제41권 제11호 2014년 11월

ISSN 2383-630X(Print)
ISSN 2383-6296(Online)

정보과학회논문지

Journal of KIISE

VOLUME 41, NUMBER 11, NOVEMBER 2014

시스템 및 이론

라스트 레벨 캐쉬 성능 향상을 위한 캐쉬 교체 기법 연구 ..... 두 콩 튜안, 손동오, 김종면, 김철홍 871

비휘발성 버퍼 캐시를 이용한 파일 시스템의 주기적인 flush 오버헤드 개선 ..... 이은지, 강효정, 고 건, 반효경 878

SHA-3 해시 함수 검정 프로그램과 16bit-UICC 용 SHA-3 구현 ..... 이희웅, 홍도원, 김현일, 서창호, 박기식 885

경량 동적 코드 변환 기법을 이용한 동적 인스트루멘테이션 기법 설계 및 구현 ..... 김지홍, 이동우, 김인혁, 엄영익 892

소프트웨어 및 응용

다시점 비디오 부호화를 위한 개선된 예측 구조와 움직임 추정 기법 ..... 윤효순, 김미영 900

기계학습을 이용한 중등 수준의 단문형 영어 작문 자동 채점 시스템 구현 ..... 이경호, 이공주 911

통계적 얼굴 모델을 이용한 부분적으로 가려진 얼굴 검출 ..... 서정인, 박혜영 921

시점 불변인 특징과 확률 그래프 모델을 이용한 인간 행위 인식 ..... 김혜숙, 김인철 927

소프트웨어 가시화를 통한 품질 개선 사례 연구 ..... 박보경, 권하은, 손현승, 김영수, 이상은, 김영철 935

데이타베이스

시계열 데이터 기반의 부분 노이즈 제거 윤곽선 이미지 매칭 ..... 김범수, 이상훈, 문양세 943

정보통신

무선 센서 네트워크에서의 분산 컴퓨팅 모델 ..... 박총명, 이충산, 조영태, 정인범 958

무선 센서 망을 이용한 연속개체 탐지에서 이동싱크의 에너지 효율적인 위치갱신 방안 ..... 김천용, 조현종, 김상대, 김상하 967

초고화질 스트리밍 서비스의 QoS를 향상시키기 위한 라우터 버퍼 기반의 혼잡 제어 기법 ..... 오준열, 윤두열, 정광수 974

순환형 데이터 블록 체이닝을 이용한 차량용 블랙박스의 영상 데이터 무결성 보장 기법 ..... 이 강, 김경미, 조용준 982

■ 2014년도 학생논문 경진대회 수상작

Multi-pass Sieve를 이용한 한국어 상호참조해결 ..... 박천음, 최경호, 이창기 992

한국정보과학회
KOREAN INSTITUTE OF INFORMATION SCIENTISTS AND ENGINEERS

ISSN 2383-630X(Print) / ISSN 2383-6296(Online)
Journal of KIISE, Vol. 41, No. 11, pp. 935-942, 2014. 11
http://dx.doi.org/10.5626/JOK.2014.41.11.935

소프트웨어 가시화를 통한 품질 개선 사례 연구

(A Case Study on Improving SW Quality through Software Visualization)

박 보 경 † (Bo Kyung Park)
권 하 은 † (Ha Eun Kwon)
손 현 승 † (Hyun Seung Son)
김 영 수 †† (Young Soo Kim)
이 상 은 †† (Sang-Eun Lee)
김 영 철 ††† (R. Young Chul Kim)

요 약 오늘날 소프트웨어는 규모가 크고 시장 출하 기간의 단축 상황에서도 고품질 이슈가 중요하다. 그리고 산업 현장에서는 빠른 개발을 위해 아직도 코드 중심 개발에 초점을 두고 있다. 따라서 1) 개발자의 나쁜 코드 개발 습관의 개선 측면 그리고 2) 소프트웨어 비설계화, 비문서화 및 코드 내부 구조 비가시화의 유지보수 측면 등을 해결해야 한다. 이에 코드 가시화의 필요성이 대두되고 있다. 본 논문에서는 객체지향 코드의 내부 구조 시각화 방법을 위해 Tool-Chain을 이용한 내부 구조 가시화 방법과 품질 개선 절차를 제안한다. 사례로써 NIPA의 SW Visualization 기법을 실제 객체 코드에 적용한다. 먼저 객체지향 코드의 모듈 단위를 클래스로 정의하고, 코드의 정량적 분석 및 가시화를 통해 코드의 복잡도(Code Complexity)를 줄이고자 하였다.

키워드: 코드 가시화, 역공학, 복잡도, 소프트웨어 품질

Abstract Today, it is very important issue to high quality of software issue on huge scale of code and time-to-market. In the industrial fields still developers focuses on Code based development. Therefore we try to consider two points of views 1) improving the general developer the bad development habit, and 2) maintenance without design, documentation and code visualization. To solve these problems, we need to make the code visualization of code. In this paper, we suggest how to visualize the inner structure of code, and also how to proceed improvement of quality with constructing the Tool-Chain for visualizing Java code's inner structure. For our practical case, we applied Object Code with NIPA's SW Visualization, and then reduced code complexity through quantitatively analyzing and visualizing code based on setting the basic module unit, the class of object oriented code.

Keywords: code visualization, reverse engineering, complexity, SW quality

· 이 논문은 본 연구는 미래창조과학부 및 한국산업기술평가관리원의 산업원천기술개발사업[10035708, 고신뢰 자율제어 SW를 위한 CPS(Cyber-Physical Systems) 핵심 기술 개발]과 2014년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(NRF-2013R1A1A2011601)
· 이 논문은 2014 한국컴퓨터종합학술대회에서 '객체지향 코드의 정적 분석을 위한 Tool-Chain화 사례 연구'의 제목으로 발표된 논문을 확장한 것임

† 학생회원 : 홍익대학교 컴퓨터정보통신공학과
park@selab.hongik.ac.kr
kwon@selab.hongik.ac.kr
son@selab.hongik.ac.kr
†† 비 회 원 : 정보통신산업진흥원 소프트웨어공학센터
ysgold@nipa.kr
selee@nipa.kr
††† 정 회 원 : 홍익대학교 컴퓨터정보통신공학과 교수(Hongik Univ.)
bob@selab.hongik.ac.kr
(Corresponding author임)

논문접수 : 2014년 8월 6일
(Received 6 August 2014)
심사완료 : 2014년 9월 15일
(Accepted 15 September 2014)

정보과학회논문지 제41권 제11호(2014. 11)

1. 서 론

최근의 소프트웨어는 대형화, 복잡화 및 다기능화 됨에 따라 소프트웨어의 체계적인 품질 관리 기술의 필요성이 증가하고 있다. 또한 IT 융·복합화로 인해 최종제품에서 소프트웨어의 역할과 기능의 중요성이 점차 확대되고 있다[1].

고품질의 소프트웨어를 위해서 다음과 같은 방법이 존재한다. 첫째는 SP, GS, TMMi, CMMi 등과 같은 인증을 통해 소프트웨어의 품질을 향상시킬 수 있다. 두 번째는 최신 소프트웨어 개발 방법론이나 프로세스를 통해서 고품질의 소프트웨어를 개발하는 것이다. 세 번째는 개발된 소프트웨어를 완벽한 테스트 프로세스와 테스팅을 통해 소프트웨어의 오류를 제거하여 품질을 향상시키는 방법이다. 하지만 이 방법들은 오류를 발견하는데 용이하지만, 아키텍처의 변경이나 내부 코드의 가시화는 어렵다. 본 논문은 White Box 기반의 고품질 소프트웨어를 위해, 기존 코드의 문제점을 발견하고 이를 변경/수정하는 방향에 초점을 둔다. 또한 소프트웨어 개발자들의 나쁜 습관(Bad Smell)을 개선하는데 중점을 두고 있다. 이를 위해 코드의 가시화를 통한 역공학(Reverse Engineering) 기법을 적용했다. 역공학은 개발자의 도움 없이 기존 시스템을 분석하여 소프트웨어 구조를 파악하고, 소프트웨어 컴포넌트(모듈) 간의 상호관계를 식별하기 때문에 더 높은 수준의 레벨로 표현 가능하다[2].

본 논문은 코드의 가시화(SW Visualization) 기법을 적용하여 객체지향 코드 내부 구조의 가시화 방법과 프로세스를 제시하고, 결합도(Coupling) 기반의 소프트웨어 품질 개선 절차를 제안한다. 이를 위해 기존의 오픈소스 도구들을 활용해서 하나의 도구 체인(Tool-Chain)을 구성한다. 또한 기존의 객체지향 코드(JAVA 기반의 Use Case Drawing Tool)를 소프트웨어 가시화에 적용하여 활용방안을 도출한다. 이 방법을 통해 기존 코드의 구조 확인이 가능한 시각화된 결과물을 도출할 수 있다. 이 결과물을 결합도 측정 모듈과 품질 지표를 정의하고, JAVA 언어의 코드 패턴에 맞는 리팩토링을 수행한다. 이를 기반으로 SW의 품질 개선 결과를 보인다.

본 논문의 구성은 다음과 같다. 2장에서는 관련연구로 역공학과 소프트웨어 가시화에 대해서 소개하고, 3장에서는 객체지향 기반의 정적 분석을 위한 Tool-Chain 방법과 리팩토링 방안을 설명한다. 4장에서는 적용사례를 기술한다. 5장에서는 결론 및 향후 연구에 대해서 언급한다.

2. 관련 연구

2.1 소프트웨어 가시화(SW Visualization)

좋은 소프트웨어(Good Software)를 위해, 소스 코드와 SW 개발 프로세스에 대한 관리가 요구된다. 하지만 이를 수행하기에는 인력과 비용이 부족하며, 또한 전문적인 방법이 필요하다. 소프트웨어 가시화는 소프트웨어의 품질관리와 유지보수 향상을 위한 기법으로서, 시각화와 문서화를 통해 소스코드와 개발 프로세스를 관리한다[1].

이런 가시화는 지표설정에 따른 명확한 목표를 수립하고, 시스템 기반의 개발 활동을 수반한다. 또한 시각화를 통한 지속적인 모니터링은 SW 개발 및 품질관리가 가능하다. 문서화는 개발 과정에서 추출할 수 있는 다양한 산출물들을 문서화함으로써 업무 부담을 최소화하고 활용성을 극대화 한다. 즉, 외부와의 의사소통을 통해 기업의 개발 노하우를 관리하고, 내부 인력간의 업무 이해도를 향상시킬 수 있다.

2.2 역공학(Reverse Engineering)

역공학은 기존 대상 시스템을 분석하여 시스템의 구성요소와 관계를 파악하고, 더 높은 추상 수준의 표현물을 복원하는 방법이다[3]. 이 기법을 통해 시각화를 수행함으로써, 소프트웨어 개발자의 도움 없이 소프트웨어를 분석하고 이해하여 기존 시스템을 점검할 수 있다. 역공학 기법을 활용함으로써, 추적성과 복잡성에 대한 대처가 가능하고, 새로운 시각화 산출물을 확보할 수 있다. 또한 잃어버린 정보에 대한 복구와 부작용을 감지하고, 이를 통한 재사용이 가능하다. 따라서 소프트웨어 가시화를 위해 역공학 기법이 필요하다. 예를 들어, 대상 시스템의 코드 산출물은 파서 및 구조 분석기를 통해 소스코드의 내부 구조를 분석한다. 분석된 결과는 정보 저장소에 저장한다. 저장된 정보를 이용하여 뷰 합성기를 통해 새로운 산출물(설계, 문서화 등)을 추출한다. 본 논문에서는 오픈소스 도구들을 활용하여 가시화와 역공학을 위한 도구를 구성하였다. 이 도구의 구성은 3장에서 자세히 언급한다.

3. Tool-Chain을 통한 SW 품질 향상 방안

그림 1은 Tool-Chain 프로세스를 통한 SW 품질 개선 방안이다. 이 방법은 객체지향 코드 정적 분석을 위한 Tool-Chain 프로세스와 SW 품질 개선 방안으로 구성된다. Tool-Chain 프로세스는 Parser(또는 Analyzer), Database, View Composer의 3가지 오픈소스 기반 도구로 구성한다. 각 단계의 결과물은 다음 단계의 입력으로 사용되며, 소스코드 분석, 정보 저장 및 구조 분석, 시각화의 서로 독립적인 기능을 수행한다. Tool-Chain 프로세스를 통해 추출된 결과물은 소프트웨어 품질 개선에 사용된다. 본 논문에서는 결합도(Coupling)와 응집도(Cohesion)를 기반으로 소프트웨어의 품질을 측정한다. 결합도와 응집도 측정 모듈과 품질 지표를 정의하

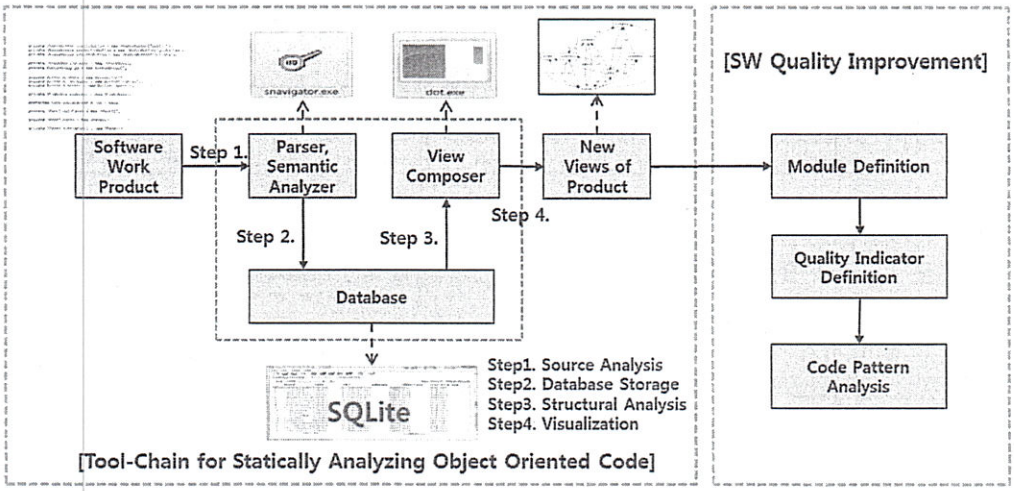


그림 1 Tool-Chain을 통한 SW 품질 개선 방안
Fig. 1 Improving SW Quality through SW Tool-Chain

고, JAVA에서의 코드 패턴을 결정하여, SW 품질 관리 방안을 제시한다.

3.1 객체지향 Code 정적분석을 위한 Tool-Chain

현재 상용화된 Open Source 도구들은 Tool-Chain으로 연계하여 정적 분석을 한다. 또한 객체지향 코드에서 시각화된 뷰(View)를 자동으로 생성한다. Tool-Chain 프로세스는 총 4가지 단계로 구성된다.

• Step 1: 소스 분석

소스 분석 단계에서는 Parser를 통해 객체지향 코드를 분석한다. 분석된 소스코드는 Parser의 형식에 맞게 파일로 추출된다. 객체지향 코드는 클래스, 메소드, 변수 등의 요소들로 구성되는데, 객체지향 코드를 이러한 요소들로 분해하고, 각 요소들 간의 연관 관계를 추출한다.

• Step 2: 데이터베이스 저장

데이터베이스 저장 단계에서는 추출된 정보를 분류하여 데이터베이스에 저장한다. 클래스, 메소드, 변수 등의 요소들은 하나의 요소 테이블에 저장한다. 요소들 간의 연관관계는 1:1로 매핑하여 저장한다. 예를 들어 클래스 A가 메소드 a,b,c를 갖고 있다면 A-a, A-b, A-c의 3가지 매핑 정보가 저장된다.

• Step 3: 구조 분석

구조 분석 단계는 데이터베이스에 분류된 정보를 미리 정한 모듈 정의에 따라 재해석한다. 이전 단계(2단계)에서 분류된 요소들로부터 모듈을 추출한다. 다음으로 추출된 모듈과 나머지 요소들 간의 관계 정보를 새롭게 생성한다. 본 논문에서는 모듈 단위를 클래스로 정의하기 때문에 클래스와 클래스, 클래스와 메소드, 클래스와 변수의 연관관계 정보를 생성한다.

• Step 4: 시각화 단계

이 단계에서는 3단계에서 재해석한 정보에서 의미를 찾고 시각화한다. 예를 들어 2단계에서 예로든 A-a, A-b, A-c 정보에서는 클래스 A가 3개의 메소드를 포함한다는 의미이다. 이러한 의미들을 View Composer가 인식 가능한 스크립트로 작성하여 시각화된 뷰를 얻는다.

3.2 SW 품질 향상을 위한 리팩토링

본 논문에서는 결합도(Coupling)와 응집도(Cohesion)를 기반으로 소프트웨어의 품질 측정을 고려한다. 제안한 Tool-Chain 프로세스를 통해 추출된 결과물은 소프트웨어 품질 향상 방안으로 수행한다. 즉, 결합도와 응집도 측정 모듈과 품질 지표를 정의하고, JAVA 코드 패턴을 결정하여, SW 품질 관리 방안을 제시한다. 이 방법은 총 3단계로 구성된다.

• Step 1: 모듈 정의

모듈 정의 단계에서는 시각화 대상 소프트웨어 코드에 적합한 모듈 단위를 정의한다. 객체지향 언어인 JAVA 코드는 객체를 중심으로 설계되기 때문에 객체를 추상화한 클래스(Class)를 모듈로 정의한다.

• Step 2: 품질 지표 정의

소프트웨어 설계 시, 모듈 간 결합도를 최소화하고, 응집도를 높임으로써, 고품질의 소프트웨어 개발이 가능하다. 따라서 결합도와 응집도는 정량적으로 측정 지표를 정한다[4]. 품질 지표 정의 단계에서는 결합도와 응집도의 단계마다 점수를 정의한다. 결합도는 모듈 간에 상호 의존하는 정도 또는 두 모듈 사이의 연관관계를 의미한다[3]. 독립적인 모듈이 되기 위해서는 각 모듈 간의 결합도는 약해야 하며, 의존성은 적어야 한다. 결합도는 자료, 스탬프, 제어, 외부, 공유, 내용 결합도로 구성된다. 내용 결합도로 갈수록 점수가 높아진다. 응집도는 모듈 안의 요소들이 서로 관련되어 있는 정도를

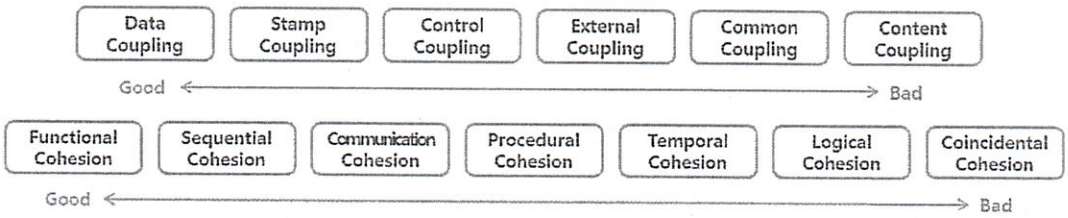


그림 2 결합도와 응집도의 품질 지표
Fig. 2 Quality Indicators of Coupling and Cohesion

의미한다. 독립적인 모듈이 되기 위해서는 각 모듈의 응집도가 강해야 한다[3]. 응집도는 기능, 순차, 교환, 시간, 논리, 우연적 응집도로 구분한다. 우연적 응집도(1)가 가장 낮으며, 기능적 응집도(6)가 가장 높다. 그림 2는 결합도와 응집도의 품질 지표이다.

• Step 3: 코드 패턴 정의

코드 패턴 정의 단계에서 검출하기 위한 코드 패턴을 정의한다. 이 논문에서는 클래스를 모듈 단위로 정의하였기 때문에 클래스와 클래스의 연관관계에 따라 결합도가 결정된다. JAVA에서의 응집도는 메소드 내의 프로세스 절차에 따라 결정된다.

4. Case Study

4.1 객체지향 Code 정적 분석을 위한 Tool-Chain

본 연구에서 사용된 사례는 홍익대학교 소프트웨어공학 연구실의 UseCase Diagram Drawing Tool 코드를 적용하였다. 구축된 Tool-Chain을 구성하는 각각의 오픈 소스 도구들은 Parser: Source Navigator 6.0(SN), Database: SQLite, View Composer: DOT를 사용하였다. UseCase Diagram Drawing Tool은 JAVA를 기반으로 한다.

- 소스 분석 단계는 Source Navigator(SN)에 코드를 입력하면 SNDB 파일을 생성한다. SNDB 파일은 *.|, *.by, *.cl, *.f, *.mi 등의 확장자를 추출한다. 각 파일은 소스코드의 특정 구성요소에 대한 정보를 저장한다. cl은 클래스를 나타내며, mi는 메소드, by는 클래스와 메소드 각각의 연관관계 정보를 나타낸다. 본 사례에서는 cl, mi, by 파일을 통해 클래스와 메소드 정보를 이용하여 시각화 결과물을 추출한다.
- 데이터베이스 저장 단계에서는 SNDB 파일이 바이너리 형태로 정보를 저장하고 있기 때문에 Source Navigator(SN)에 포함된 dbdump를 이용하여 cl, mi, by 파일의 내용을 추출하고 분류하여 저장한다. 그림 3은 SNDB 파일에서 추출된 정보를 DB 스키마에 정의된 분류 체계로 저장하는 과정이다. SNDB 파일의 확장자는 Struct, #define, Enum, Mascro, Typedef 등의 정보를 담고 있다. 각각의 파일로부터 추출한 정보를 DB 스키마에 정의된 분류 체계에 따라 저장한다.
- 구조 분석 단계는 데이터베이스에 저장된 클래스, 메소드, 연관관계 정보를 재해석하여 정량적인 수치 값을 측정한다. 이 수치값은 응집도와 결합도를 사용하였다. 그림 4는 클래스 간의 응집도, 결합도를 정의한 것이다. 클래스 A, B가 각각 메소드 a, b, c, d와 e, f, g, h, i, j와 연관된다. 클래스를 모듈 단위로 하여 응집도와 결

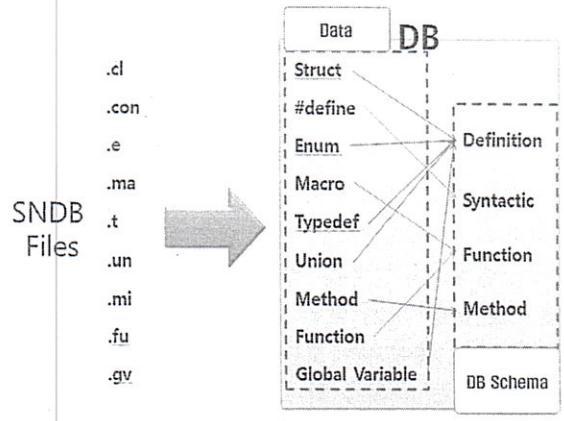


그림 3 SNDB 파일에서 추출된 정보의 분류 및 저장
Fig. 3 Classification and Storage of the Extracted Data in SNDB File

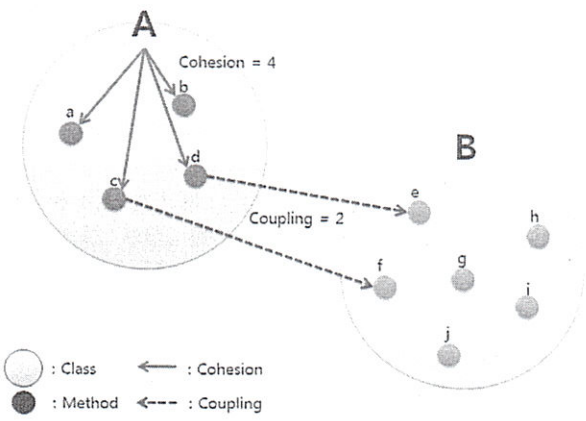


그림 4 클래스 간의 응집도 및 결합도 정의
Fig. 4 Cohesion and Coupling Definition Between the classes[2]

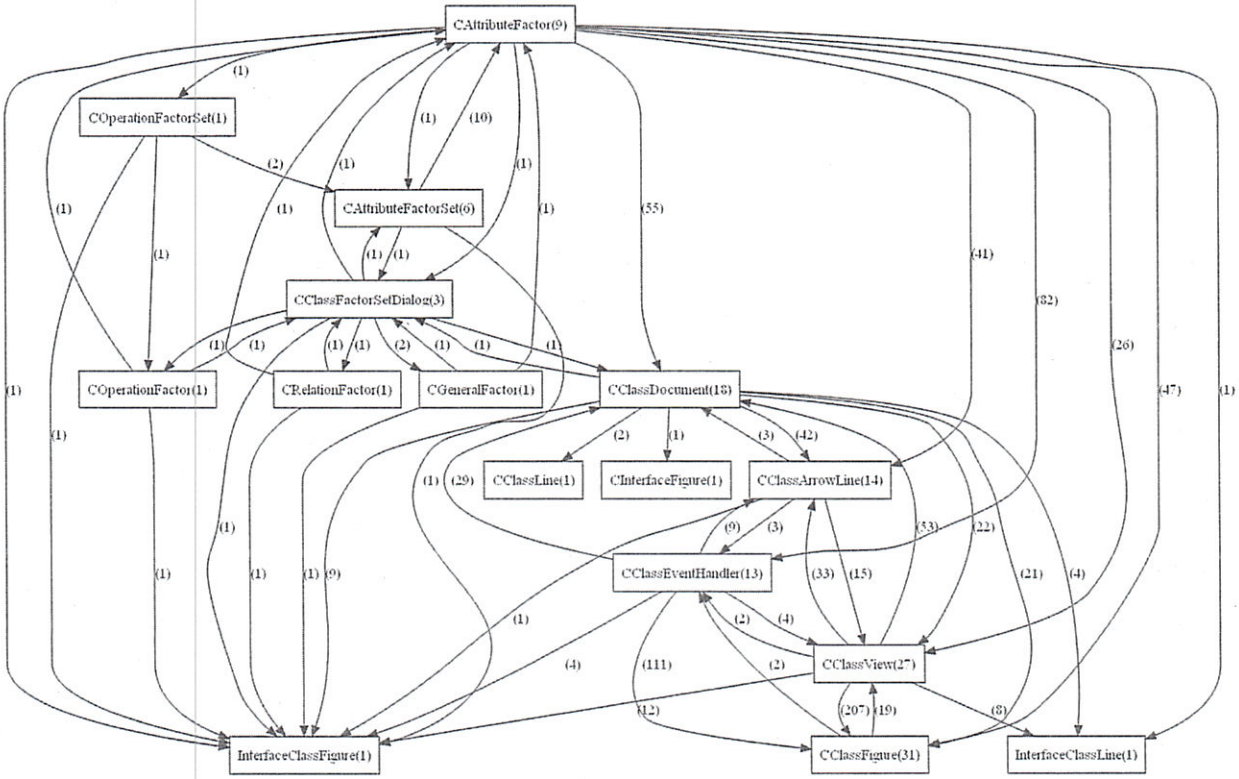


그림 5 코드 시각화 그래프
Fig. 5 Code Visualization Graph

합도를 측정한다. 응집도는 하나의 클래스가 소유한 메소드의 개수로 측정하며, 객체 A는 4개의 메소드를 소유한다. 따라서 A의 응집도는 4가 된다. 결합도는 클래스와 다른 클래스의 메소드와의 관계에 기인하며, 객체 A의 메소드 c, d에서 각각 객체 B의 메소드 e, f가 호출된다. 객체 내에서 다른 클래스의 메소드 호출은 둘 사이의 의존관계가 존재함을 나타내기 때문에 A의 결합도는 2가 된다.

- 시각화 단계는 측정된 클래스의 응집도와 결합도를 DOT를 통해 그림 5와 같이 시각화 한다. 그래프 노드 내부에는 클래스의 식별자와 해당 클래스의 응집도가 표시된다. 노드와 노드를 연결하는 화살표에는 출발 클래스가 가리키는 클래스의 결합도가 나타난다. 그림 5에서는 CClassView 클래스로부터 출발한 화살표가 CClassFigure 클래스를 가리키고 있다. 이는 두 클래스 사이에 결합도가 존재함을 의미한다. 게다가 결합도 값은 207로 다른 결합도에 비해 높은 값을 가진다. 이는 의존도가 높아 CClassFigure 클래스가 수정될 경우, 많은 코드 수정이 일어날 수 있음을 의미한다.

4.2 SW 품질 향상을 위한 리팩토링 과정

Tool-Chain에서 추출된 시각화 그래프와 정보를 이용하여 SW 품질 개선을 위한 리팩토링을 수행한다. 그러나 본 논문에서는 결합도를 이용한 SW의 품질을 측정만을 언급한다.

- 모듈 정의 단계에서는 모듈의 단의를 정의한다. 소프트웨어의 내부 구조를 설계하는 기법으로 추상화, 모듈화, 캡슐화가 있다[1]. 그 중 모듈화(modularity)는 소프트웨어를 각 기능별로 분할하여 각각의 모듈에 대한 복잡도를 낮추고, 독립성을 증가시켜 변형과 구현을 용이하게 한다. 모듈의 독립성은 결합도와 응집도에 의해 측정될 수 있으며, 독립성을 높이려면 모듈 결합도를 약하게 하고 응집도를 강하게 하여 모듈의 크기를 작게 만들어야 한다. 따라서 모듈의 단위를 클래스로 정의하고, 결합도 점수가 100이상인 경우를 제거 대상으로 선정하였다. 또한 반복적인 리팩토링을 수행하여 결합도를 낮추고자 하였다.
- 품질 지표 정의 단계는 결합도의 단계마다 점수를 정의한다. 결합도는 자료, 스탬프, 제어, 외부, 공유, 내용 결합도로 구성된다. 표 1은 6개의 결합도에 대한 정의와 예시, 측정지표를 나타낸다.
- 코드 패턴 정의 단계에서는 검출하기 위한 코드 패턴을 결정한다. 표 1에서 각 결합도에 대한 예시가 코드 패턴이다. 이를 기반으로 코드 작성 기준으로 코딩 규칙을 결정한다. 코딩 규칙을 수립하고 준수하면 코드의 일관성이 유지되고, 코드를 이해하기 쉽다. 표 2는 리팩토링을 위한 코딩 규칙이다.

표 1 결합도 정의, 예시 및 측정 지표
Table 1 Coupling Definition, Examples and Measures

Coupling	Data	Stamp	Control	External	Common	Content
Definition	On calling a method of other class, the method's parameters are consisted of the basic data types.	On calling a method of other class, the method's parameters are consisted of an object or an array.	When the return types of other class calls "boolean" method.	When calls the object using the external resources such as File, Font, and FontRenderContext.	When accesses the public or static variables or methods as statically declared from other class.	• Directly accesses to The variable owned by different class. • Exception Handling

Examples	getSum(3, 5);	int array[3]={1,2,3};
		
		getSum(array);
			if(circle.getRadisu() < 0.0) {
			
			circle.drawReverse();
			} else {
			circle.draw();
			}
				File config_txt = new File("config.txt");	Double d = getInput();
					
					Math.round(d);
						Graph g = new Graph();
						g.circle.radius=5.0;
						
						}catch(Exception e){
						.....
						}

Measures	1	2	3	4	5	6

표 2 리팩토링을 위한 코딩 규칙
Table 2 Coding Rules for Refactoring

Type	Naming Rule	Identifier Rule	SQL Query
Class	C_	The first letter is an uppercase. The blank is represented by the underbar(_). The first letter is an uppercase after a blank.	class C_Human{ }
Interface	I_	The first letter is an uppercase. The blank is represented by the underbar(_). The first letter is an uppercase after a blank.	interface I_Runnable { }
Method	M_	The first letter is an lowercase. The blank is represented by the underbar(_). The first letter is an uppercase after a blank.	int M_getSum( );
Global Value (Class Attribute)	c_	The first letter is an lowercase. The blank is represented by the underbar(_). The first letter is an uppercase after a blank.	int c_height;
Parameter Value	P_	The first letter is an lowercase. The blank is represented by the underbar(_). The first letter is an uppercase after a blank.	M_getSum(int p_a, int p_b);
Local Value	m_	The first letter is an lowercase. The blank is represented by the underbar(_). The first letter is an uppercase after a blank.	String m_tmpStr;
Record(Array, List, Map...)	r_	The first letter is an lowercase. The blank is represented by the underbar(_). The first letter is an uppercase after a blank.	int[ ] r_m_list;
Static Method	S_	The main method is excluded. The first letter is an lowercase. The blank is represented by the underbar(_). The first letter is an uppercase after a blank.	public static int S_M_getD( );
Static Value	S_	The first letter is an lowercase. The blank is represented by the underbar(_). The first letter is an uppercase after a blank.	static int S_c_version;

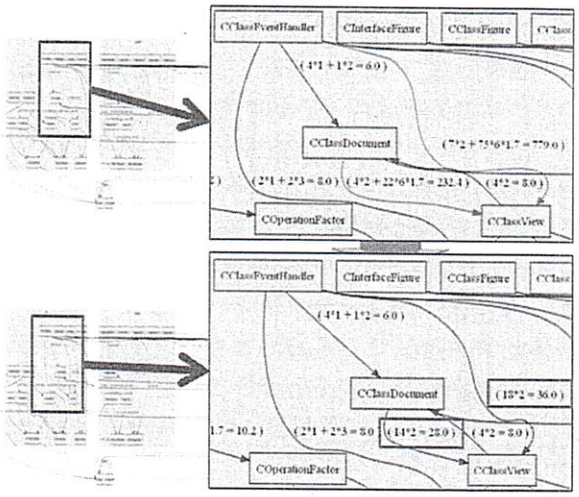


그림 6 내용 결합도의 리팩토링 결과(클래스단위)
Fig. 6 Refactoring Results of Content Coupling(Class)

4.3 개선 결과

그림 6은 내용 결합도에 대한 리팩토링 수행 결과이다(클래스 단위). 빨간 선은 결합도가 100 이상인 것을 나타낸다. 리팩토링 하기 전 CClassEventhandler와 CClassView 클래스 간의 결합도는 779이다. 이 부분을 리팩토링 함으로써 36으로 줄일 수 있다. 그림 7은 내용 결합도를 패키지 단위로 리팩토링을 수행한 결과이다. Class 패키지의 결합도는 1133이다. 리팩토링을 수행함으로써 Class 패키지의 결합도를 187.6으로 줄일 수 있다.

본 연구에서는 총 3번의 리팩토링을 수행하였다. 2번의 내용 결합도 리팩토링을 수행하였으며, 1번의 공통 결합도 리팩토링을 수행하였다. 그림 8은 리팩토링에 따른 결합도 변화이다. baseline은 리팩토링을 수행하지 않은 최초의 경우를 나타내며 checkpoint1과 2는 내용 결합도에 대한 리팩토링이다. checkpoint3은 공통 결합도를 나타낸다. 각각의 결합도에서 감소된 구간은 각각 다르지만, 전체적으로 결합도가 감소된 것을 알 수 있다. 최초 baseline 대비 4000정도의 결합도를 감소시켰다.

5. 결론 및 향후 연구

본 연구에서는 고품질의 소프트웨어를 위해서, 기존 코드의 문제점을 발견하고 변경(수정)하는 것과 소프트웨어 개발자들의 나쁜 습관을 개선하는 것이 주목적이다. 이러한 문제의 해결 방법으로 코드 가시화를 통해서 개발자의 도움 없이 기존 시스템 분석 및 구조를 파악하여 리팩토링을 하고자 한다. 이를 통해 역공학 기법 적용 및 기존 오픈소스 도구기반 Tool-Chain을 구성하였다. 사례로써, 객체지향 코드의 정적분석을 통해 결합도 기반으로 추출된 시각화 그래프는 가시성 및 결합도의 정량적 수치를 제공하여 소프트웨어 품질을 측정하였다. 이 결과물들은 리팩토링 과정을 수행함으로써 각각의 결합도를 감소시킬 수 있었다. 이러한 과정을 통해 개발 중인 소프트웨어의 진척 상황을 실시간으로 확인 가능하며, 코드의 정량적인 분석이 가능하다. 또한 개발자들이 개발한 코드가 어느 정도의 기준으로 결합도 문제를 일으켰는지 확인 가능하기 때문에 개발자의 나쁜 습관을 개선할 수 있을 것으로 기대한다. 하지만 본 논

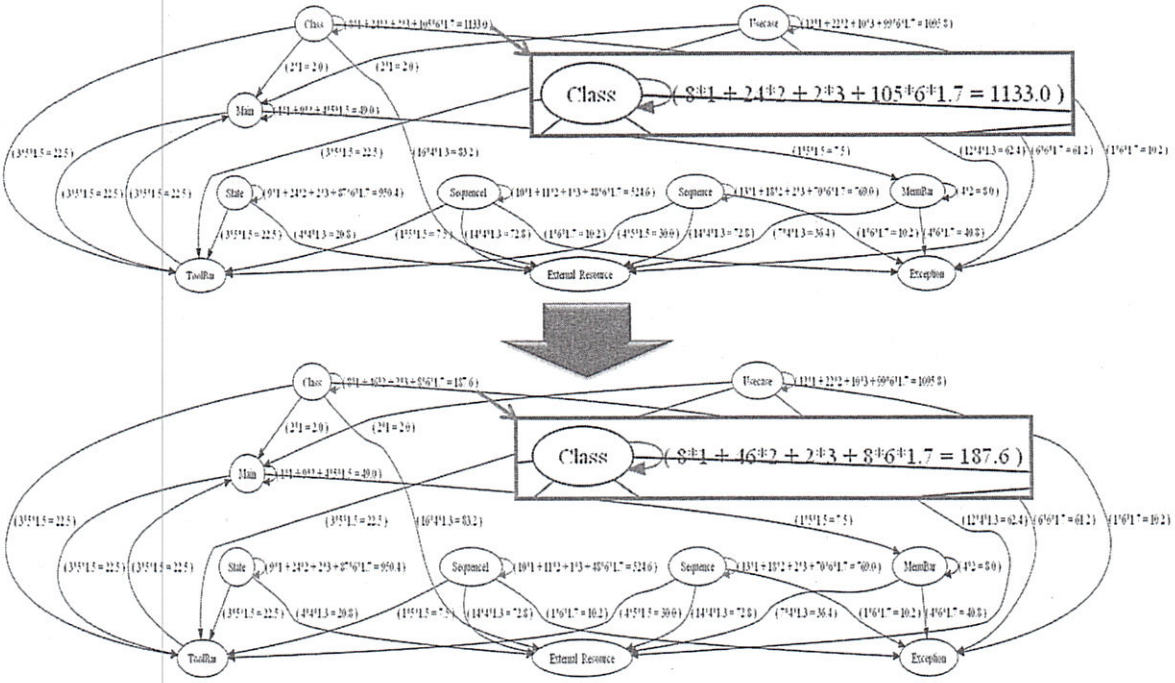


그림 7 내용 결합도의 리팩토링 결과(패키지 단위)

Fig. 7 Refactoring Results of Content Coupling(Package)

	Total	Class	Main	State	ToolBar	Usecase	Sequence	Sequence1	MenuBar	etc
baseline	5172.3	1133	49	950.4	0	1095.8	769	524.6	8	642.5
checkpoint1	4226.9	187.6	49	950.4	0	1095.8	769	524.6	8	642.5
checkpoint2	1258.1	187.6	49	91	0	114	97	69	8	642.5
checkpoint3	1220.6	187.6	19	91	0	114	97	69	8	635

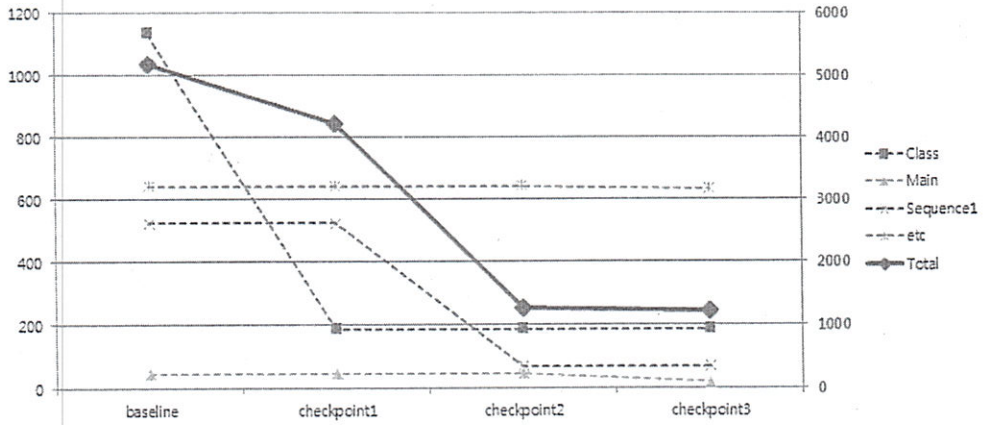


그림 8 리팩토링에 따른 결합도 변화

Fig. 8 Coupling Change according to Refactoring

문에서는 결합도 부분만을 측정하였다. 향후에는 응집도 기반의 소프트웨어 품질 측정을 계속적으로 연구할 예정이며, 다양한 사례에 적용하고자 한다.

References

[1] NIPA SW Engineering Center, *SW Development Quality Management Manual (SW Visualization)*, 2013.

[2] Bokyung Park, Haeun Kwon, Hyeoseok Yang, Soyoung Moon, Youngsoo Kim, R. Youngchul Kim, "A Study on Tool-Chain for statically analyzing Object Oriented Code," *KCC2014*, pp. 463-465, 2014.

[3] E. M. Choi, *Object-Oriented Software Engineering*, Scitech, 2005.

[4] Haeun Kwon, Hyun Seung Son, Chae Yun Seo, Youngsoo Kim, Byung Ho Park, R. Youngchul Kim, "A study on Comparing Object Oriented Paradigm

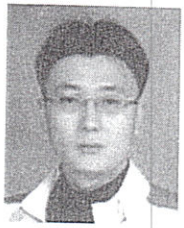
with the Cohesing and Coupling mechanism between Traditional modules," *KCC2014*, pp. 556-558, 2014.

박 보 경

2008년 홍익대학교 컴퓨터정보통신(학사) 2012년 홍익대학교 일반대학원 소프트웨어공학(석사). 2012년~현재 홍익대학교 일반대학원 소프트웨어공학 박사과정. 관심분야는 소프트웨어공학, 요구공학, 역공학

권 하 은

2013년 홍익대학교 컴퓨터정보통신(학사) 2014년~현재 홍익대학교 일반대학원 소프트웨어공학 석사과정. 관심분야는 소프트웨어공학, 역공학, 메타모델



손 현 승

2007년 홍익대학교 컴퓨터정보통신(학사) 2009년 홍익대학교 일반대학원 소프트웨어공학(석사). 2009년~현재 홍익대학교 일반대학원 소프트웨어공학 박사과정. 관심분야는 임베디드 소프트웨어 자동화 도구 개발, 임베디드 RTOS 개발, 메타모델 설계 및 모델 변환, 모델 검증 기법 연구

김 영 수

1992년 광운대학교 수학과(학사). 1994년 광운대학교 전자계산학과(석사). 2010년~현재 정보통신산업진흥원 SW공학센터, 정보처리기술사(시스템 응용), 현장 멘토링 총괄, SW Visualization 품질 멘토링. 관심분야는 임베디드 소프트웨어공학, 역공학/재공학 기법 연구

이 상 은

1980년 서울대학교 공과대학 전자공학(학사). 1995년 서강대학교 경영대학원(석사). 2008년 호서대학교 벤처전문대학원(박사). 1982년~1986년 (주)LG전자 중앙연구소. 1986년~1994년 (주)한국휴렛팩커드 솔루션개발 부장. 1994년~2000년 (주마이크로소프트 상무, 기술지원본부 및 컨설팅 본부 이사, 파트너사업부 및 솔루션사업부 총괄 이사. 2000년~2001년 (주)인포섹 대표이사. 2002년~2003년 (주)한국내쇼날소프트웨어 한국 대표이사. 2003년~2009년 한국소프트웨어진흥원 소프트웨어공학단 단장. 2009년~현재 정보통신산업진흥원 SW공학센터 센터장

김 영 철

2000년 Illinois Institute of Technology (IIT)(공학박사). 2000년~2001년 LG산전 중앙연구소 Embedded system 부장 2001년~현재 홍익대학교 컴퓨터정보통신 교수. 관심분야는 테스트 성숙도 모델(TMM), 임베디드 소프트웨어 개발 방법론, 모델 기반 테스팅, 메타모델, 비즈니스 프로세스 모델, 사용자 행위 분석 방법론 등

Journal of KIISE

VOLUME 41, NUMBER 11, NOVEMBER 2014

Computer Systems and Theory

A New Cache Replacement Policy for Improving Last Level Cache Performance ····· Cong Thuan Do, Dong Oh Son, Jong Myon Kim, Cheol Hong Kim 871

Improving Periodic Flush Overhead of File Systems Using Non-volatile Buffer Cache ····· Eunji Lee, Hyojung Kang, Kern Koh, Hyokyung Bahn 878

An Implementation of an SHA-3 Hash Function Validation Program and Hash Algorithm on 16bit-UICC ····· Hee-Woong Lee, Dowon Hong, Hyun-il Kim, ChangHo Seo, Kishik Park 885

Design and Implementation of a Dynamic Instrumentation Framework based on Light-weight Dynamic Binary Translation ····· Jeehong Kim, Dongwoo Lee, Inhyeok Kim, Young Ik Eom 892

Software and Applications

Improved Prediction Structure and Motion Estimation Method for Multi-view Video Coding ····· Hyo Sun Yoon, Mi Young Kim 900

Developing an Automated English Sentence Scoring System for Middle-school Level Writing Test by Using Machine Learning Techniques ····· Gyoung Ho Lee, Kong Joo Lee 911

Detection of Faces with Partial Occlusions using Statistical Face Model ····· Jeongin Seo, Hyeyoung Park 921

Human Activity Recognition using View-Invariant Features and Probabilistic Graphical Models ····· Hyesuk Kim, Incheol Kim 927

A Case Study on Improving SW Quality through Software Visualization ····· Bo Kyung Park, Ha Eun Kwon, Hyun Seung Son, Young Soo Kim, Sang-Eun Lee, R. Young Chul Kim 935

Databases

Partial Denoising Boundary Image Matching Based on Time-Series Data ····· Bum-Soo Kim, Sanghoon Lee, Yang-Sae Moon 943

Information Networking

Distributed Computing Models for Wireless Sensor Networks ····· Chongmyung Park, Chungsan Lee, Youngtae Jo, Inbum Jung 958

An Energy-Efficient Location Update Scheme for Mobile Sinks in Continuous Object Detection Using Wireless Sensor Networks ····· Cheonyong Kim, Hyunchong Cho, Sangdae Kim, Sang-Ha Kim 967

A Router Buffer-based Congestion Control Scheme for Improving QoS of UHD Streaming Services ····· Junyeol Oh, Dooyeol Yun, Kwangsue Chung 974

A Car Black Box Video Data Integrity Assurance Scheme Using Cyclic Data Block Chaining ····· Kang Yi, Kyung-Mi Kim, Yong Jun Cho 982

■ 2014 Best Paper Contest for Students

Korean Coreference Resolution using the Multi-pass Sieve ····· Cheon-Eum Park, Kyoung-Ho Choi, Changki Lee 992

KOREAN INSTITUTE OF INFORMATION SCIENTISTS AND ENGINEERS