ISSN 2287-4348
Vol.4 No.2

한국스마트미디어학회
2015 추계학술대회
KISM Fall Conference 2015

2015.10.23(금)~24(토)

장소 : 조선대학교 전자정보공과대학(IT공과대학)

주최	(사)한국스마트미디어학회
주관	조선대학교, 미래창조과학부
후원	조선대학교 산업융합특성화인재양성사업단
	조선대학교IT연구소
	조선대학교 미술대학
	비온시이노베이터
	인포데이타
	콤텍시스템
	디자인바이
	LIG시스템

Smart Media
KOREAN INSTITUTE OF SMART MEDIA
한국스마트미디어학회

논문 발표순서

Oral 발표 - Session 4 . IS / Smart Information / Smart Media

10월 24일 (토) 09:00 ~ 11:00	세미나실1 / 좌장 : 김영철(홍익대)

09:00-09:15 제목 : **사실적인 폭파효과를 구현하기 위한 기술적 방법 제시**
저자 : 김동식(동서대), 황민식(동서대), 김용희(동서대), 윤태수(동서대)

09:15-09:30 제목 : **IoT환경에서 요구되는 사이버보안기술**
저자 : 손명희(SW・콘텐츠연구소), 김정녀(SW・콘텐츠연구소)

09:30-09:45 제목 : **사물인터넷을 위한128-bit LEA 하드웨어 구현 연구**
저자 : 윤기하(전남대), 박성모(전남대)

09:45-10:00 제목 : **주야간 사고예방을 위한 차량용 블랙박스 시스템에 관한 연구**
저자 : 김강효(조선대), 반성범(조선대)

10:00-10:15 제목 : **객체 기반 프로그래밍으로부터 구조적 및 행위적 설계 자동 추출**
저자 : 권하은(홍익대), 박보경(홍익대), 김영철(홍익대)

10:15-10:30 제목 : **클라우드 서비스 기반의 SW Visualization 시스템 설계**
저자 : 황준순(홍익대), 손현승(홍익대), 이근상(홍익대), 김영철(홍익대)

10:30-10:45 제목 : **소프트웨어 정량적 시간반응성을 통한 소프트웨어 구조적 설계 개선**
저자 : 강건희(홍익대), 김영철(홍익대)

10:45-11:00 제목 : **유스케이스 기반의 요구사항 분석을 통한 Goal과 Risk 상관관계 추출**
저자 : 박보경, 장우성, 강건희, 권하은, 변은영, 김영철(홍익대)

소프트웨어 정량적 시간반응성을 통한 소프트웨어 구조적 설계 개선

강건희, 김영철
홍익대학교 소프트웨어공학연구실
e-mail : {kang, bob}@selab.hongik.ac.kr

Improving The Structural Design through Software Quantitative Time behavior

Geon-Hee Kang, R. Young Chul Kim
Software Engineering Laboratory, Hongik University

요 약

IT기술의 발전으로 인해서 우리사회에서의 소프트웨어 비중이 점점 늘어가고 있다. 그래서 소프트웨어 고품질화에 대한 이슈가 대두되고 있다. 소프트웨어의 고품질을 위한 지표는 다양하다. 이전 연구[1]에서는 소프트웨어의 재사용성과 유지보수성을 통한 소프트웨어의 고품질화 관해서 소프트웨어 결합도 개선에 대해 서술하였다. 하지만 소프트웨어의 품질을 평가하는 지표는 매우 다양하다. 그래서 소프트웨어 구조적 설계를 개선함으로서 소프트웨어 시간반응성에 대한 정량적인 변화에 대해 알아보았다. 이로 인해 실제 소프트웨어 개발현장에서 소프트웨어 재사용성과 시간반응성의 관계를 판단하여 요구사항에 맞는 고품질의 소프트웨어를 개발할 수 있을 것으로 판단된다.

1. 서 론

IT기술의 발전으로 인해서 하드웨어의 발전도 많이 이루어졌지만 소프트웨어의 시장의 엄청난 발전으로 인해 다양한 소프트웨어가 만들어지고 있고, 우리 생활 다양한 곳에 이미 소프트웨어가 존재하지 않는 곳이 없다. 그렇기 때문에 고품질의 소프트웨어를 만들어야 한다. 소프트웨어 공학적으로 고품질의 소프트웨어라고 평가하기 위한 다양한 품질지표가 존재한다[2]. 그 중 소프트웨어의 유지보수성이 높은 상품을 개발하기 위해서는 소프트웨어의 응집도가 높고 결합도가 낮아야 한다. 그러나 응집도와 결합도만으로는 다양한 방면의 고품질을 이룰 수는 없다.

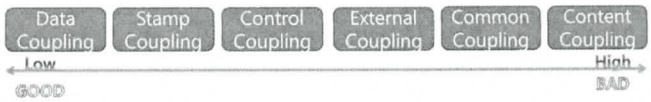
그래서 본 논문에서는 소프트웨어의 구조적 설계를 개선을 함으로서 소프트웨어의 시간반응성의 어떠한 변화가 생기는지에 대해서 확인하고 시간반응성 상의 고품질이 될 수 있는지에 대해서 알아본다.

본 논문은 2장 관련연구로 소프트웨어 결합도와 소프트웨어 시간반응성에 대해서 언급하고 3장에서는 소프트웨어 구조 개선에 대한 설명과 개선에 따른 시간반응성의 변화에 대해 보여준다. 마지막으로 4장에서는 결론에 대해서 서술한다.

2. 관련연구

2.1 소프트웨어 결합도

소프트웨어의 결합도는 프로그램에서 모듈간의 상호 의존 혹은 연관관계이다. 그래서 강한 결합도는 모듈 간의 의존성을 높여 소프트웨어의 변경, 유지보수 더 나아가 모듈의 재 사용성에 엄청난 영향을 끼친다. 결합도는 자료, 스탬프. 제어, 외부, 공유, 내용 총 6가지의 결합도가 존재한다. 자료결합도에서 내용결합도로 갈수록 모듈간의 상호의존성이 높아진다. 그렇기 때문에 소프트웨어의 유지보수성이 떨어지게 된다. 그리고 소프트웨어의 유지보수성이 떨어질수록 코드의 재사용성이 떨어져 유지보수에 들어가는 시간과 자본이 많이 들게 된다[3]. 그림 1 은 결합도의 상관관계를 표현한 그림이다.



(그림 1) 결합도의 상관관계

본 연구에서는 이전의 연구[1]를 통해 높은 결합도를 리펙토링 하여 낮추는 기법을 소개 하였다. 해당 기법을 사용하여 소프트웨어의 복잡도를 개선함과 동시에 소프트웨어 수행시간이 어떠한 변화가 생기는지 알아보고자 한다.

2.2 소프트웨어 시간반응성

소프트웨어 시간반응성은 ISO9126-1[4]에서의 소프트웨어의 6가지 특성(기능성, 신뢰성, 사용성, 효율성, 유지보

* 이 논문(저서)은 2015년 교육부와 한국연구재단의 지역혁신창의인력양성사업(NRF-2015H1C1A1035548)과 2015년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(NRF-2013R1A1A2011601)

수성, 이식성)에서 효율성에 포함되는 3가지 부특성(시간반응성, 자원효율성, 준수성)중 하나이다. ISO 9126에서 효율성은 명시된 조건에서 사용되는 자원의 양에 따라 요구된 성능을 제공하는 소프트웨어의 능력 이라고 정의 되고 있고, 시간반응성은 명시된 조건에서 그 기능을 수행할 때 적절한 반응 및 처리시간과 처리율을 제공하는 능력이라고 정의 되어있다.

3. 소프트웨어 정량적 시간반응성을 통한 소프트웨어 구조적 설계 개선

본 연구에서는 소프트웨어의 구조적 설계 개선을 위해 스탬프결합도에서 자료결합도, 외부결합도에서 스탬프결합도, 공유결합도에서 자료결합도, 내용결합도에서 자료결합도의 4가지 경우를 리펙토링 하고 시간반응성을 측정 하였다.

3.1 스탬프결합도->자료결합도의 변화

스탬프결합도일 때는 SampleDataClass 객체를 선언하여 연산에 필요한 데이터(int, float, double, long)를 입력을 한 뒤 객체를 Stamp클래스의 stampTest 메소드에 전달한다. 그리고 메소드 내부에서 getter를 통해 값을 받아 10만 번 곱 연산을 하였다. 그리고 자료결합도로 리펙토링 한 후에는 직접 값을 입력받아Data클래스의 dataTest메소드에서 10만 번 곱 연산을 실행하였다. 표 1는 스탬프결합도에서 자료결합도로 리펙토링한 소스코드이다.

(표 1) 스탬프결합도 -> 자료결합도 리펙토링 소스코드

	Source Code
(전)	Stamp stmp = new Stamp(); SampleDataClass sdc = new SampleDataClass(1, 1.01f, 2, 2); stmp.stampTest(sdc);
자료 (후)	Data data = new Data(); data.dataTest(1, 1.01f, 2, 2);

스탬프결합도에서 자료결합도로 변경한 결과 1.1686(s)에서 1.1617(s)로 미비하지만 0.59% 속도가 감소한 것을 볼 수 있다.

3.2 외부결합도->스탬프결합도의 변화

외부결합도의 리펙토링은 서로 다른 모듈이 하나의 외부 자료를 참조할 때 외부데이터를 불러오는 하나의 모듈을 따로 만들고 해당 모듈을 참조하게 한다. 표 2은 외부결합도를 스탬프결합도로 리펙토링한 소스코드이다.

(표 2) 외부결합도 -> 스탬프결합도 리펙토링 소스코드

	Source Code

외부
(전)
	External1 ex1 = new External1();
	External2 ex2 = new External2();
	ex1.RamdomNumberRead();
	ex2.RamdomNumberRead();

스 탬 프
(후)
	External ex = new External();
	External1 ex1 = new External1();
	External2 ex2 = new External2();
	ArrayList<Integer> arr = ex.RamdomNumberRead();
	ex1.calculate(arr);
	ex2.calculate(arr);

외부결합도를 지닌 각 클래스의 RamdomNumberRead 메소드에서 Random.txt파일에 있는 임의의 10만개의 값을 받아 곱 연산을 하는데 0.04236(s)가 소요 되었다. 그리고 스탬프 결합도로 리펙토링한 후 arr변수에 Random.txt의 임의의 10만개 값을 저장한 뒤 calculate메소드에 전달하여 곱 연산 하는데 0.01980(s)소요 되어 의 속도향상을 보였다.

3.3 공유결합도->자료결합도의 변화

공유결합도는 공유변수참조를 getter혹은 setter로 바꿈으로서 자료결합도로 낮출 수 있다. 표 3는 공유결합도를 자료결합도로 리펙토링한 소스코드이다.

(표 3) 공유결합도 -> 자료결합도 리펙토링 소스코드

	Source Code
공유 (전)	Common.common *= Common.common;
자료 (후)	int common = Common.getCommon(); Common.setCommon(common*common);

표 3과 같이 공유변수를 직접 참조하여 곱 연산을 하던 것을 getter를 통해 값을 받고 setter를 통해 값을 저장하게 하여 결합도를 낮추었다. 이 연산을 10만번 반복하여 실행하였다. 공통결합도일 때는 0.004026(s)에서 자료결합도일 때 0.006975(s) 73.25%속도가 감소하였다.
기존에 존재하던 메모리를 직접 참조하는 방식이 아닌 getter나 setter를 사용하여 값을 제어하기 때문에

3.4 내용결합도->자료결합도의 변화

내용결합도는 한 모듈이 다른 모듈의 변수를 직접 참조하거나 사용하는 것을 공유결합도를 리펙토링한 것과 같이 getter와 setter를 통해 결합도를 낮추게 된다. 표 4는 내용결합도를 자료결합도로 리펙토링한 소스코드이다.

(표 4) 내용결합도 -> 자료결합도 리펙토링 소스코드

	Source Code
내용 (전)	result *= cnt.contents1*(int)cnt.contents2*(int)cnt.contents3*(int)cnt.contents4;
자료 (후)	result *= cnt.getContents1()*(int)cnt.getContents2()*(int)cnt.getContents3()*(int)cnt.getContents4();

내용결합도도 마찬가지로 직접 참조하던 변수을 getter와 setter를 통해 참조를 하여 결합도를 줄이게 된다. 참조 받은 값을 모두 곱하는 곱 연산을 10만 번 하여 내용결합도일 때는 0.1204(s)에서 자료 결합도일 때 0.1264(s)로 4.98% 속도가 감소하였다. 공유결합도 보다 속도의 증가폭이 크지 않은 이유는 setter를 통한 데이터의 입력이 없기 때문으로 판단된다.

4. 결론

본 연구는 소프트웨어의 유지보수성과 재사용성을 높이기 위한 결합도의 개선을 통해서 소프트웨어의 시간반응성에 대한 변화에 대해 시험을 해보았다. 스탬프결합도를 자료결합도로, 외부결합도를 스탬프결합도로 개선하였을 때는 각각 0.59%, 53.26% 속도의 향상을 보였지만 상대적으로 높은 결합도인 공유결합도와 내용결합도를 자료결합도로 개선할 때 에는 73.25%, 4.98% 속도감소를 보였다. 이와 같이 소프트웨어의 구조적 개선과 시간반응성의 관계를 통해 실제 개발현장에서 요구사항에 맞는 품질지표를 올바르게 판단하고 소프트웨어의 고품질화에 도움이 될 것으로 판단된다.

참 고 문 헌

[1] 강건희, 손현승, 김영수, 박용범, 김영철. “SW 가시화 기반 리펙토링 기법 적용을 통한 정적 코드 복잡도 개선”, 『한국정보처리학회 춘계학술발표논문집』 제221권 제2호, 2014. pp.646-649.

[2] Nipa Software Engineering Center “SW Development Quality Management Manual”, 2013.

[3] Arisholm, Erik, Lionel C. Briand, and Audun Foyen. "Dynamic coupling measurement for object-oriented software." Software Engineering, IEEE Transactions. Vol.30 No.8, 2004, pp.491-506.

[4] ISO, “ISO/IEC 9126 : Information Technology - Software Quality Characteristics and Metrics“, 1996.

한국스마트미디어학회
2015 추계학술대회

Proceedings of KISM Fall Conference 2015

제 4권 제 2호
2015년 10월 19일 인쇄
2015년 10월 19일 발행

발행인 / 장병완 대회장
편집인 / 서창호, 반성범, 김경백, 김병기 학술위원장
발행처 / (사) 한국스마트미디어학회
광주 남구 송암로60 광주CGI센터 기업동 3층 (송하동)
전화 : 062)655-3507~9 / 팩스 : 062)655-3510
홈페이지 : www.kism.or.kr
E-Mail : kism1122@kism.or.kr
디자인 및 편집 / 류시천 조직위원장, 백일디자인
인쇄 / 백일디자인