표시)부정적인 상관관계를 가질 수도 있다(-로 표시).

모듈은 기본적으로 C++과 Java와 같은 객체지향 언어에서는 클래스가 될 수 있고 C와 같은 절차지향 언어에서는 파일이 될 수 있다. 클래스의 개념이 없는 C언어에서는 상속성, 다형성, 추상성의 측정결과가 0이기 때문에 C++, Java에 비해 모듈성이 떨어진다. 이는 C언어가 지향하는 바는 객체지향과 다르며, 모듈성과 거리가 멀기 때문이다.

<표 2> 모듈성의 구성 품질 속성, 품질 측면 및 측정 지표

품질 속성	품질 측면 (모듈성과의 상관관계)	측정 지표
캡슐화	정보 은닉(+)	$\frac{1-public\text{맴버변수수}}{\text{맴버변수수}}$
맞춤성	결합도(-)	외부 모듈과의 함수 참조 수
	상속성(+)	$\frac{\text{상속하는함수수}}{\text{접근가능한함수수}}$
	다형성(+)	가상함수의 개수
이해 용이성	추상성(-)	조상 클래스 수
	응집도(+)	CAMC(Cohesion Among Methods in a Class)[24]
	복잡도(-)	$\frac{\text{순환복잡도}}{\text{총함수수}}$
	평균 함수 (메소드) 라인 수(-)	$\frac{\text{총라인수}}{\text{총함수수}}$
코드 품질	평균 클래스 (파일) 라인 수(-)	$\frac{\text{총라인수}}{\text{클래스(파일)수}}$
	중복코드 비율(-)	$\frac{\text{중복코드라인수}}{\text{총라인수}}$



[그림 2] 모듈성 계산식 도출 절차

3.3 모듈성 계산식의 도출

모듈성을 측정하기 위해서 다음 두 가지 사항을 고려할 필요가 있다. 첫째, 각 품질 측면은 모듈성에 대해 중요한 정도가 다르다. 그 이유는 각 품질 측면이 모듈성에 미치는 영향력이 다르기 때문이다. 따라서 모듈성을 산출할때, 각 품질 측면은 중요도에 따라 다른 비중으로 합산될 필요가 있다. 이를 위해, 각 품질 측면의 가중치를 결정하는 활동이 필요하다. 둘째, 각 품질 측면 측정값을 합산하여 모듈성 점수를 계산하기 위해서는 각 품질 측면의 측정 단위가 동일해야 한다.

각 품질 측면의 가중치를 결정하기 위해서 AHP를 통해 전문가들의 의견을 수집한다. 이를 활용하여 결정된 가중치와 모듈의 품질 측면 측정값를 조합하여 모듈성을 측정할 수 있다. 각 품질 측면의 측정 단위 차이를 고려하기 위해서 통계적 방법을 적용한다.

[그림 2]는 모듈성의 품질 측면 별 가중치를 결정하여 모듈성을 측정하는 방법을 나타낸 활동도이다. 총 다섯 가지 활동으로 나뉘고 각 활동은 입력물 또는 산출물을 가진다. 본 소절에서는 각 활동에 대해 설명한다.

3.3.1 모듈 집단의 구성 및 측정

활동 1은 모듈성을 측정할 모듈들을 수집하고 각 모듈의 품질 측면들을 측정 지표를 이용하여 측정하는 활동이다. 품질 측면 별 측정값들은 모듈성을 산출하기 위해 사용된다.

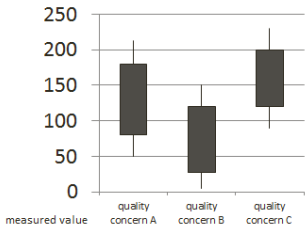
3.3.2 품질 측면 별 측정치 정규화

모듈성을 산출하기 위해 품질 측면 별 측정값들을 바로 사용할 수 없다. 그 이유는 같은 측정값 수치에 대해서 같은 수준을 나타내지 않기 때문이다. 예를 들어, [그림 3]은 여러 모듈에 대한 품질 측면 A, B, C의 측정값 분포를 나타낸다. 이들은

측정값 100에 대해, A와 C품질 측면은 낮은 수준에 해당하지만 B품질 측면은 높은 수준에 해당한다. 높은 수준에 해당하는 측정값에 높은 점수를 부여하고 낮은 수준에 해당하는 측정값에 낮은 점수를 부여하기 위해서는 측정값을 수준에 맞는 점수로 변환할 필요가 있다. 이를 위해, 활동 2-A에서 측정값의 정규화를 수행한다.

정규화는 식(1)을 사용하여 수행한다. 각 품질 측면 측정값(X_i)의 점수(X_i')는 평균($\overline{X}$)을 감산하여 수준을 동일하게 조정할 수 있다. 예를 들어, A 품질 측면 측정값이 1, 2, 3이고 B 품질 측면 측정값이 1001, 1002, 1003이라고 할 때, B 품질 측면 측정값이 A품질 측면 측정값보다 큰 경향을 띈다. 이를 식(1)을 통해 정규화하여 1과 1001에는 -1점, 2와 1002에는 0점, 3과 1003에는 1점을 부여할 수 있다.

(1) $X_i' = X_i - \overline{X}$



[그림 3] 정규화되지 않은 측정값의 분포

3.3.3 AHP 수행

모듈성을 구성하는 각 품질 측면 별 중요도를 결정하기 위해, 여러 전문가들이 AHP를 통해 쌍대 비교를 수행하여 모듈성에 대해 어떤 품질 측면이 얼마나 더 중요한지 판단한다. AHP를 수행하는 전문가들은 먼저 모듈성에 대한 기반 지식을 숙지해야 한다. 그 이유는 모듈성의 각 품질 측면들이 선정된 이유와 측정 방법을 이해해야만 각 품질 측면이 모듈성을 결정하는데 얼마나 중요한지 명확히 판단할 수 있기 때문이다.

각 품질 측면이 $C_1 \cdots C_n$이고 C_i가 C_j보다 중요한 정도를 V_{ij}라고 할 때, 전문가들은 <표 3>을 참고하여 <표 4>를 작성할 수 있다.

<표 3> 상대적 중요도에 대한 값

품질 측면 C_i에 대한 품질 측면 C_j의 상대적 중요도	값
매우 더 중요	5
더 중요	3
동일	1
덜 중요	1/3
매우 덜 중요	1/5

<표 4> 쌍대 비교표

i \ j	C_1	C_2	…	C_n
C_1	1	V_{12}	…	V_{1n}
C_2	1/ V_{12}	1	…	V_{2n}
…	…	…	…	...
C_n	1/ V_{1n}	1/ V_{2n}	…	1

쌍대 비교는 전문가의 주관적 판단에 따라 수행되므로 <표 4>가 일관성 있게 작성되었는지 평가하기 위해 일관성 검사를 수행한다. 일관성은 일관성 지수(Consistency Index: CI)와 일관성 비율(Consistency Ratio: CR)을 계산하여 평가할 수 있다. 일관성 지수는 쌍대 비교의 대상이 되는 품질 측면의 개수를 n, 최대 고유벡터(eigenvector)를 $\lambda_{\max}$라 할 때, 식(2)를 사용하여 계산할 수 있다.

(2) $CI = \frac{\lambda_{\max} - n}{n-1}$

일관성 비율은 일관성 지수와 확률 지수(Random Index: RI)를 사용하여 식(3)을 통해 계산할 수 있다. 일반적으로 일관성 비율이 0.1이하일 경우, 논리적으로 일관성을 유지한 것으로 판단한다[25]. 만일, 전문가가 평가한 쌍대 비교표의 일관성 비율이 0.1을 초과한다면 해당 쌍대 비교표는 일관적으로 판단되지 않은 것이므로 쌍대 비교를 재수행할 필요가 있다.

(3) $CR = \frac{CI}{RI}$

일관성 검사를 통해 일관성이 검증된 전문가들의 평가자료는 기하평균법(geometric average)을 사용하여 종합할 수 있다. 기하평균법은 상대적인 비율값들의 평균치를 계산할 때 사용되는데, 쌍대비교는 두 품질 측면간의 상대적인 중요도 비율을 평가하는 방법이기 때문에 기하평균법을 사용하는 것이 적합하다.

종합된 쌍대 비교표에 식(4)를 이용하여 각 품질 측면(X_i)의 기하평균(G_i)을 계산하고 식(5)을 사용하여 각 품질 측면의 중요도(AHP_{X_i})를 산출한다.

(4) $G_i = \left(\sum_{j=1}^{n} V_{ij} \right)^{\frac{1}{n}}$

(5) $AHP_{X_i} = \frac{G_i}{\sum_{k=1}^{n} G_k}$

3.3.4 가중치 결정

각 품질 측면의 가중치는 앞서 결정한 품질 측면 별 중요도를 이용할 수 있다. 그러나 각 품질 측면은 측정 단위가 다르기 때문에 동일한 수치에 대한 크기가 다르다. 예를 들어, 코드 라인 수에서 1이라는 수치는 크기가 작지만 중복코드 비율에서 1이라는 수치는 굉장히 큰 수치이다. 따라서 단위 차이 문제를 해결하기 위해 모듈 집단의 표준편차를 활용한다. 표준편차는 표본의 측정값이 평균으로부터 얼마나 퍼져있는지 나타낸다. 이는 단위 차이에 따라 큰 영향을 받는다. 이를 해결하기 위해, 각 수치들을 표준편차로 나누어 측정 단위의 영향을 배제시킬 수 있다. 예를 들어, 다음과 같이 집합 A, B, C가 있을 때,

A(cm)={0, 2, 2, 5, 7, 7, 7, 10, 10, 10}
B(mm)={0, 20, 20, 50, 70, 70, 70, 100, 100, 100}
C(kg)={0, 0, 5, 5, 5, 5, 10, 10, 10, 10}

이들을 구성하는 측정값들은 단위가 다르기 때문에 서로 비교할 수 없다. 물론, 집합 A와 B는 실질적으로 크기가 같은 측정값들로 구성되지만 수치상으로는 다르다. 이들을 표준편차로 나누면 다음과 같다.

A′={0, $\frac{2}{\sqrt{12}}, \frac{2}{\sqrt{12}}, \frac{5}{\sqrt{12}}, \frac{7}{\sqrt{12}}, \frac{7}{\sqrt{12}}, \frac{7}{\sqrt{12}}, \frac{10}{\sqrt{12}}, \frac{10}{\sqrt{12}}, \frac{10}{\sqrt{12}}$}
B′={0, $\frac{2}{\sqrt{12}}, \frac{2}{\sqrt{12}}, \frac{5}{\sqrt{12}}, \frac{7}{\sqrt{12}}, \frac{7}{\sqrt{12}}, \frac{7}{\sqrt{12}}, \frac{10}{\sqrt{12}}, \frac{10}{\sqrt{12}}, \frac{10}{\sqrt{12}}$}
C′={0, 0, $\frac{5}{\sqrt{12}}, \frac{5}{\sqrt{12}}, \frac{5}{\sqrt{12}}, \frac{5}{\sqrt{12}}, \frac{10}{\sqrt{12}}, \frac{10}{\sqrt{12}}, \frac{10}{\sqrt{12}}, \frac{10}{\sqrt{12}}$}

집합 A′, B′, C′는 측정 단위에 의한 측정값의 차이를 고려한 결과이다. 따라서 A′와 B′는 같은 측정값들로 구성되어 있다고 볼 수 있다. 마찬가지로, B′와 C′ 또한 단위 차이가 고려되어 수치상으로 서로 비교할 수 있다.

결과적으로, 각 품질 측면의 모듈성에 대한 중요도와 측정단위 차이를 고려한 가중치는 식(6)과 같다.

(6) $\omega_i = \frac{AHP_{X_i}}{\sigma_{X_i}}$

3.3.5 모듈성 계산

여러 가지 품질 측면들을 기준으로 모듈의 모듈성을 결정하는 문제는 다기준 의사결정(Multi-Criteria Decision Making) 문제이다. 다기준 의사결정 문제를 해결하기 위한 간단한 방법 중 하나는 가중합 모델(Weighted-Sum Model)을 이용하는 방법이다. 그 이유는 각 기준들의 점수와 가중치를 단순히 곱의 합으로 쉽게 계산하여 의사결정을 수행할 수 있기 때문이다. n 개의 품질 측면 별 가중치(ω_i)와 측정값(X_i)을 이용하여 모듈성 점수(M)를 산출하기 위한 가중합 모델은 식(7)과 같다.

(7) $M = \sum_{i=1}^{n} \omega_i X_i$

가중합 모델에서 앞서 유도한 가중치와 정규화된 측정값을 적용하기 위해 식(7)에 식(1)과 식(6)을 대입하면 식(8)과 같다. 식(8)을 이용하여 각 품질 측면의 중요도와 측정 단위 차이가 고려된 모듈성을 산출할 수 있다.

(8) $M = \sum_{i=1}^{n} \frac{AHP_{X_i}}{\sigma_{X_i}} \cdot (X_i - \overline{X})$

3.4 검증 방법

모듈성 산출 결과가 모듈의 품질을 잘 반영하는지 검증하기 위해, 제안 방법으로 산출한 모듈성과 전문가들이 직접 판단한 모듈성의 일치도를 확인할 수 있다. 그러나 전문가들이 수집된 모든 모듈의 내부 코드를 분석하여 모듈성을 평가하는 것은 상당히 복잡하고 어렵다. 따라서 수집된 n개의 모듈 중 m($\leq$ n)개를 임의로 선정하고 제안 방법으로 결정한 m개 모듈의 순위와 전문가들이 직접 판단한 m개 모듈의 순위를 비교한다. 두 방법의 순위가 얼마나 밀접한 관계를 가지는지 측정하기 위해 스피어만 순위상관계수(Spearman's rank-order correlation coefficient)(ρ)를 사용할 수 있다. 스피어만 순위상관계수는 두 변수 사이의 상관된 정도를 변수값의 순위를 이용하여 분석하는 방법으로, 각 변수값에 대한 순위 차이(d_i)를 이용하여 식(9)을 통해 계산할 수 있다. 이는 -1과 1사이의 값을 가지는데, 두 변수의 순위가 완전히 일치하면 1, 두 변수가 완전히 반대이면 -1이 된다.

(9) $\rho = 1 - \frac{6\sum d_i^2}{n(n^2-1)}$

<표 5> 스피어만 상관계수 임계치

쌍 개수 \ 유의수준	10%	5%	1%
4	1.0000		
5	0.9000	1.0000	
6	0.8286	0.8857	1.0000
7	0.7143	0.7857	0.9286
8	0.6429	0.7381	0.8810
9	0.6000	0.7000	0.8333
10	0.5636	0.6485	0.7939
11	0.5364	0.6182	0.7545
12	0.5035	0.5874	0.7273
…	…	…	…

스피어만 순위상관계수를 이용하여 두 방법으로 결정한 순위가 서로 관련이 있다는 것을 보이기 위해 유의성 검정(significance test)[26]을 수행할 수 있다. 이를 통해, 산출된 순위상관계수가 '두 방법으로 결정한 순위가 관련이 없다'라는 귀무가설의 임계치를 벗어난다는 사실을 보임으로써 대립가설을 채택할 수 있다. 임계치는 쌍의 개수와 신뢰수준 혹은 유의수준에 따라 결정된다. 일반적으로 90%, 95%, 99%의 신뢰수준 혹은 10%, 5%, 1%의 유의수준을 사용하는 것이 보통이다. <표 5>는 쌍의 개수와 신뢰수준 별 임계치을 나타낸 표이다.

각 전문가에 대해, 두 방법의 순위상관계수가 임계치보다 높으면 귀무가설은 기각되고 두 방법으로 결정한 순위는 상관성이 있다고 판단한다.

4. 오픈소스를 이용한 사례 연구

본 절에서는 오픈소스 소프트웨어를 대상으로 제안 방법을 적용한 사례 연구를 소개한다. 먼저, 4.1절에서는 모듈성을 측정하기 위해 선정된 모듈을 제시하고 그들의 각 품질 측면을 측정한다. 그리고 4.2절에서는 품질 측면 별 가중치 결정을 위한 AHP를 수행 과정을 보이고, 4.2절에서는 각 모듈들의 품질 측면 측정값과 AHP 수행 결과를 이용하여 각 모듈의 모듈성을 측정한다. 마지막으로, 4.4절에서는 제안 방법을 통한 모듈성이 올바로 측정되는지 검증한다.

<표 6> 선정된 오픈소스 소프트웨어

언어	오픈소스	선정 모듈 개수
C	bash, gcc, php, gimp, chrome, linux	33
C++	Filezilla, inkscape, webkit, gwenview, libxml++, 7zip	33
Java	Tomcat, hibernate, spring, junit, Log4j	34
합계	17개	100

<표 7> 종합된 전문가들의 AHP수행 결과

	정보 은닉	결합도	상속성	다형성	추상성	응집도	복잡도	평균 함수 라인 수	평균 클래스 라인 수	중복코드 비율
정보 은닉	1.00	0.47	1.34	1.41	1.20	0.54	1.05	1.90	1.72	1.19
결합도	2.12	1.00	3.30	3.13	2.70	1.25	1.84	3.91	3.11	2.42
상속성	0.75	0.30	1.00	1.16	0.93	0.51	0.90	1.54	1.44	0.79
다형성	0.71	0.32	0.86	1.00	0.84	0.36	0.74	1.57	1.48	0.85
추상성	0.84	0.37	1.07	1.20	1.00	0.42	0.72	1.46	1.24	0.76
응집도	1.86	0.80	1.95	2.81	2.39	1.00	1.63	2.50	2.50	2.12
복잡도	0.95	0.54	1.11	1.35	1.38	0.61	1.00	2.12	2.04	1.56
평균 함수 라인 수	0.53	0.26	0.65	0.64	0.68	0.40	0.47	1.00	1.07	0.68
평균 클래스 라인 수	0.58	0.32	0.70	0.68	0.81	0.40	0.49	0.93	1.00	0.85
중복코드 비율	0.84	0.41	1.27	1.17	1.32	0.47	0.64	1.46	1.17	1.00
중요도	**0.099**	**0.209**	**0.078**	**0.071**	**0.076**	**0.167**	**0.106**	**0.054**	**0.058**	**0.082**

<표 8> 오픈소스 모듈의 모듈성 점수와 순위

모듈명	모듈성 점수	순위	모듈명	모듈성 점수	순위	모듈명	모듈성 점수	순위
FileHandler	0.343	24	CControlSocket	-0.210	75	readline	-0.848	95
AsyncFileHandler	0.140	46	CBookmarksDialog	-0.670	92	relocatable	-0.136	68
MethodExpressionImpl	0.321	27	CFileItem	-0.007	58	node	-0.202	72
ELArithmetic	-0.091	64	CFileZillaApp	0.004	56	policy	-0.577	89
ELParser	-1.439	97	CFilterManager	-0.327	82	trusted	-0.460	87
Token	0.351	22	COptionsPage	0.394	18	internal	-0.203	73
Compiler	-0.271	79	CProxySocket	-0.807	94	symtab	0.081	51
Generator	-0.863	96	CEncoder	0.073	52	mkdeps	0.083	50
ParserController	-0.145	69	CThreadInfo	-0.206	74	runtime	-0.243	77
…	…	…	…	…	…	…	…	…

<표 9> 제안 방법을 통한 모듈 순위와 전문가가 평가한 모듈 순위간의 일치성 검사 결과

모듈명	평가 순위											
	제안 방법	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	평균
stats	4	6	2	5	2	1	1	1	2	2	4	1
trusted	11	9	11	12	11	7	8	11	7	10	10	10.5
history	8	8	8	8	9	12	11	10	5	8	7	8
gimp	12	12	10	11	12	9	9	12	12	12	11	12
CFilterManager	10	10	9	10	10	8	7	4	11	9	9	9
CData	1	1	1	1	4	11	10	8	4	1	2	3
COptionsPage	2	2	4	2	1	3	3	3	1	3	5	2
CEncoder	7	5	5	3	7	6	4	6	9	5	6	6
ServletServerHttpRequest	5	7	6	4	3	4	5	2	3	6	8	5
Event	3	4	3	6	5	2	2	9	8	4	1	4
TestCaseTest	6	3	7	7	6	5	6	7	6	7	3	7
CControlSocket	9	11	12	9	8	10	8	5	10	11	12	10.5
스피어만 상관계수(ρ)		0.90	0.90	0.90	0.92	0.44	0.52	0.43	0.74	0.94	0.84	0.93
유의성 검정(ρ>0.51)		✓	✓	✓	✓	X	✓	X	✓	✓	✓	✓

4.1 오픈소스 모듈의 선정 및 측정

오픈소스는 마이크로소프트에서 설립한 Ohloh 사에서 제공하는 오픈소스 목록을 사용자 수로 내림차순으로 정렬하여 <표 6>과 같이 선정하였고 이들로부터 100개의 모듈을 임의로 선정하였다. 모듈은 정의에 따라, 관련된 기능들의 집합으로 구성되어야 한다. 따라서 C언어로 구현된 오픈소스에서는 파일 단위로 선정하고 C++과 Java로 구현된 오픈소스에서는 클래스 단위로 선정하였다. 선정된 100개의 모듈들은 <표 2>의 측정 지표를 사용하여 측정되었다.

4.2 AHP 수행

모듈성의 품질 측면 별 중요도를 결정하기 위해 10명의 전문가들을 대상으로 AHP를 수행하였다. 전문가는 소프트웨어 공학을 전공하고 있는 석사과정 학생 3명과 박사과정 학생 3명, 현업 실무자 4명으로 구성되고, 각 전문가는 5년 이상의 개발 경력 또는 소프트웨어 공학 연구 경력을 가진다. 그리고 본 연구를 함께 진행하면서 모듈성에 관한 기반 지식을 상세히 숙지하였다.

각 전문가가 작성한 쌍대 비교표는 모두 일관성 비율이 0.1이하로 측정되었다. 따라서 10개의 쌍대 비교표는 논리적으로 일관성을 유지한다고 판단할 수 있다. <표 7>은 10명의 전문가들의 AHP수행 결과를 종합하고 식(4)와 식(5)를 사용하여 품질 측면 별 중요도를 산출한 결과이다. 여러 가지 품질 측면들 중, 모듈성을 결정하는데 결합도, 응집도, 복잡도 순으로 중요하다는 사실을 알 수 있다.

4.3 모듈성 계산

모듈의 품질 측면 측정값들과 품질 측면 별 중요도를 이용하여 식(8)을 통해 모듈성을 계산할 수 있다. <표 8>은 선정된 모듈의 모듈성 산출 결과와 순위를 나타낸다. 이를 활용하여, 선정된 모듈 내에서 각 모듈의 모듈성 수준을 판단 할 수 있다. 예를 들어, <표 8>의 ELParser의 모듈성 순위는 97위이다. 모듈성 순위가 낮게 산출된 이유는 모듈을 이루는 함수들간의 참조 수가 약 900개로 매우 많기 때문에 수정하기 어렵고, 약 2,000줄의 매우 큰 모듈이면서 그 중 68%의 중복 코드를 지니기 때문에 잠재적 결함 발생률이 매우 높다. 따라서 ELParser는 모듈성 수준이 낮게 평가된다.

측정 결과를 분석한 결과, C로 구현된 모듈보다 C++과 Java로 구현된 모듈의 모듈성 수준이 더 높게 나타나는 경향을 보였다. 이는 객체지향 언어와는 달리, C언어의 지향점이 모듈성과 거리가 있기 때문이다. C언어와 같은 절차지향 언어에 클래스의 개념을 새롭게 추가하여 재사용성과 확장성 등을 높인 언어가 객체지향 언어이다. 따라서 객체지향 언어는 기본적인 모듈성을 갖춘 언어라고 말할 수 있다.

4.4 검증

제안 방법으로 모듈성을 계산한 결과가 모듈의 품질을 잘 반영하는지 검증하기 위해, 본 논문에서는 12개의 모듈을 대상으로 제안 방법을 통해 결정한 순위가 전문가들이 직접 결정한 순위와 높은 일치성을 지닌다는 것을 보인다. 일치성을 평가하기 위해 100개 모듈에서 임의로 선정한 12개 모듈을 대상으로 식(9)를 사용하여 스피어만 순위상관계수를 계산하고 유의성 검정을 수행하였다(<표 9>). 일치성 검사를 위해서 12개의 표본에 대한 5%의 유의수준으로 단측검정(one-tailed test)을 수행하였다. 그리고 임계치(0.5035) 이상의 상관계수를 지니는 전문가를 '√'로 표시하였고, 그렇지 않은 전문가를 'X'로 표시 하였다. 총 10명의 전문가 중, 8명의 전문가가 임계치이상의 상관계수를 지닌다는 것을 알 수 있다. 이는 제안 방법으로 결정한 모듈성이 모듈의 품질을 상당히 잘 반영하고 있다는 것을 의미한다.

5. 결론

본 논문은 소프트웨어 모듈성을 정량적으로 측정하는 방법을 제안하였다. 먼저, 모듈과 모듈성을 정의하고 모듈성을 구성하는 품질 속성과 품질 측면을 정의하였다. 그리고 모듈성을 산출하기 위한 계산식을 도출하고 이를 검증하는 방법을 제시하였다. 사례 연구에서는 오픈소스 소프트웨어 모듈들의 모듈성을 산출하였고, 유의성 검정을 통해 제안 방법의 효용성을 검증하였다.

본 논문의 제안 방법이 앞으로 현업에서 모듈의 활용 방향을 결정하는데 도움을 줄 수 있을 것이다. 또한, 제안 방법의 도구화를 통해 모듈성 평가자들의 노력을 줄여줄 수 있다.

향후 연구로는 각 모듈의 모듈성 수준을 효율적으로 높이기 위해 먼저 개선해야 할 품질 측면들을 선정하는 연구와 각 품질 측면을 개선하기 위해 사용될 수 있는 리펙토링 기법을 제시하는 연구를 수행할 예정이다.

참 고 문 헌

[1] Rosqvist, T., Koskela, M., and Harju, H. “Software Quality Evaluation Based on Expert Judgement.” Software Quality Journal, Vol. 11, No. 1, pp. 39-55, 2003.

[2] Bass, L., Clements, P., and Kazman, R. Software Architecture in Practice, 3rd Edition, Addison-Wesley, 63 page, 2012.

[3] Barbacci, M., Klein, M. H., Longstaff, T. A., and Weinstock, C. B. Quality Attributes. Technical Report, CMU/SEI-95-TR-021, December 1995.

[4] Bangare, S. L., Khare, A. R., and Bangare, P. S. “Measuring the Quality of Object Oriented Software Modularization.” International Journal on Computer Science and Engineering (IJCSE), Vol. 3, No. 1, pp. 445-450, 2011.

[5] Rosqvist, T., Koskela, M., and Harju, H. “Software Quality Evaluation Based on Expert Judgement.” Software Quality Journal, Vol. 11, No. 1, pp. 39-55, 2003.

[6] Sarkar, S., Kak, A. C., and Nagaraja, N. S. “Metrics for analyzing module interactions in large software systems.” Proceedings of the 12th Asia-Pacific Software Engineering Conference (APSEC), pp. 264-271, 2005.

[7] Washizaki, H., Yamamoto, H., & Fukazawa, Y. “A metrics suite for measuring reusability of software components.” Proceedings of the Ninth International Software Metrics Symposium (METRICS), pp. 211-223, 2003.

[8] Sant’Anna, C., Figueiredo, E., Garcia, A., & Lucena, C. J. “On the modularity of software architectures: A concern-driven measurement framework.” Proceedings of the European Conference on Software Architecture (ECSA), Madrid, pp. 207-224, 2007.

[9] Sarkar, S., Rama, G. M., and Kak, A. C. “API-based and information-theoretic metrics for measuring the quality of software modularization.” IEEE Transactions on Software Engineering, Vol. 33, No. 1, pp. 14-32, 2007.

[10] Sarkar, S., Kak, A. C., and Rama, G. M. "Metrics for measuring the quality of modularization of large-scale object-oriented software." IEEE Transactions on Software Engineering, Vol. 34, No. 5, pp. 700-720, 2008.

[11] Li, W., Henry, S. "Object-oriented metrics that predict maintainability." Journal of systems and software, Vol. 23, No. 2, pp. 111-122, 1993.

[12] Bansiya, J., Davis, C. G. "A hierarchical model for object-oriented design quality assessment." IEEE Transactions on Software Engineering, Vol. 28, No. 1, pp. 4-17, 2002.

[13] Subramanian, N., Chung, L. "Metrics for software adaptability." Proceedings of International Concerence on Software Quality Management (SQM), pp. 95-108, 2001.

[14] Liu, X., Wang, Q. "Study on application of a quantitative evaluation approach for software architecture adaptability." Proceedings of the Fifth International Conference on Quality Software (QSIC). pp. 265-272, 2005.

[15] Clements, P., Garlan, D., Bass, L., Stafford, J., Nord, R., Ivers, J., and Little, R. Documenting software architectures: views and beyond, 2nd Edition, Pearson Education, 56 page, 2010.

[15] Briand, L. C., Morasca, S., and Basili, V. R. "Property-based software engineering measurement." Transactions on Software Engineering, Vol. 22, No. 1, pp. 68-86, 1996.

[16] Rozanski, N., Woods, E. Software systems architecture: working with stakeholders using viewpoints and perspectives, Addison-Wesley, 21-22 pages, 2011.

[17] Hofmeister, C. Applied software architecture. Addison-Wesley. 106-110 pages, 2000.

[18] Bloch, J. Effective java, Addison-Wesley, 2008.

[19] Pressman, R. S., and Ince, D. Software engineering: a practitioner's approach, 7th Edition, McGraw-Hill, 400-401 pages, 1992.

[20] Luer, C. "Assessing Module Reusability." Proceedings of the First International Workshop on Assessment of Contemporary Modularization Techniques (ACoM'07), IEEE Computer Society, p. 7, 2007.

[21] Parnas, D. L. "On the criteria to be used in decomposing systems into modules." Communications of the ACM, Vol. 15, No. 12, pp. 1053-1058, 1972.

[22] Chidamber, S. R., & Kemerer, C. F. "A metrics suite for object oriented design", IEEE Transactions on Software Engineering, Vol. 20, No. 6, pp. 476-493, 1994.

[23] Briand, L. C., Wust, J., & Lounis, H. "Using coupling measurement for impact analysis in object-oriented systems". International Conference on Software Maintenance, pp. 475-482, 1999.

[24] Bansiya J., Etzkorn L., Davis C. and Li W. "A Class Cohesion Metric For Object-Oriented Designs", The Journal of Object-Oriented Programming, Vol. 11, No. 8, pp. 47-52, 1999.

[25] Saaty, R. W. "The analytic hierarchy process—what it is and how it is used." Mathematical Modelling, Vol. 9, No. 3, pp. 161-176, 1987.

[26] Zar, J. H. "Significance testing of the Spearman rank correlation coefficient". Journal of the American Statistical Association, Vol. 67, No. 339, pp. 578-580, 1972.

저자 소개

정 필 수

2012년 충남대학교 컴퓨터공학과 졸업(학사)
2012~현재 한국과학기술원 전산학과 석사과정
관심분야는 소프트웨어 아키텍처, 소프트웨어 테스팅

안 종 선

2008년 건국대학교 전자공학과 졸업(학사)
2011년 카이스트 우주탐사공학과 졸업(석사)
2012년~현재 카이스트 전산학과 박사과정
관심분야는 소프트웨어 품질, 소프트웨어 아키텍처

박 태 현

2010년 KAIST 전산학과 졸업(학사)
2012년 KAIST-CMU 소프트웨어공학 프로그램 졸업 (석사)
2012년~현재 KAIST 전산학과 박사과정
관심분야는 소프트웨어 아키텍처, 소프트웨어 제품라인 공학, 소프트웨어 테스팅

강 성 원

1982년 서울대학교 사회과학대학 졸업(학사)
1989년 University of Iowa 전산학 졸업(석사)
1992년 University of Iowa 전산학 졸업(박사)
1993년~2001년 한국통신(KT) 선임연구원
1995년~1996년 미국 국립표준기술연구소(NIST) 객원연구원
2001년~2005년 한국정보통신대학교 조교수
2003년~현재 Carnegie-Mellon University MSE프로그램 겸임교수
2005년~2009년 한국정보통신대학교 부교수
2009년~현재 KAIST 전산학과 부교수

은 나 래

2004년 이화여대 컴퓨터공학과 졸업(학사)
2006년 이화여대 컴퓨터공학과 졸업(석사)
2006년~현재 LG전자 SW역량강화센터

고 상 원

2009년 한양대 컴퓨터공학과 졸업(석사)
2009년~2012년 롯데정보통신
2012년~현재 LG전자 SW역량강화센터

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

2012~2013 소프트웨어공학소사이어티 논문지 편집위원회

소프트웨어공학소사이어티 논문지 제26권 제3호 (통권 99호)

발 행 일 ॥ 2013년 9월 30일
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)