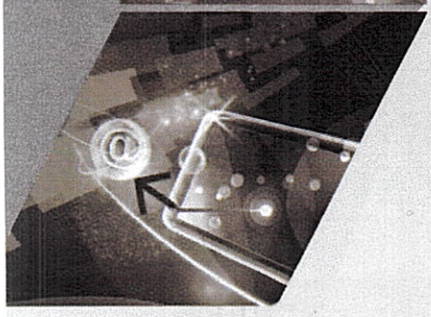
한국정보과학회
KOREAN INSTITUTE OF INFORMATION SCIENTISTS AND ENGINEERS

http://www.kiise.or.kr

2014 한국컴퓨터종합학술대회
Korea Computer Congress 2014

"사물 인터넷 시대의 SW 기술"



2014년 6월 25일(수)~27일(금)
부경대학교&해운대그랜드호텔

- 자연어처리 및 정보검색 최근 동향 워크샵(6.25)
- HPC 최근 연구 동향 워크샵(6.25)
- 데이터베이스 최근 연구 동향 워크샵(6.25)
- GPU Standard API Workshop(6.25)
- 인공지능 최근 연구 동향 워크샵(6. 26)

|후 원|

플래티넘 NAVER KOFST

골 드 SAMSUNG 삼성전자 SAMSUNG 삼성테크윈 bto 부산관광공사

실 버 SAMSUNG 삼성SDS 신한데이타시스템

브론즈 hancom 한글과컴퓨터 KCC정보통신 Suresoft

기존 모듈 간의 결합도 및 응집도 개념과 객체지향 파라다임과의 관련 비교 연구

권하은[O1] 손현승[1] 서채연[1] 김영수[2] 박병호[1] 김영철[1]
[1]홍익대학교 컴퓨터정보통신공학과 소프트웨어공학연구실
[2]정보통신산업진흥원
[1]{kwon, son, jyun, parkbh, bob}@selab.hongik.ac.kr, [2]ysgold@nipa.kr

A study on Comparing Object Oriented Paradigm with the Cohesion and Coupling mechanism between Traditional modules

Haeun Kwon[O1] Hyun Seung Son[1] Chae Yun Seo[1] Youngsoo Kim[2] Byung Ho Park[1] R. Youngchul Kim[1]
[1]SE lab, Dept. of Computer Information Comunication, Hongik University
[2]National IT Industry Promotion Agency

요 약

현재의 소프트웨어는 다기능 및 대규모화에 따른 체계적인 품질 관리가 필요하다. 기존의 소프트웨어 개발자는 나쁜 코드 습관으로 코드 자체의 결합도와 응집도를 고려하지 못 한다. 또한 고급 인력과 자동화 시스템의 부족, SW 비가시성으로 인해 SW 품질관리가 어렵다. 본 논문에서는 SW 품질 및 재사용 향상을 위해 기존 모듈과 객체지향 메카니즘을 결합도와 응집도를 기준으로 비교분석하였다. 이를 통해 추가적인 품질 지표 방법을 고려한다. 이종언어인 절차식(C) 언어의 모듈과 객체지향(JAVA) 언어의 객체 특성에 맞게 적용하여, 좋은 SW 개발 습관과 재사용성을 높이고 가시성 확보를 위한 품질 지표로 활용하고자 한다.

1. 서 론

오늘날 소프트웨어는 점차 다기능화 및 대규모화 되어 가는 추세로, 소프트웨어 개발 프로세스 전반에 걸친 체계적인 품질 관리 기술의 필요성이 증가하고 있다. 하지만 SW 개발 프로세스를 지속적으로 수행할 인력과 자동화된 시스템의 부족으로 수행하기에 어려운 실정이다[1].

또 다른 SW 품질 관리의 어려움은 SW의 비가시적인 특성에 있다. SW 개발 과정에서 발생하는 문제점은 개발 후반에 발생할수록 이를 해결하기 위한 비용과 노력이 증가한다. 그렇기 때문에 개발 과정 전반에 걸친 가시성 확보를 위한 노력이 요구된다. 기존의 소프트웨어는 절차식 및 객체지향 언어로 개발되고 있기 때문에, 각 언어의 특성에 맞는 품질 관리 방안이 필요하다. 기존의 절차식 방법론에서는 단위들 간의 의존도 측정 척도인 결합도와 하나의 단위 요소들 간의 관계 측정 척도인 응집도를 품질 척도로 활용하였다[3]. 그러나 객체지향 방법에서는 캡슐화(encapsulation), 상속(inheritance), 다형성(polymorphism)과 같은 개념을 이용하기 때문에, 절차식 방법론의 품질 척도와 다르게 적용되어야 한다.

본 논문에서는 결합도(Coupling)와 응집도(Cohesion)를 기반으로 한 소프트웨어 품질 향상 방안을 제안한다. 이를 위해 C/JAVA 소스코드에 대한 품질 지표를 NIPA의 SW Visualization 기법에 적용한다. 절차식 언어와 객체지향 언어의 특성이 다르기 때문에, C와 JAVA로 개발된 소스코드를 각각 이용한다. 결합도와 응집도 측정 모듈과 품질 지표를 정의하고, C/Java 언어에서의 코드 패턴을 결정하여, SW 품질 관리 방안을 제시한다.

2. 본 론

결합도와 응집도를 기반으로 한 소프트웨어 품질 향상 방안은 모듈 정의, 품질 지표 정의, 코드 패턴 분석의 3단계로 구성된다.

Step1. 모듈 정의

모듈 정의 단계는 시각화 대상 소프트웨어 코드에 적합한 모듈 단위를 정의하는 단계이다. C 코드는 기능의 최소 단위인 함수(Fuction)를 모듈로 정의한다. 반면에 객체지향 언어인 Java 코드는 객체를 중심으로 설계되기 때문에 객체를 추상화한 클래스(Class)를 모듈로 정의한다.

* 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).

Step2. 품질 지표 정의

소프트웨어 설계 시, 모듈간 결합도를 최소화하고, 응집도를 높임으로써, 고품질의 소프트웨어 개발이 가능하다. 따라서 결합도와 응집도는 정량적으로 측정되어야 한다. 품질 지표 정의 단계에서는 결합도와 응집도의 단계마다 점수를 정의한다. 결합도는 자료, 스탬프, 제어, 외부, 공유, 내용 결합도로 구성된다. 자료 결합도(1)가 가장 낮은 점수를 가지며, 내용 결합도(6)가 가장 높다. 응집도는 기능적, 순차적, 교환적, 시간적, 논리적, 우연적 응집도의 6단계로 구분한다. 기능적 응집도(6)가 가장 높은 응집도를 가지며, 우연적 응집도(1)가 가장 낮은 응집도를 갖는다.

그림 1 결합도, 응집도 품질지표

Step 3. 코드 패턴 정의

코드 패턴 정의 단계는 코드에서 검출하기 위한 코드 패턴을 정하는 단계이다. C와 Java에서의 모듈 정의가

표 1 결합도 응집도 코드 패턴

(가) 결합도 코드 패턴

	Java	C

	`Circle c = new Circle();`
	`c.setRadius(3);`
		`void ReadPrint() {`
		`  char c='a';`
		`  PrintC(c);`
		`}`
		`void PrintC(char ch) {`
		`  putchar(ch);`
		`}`

	`int array[3]={1, 2, 3};`
	`MathUtil mu = new`
	`MathUtil();`
	`mu.getSum(array);`
		`void Read_and_Print() {`
		`  char aStat [256];`
		`  rStat(aStat);`
		`}`
		`void rStat(char* a) {`
		`  getChar(a);`
		`}`

	`SubPro sp=new SubPro();`
	`ListOb lo=new ListOb();`
	`if(lo.isEmpty()==true) {`
	`  sp.sub_process1();`
	`} else {`
	`  sp.sub_process2();`
	`}`
		`void FileManager() {`
		`  bool isRead;`
		`  isRead = true;`
		`  RW(file, isRead);`
		`}`
		`void RW(Char* file,`
		`  bool isRead) {`
		`  if (isRead) {`
		`  printf("read");`
		`  } else {`
		`  printf("write");`
		`  }`
		`}`

	`Dto dto = new Dto();`
	
	`send(dto);`
	
	`dto = receive();`
		`void Read_and_Print() {`
		`  File *fp`
		`  char arr[50];`
		`  fp=fopen("t.txt", "r");`
		`  fprintf(fp, "ccc");`
		`}`

	`float circleArea(r) {`
	`  return r*r*PI.get();`
	`}`
	`float shpereArea(r) {`
	`  return 4*r*r*PI.get();`
	`}`
	`class PI() {`
	`  float PI=3.14;`
	`  float get(){return PI;}`
	`}`
		`char Character;`
		`void Read_and_Print() {`
		`  Read_Char();`
		`  Print_Char()`
		`}`
		`void Read_Char() {`
		`  Character=getchar();`
		`}`
		`void Print_Char() {`
		`  putchar(Character);`
		`}`

	`Circle c = new Circle();`
	`c.r = 5.0;`
	`class Circle() {`
	`  float r;`
	`  ...`
	`}`
		-

(나) 응집도 코드 패턴

Java	C

`class Sender() {`
`  void send(String msg);`
`}`
	`int Sum(int x, int y) {`
	`  return x + y;`
	`}`

`class Server() {`
`  int rat;`
`  int receive();`
`  void send(int rat);`
`}`
	`void ProcessMatrix() {`
	`  long aM[2][2], iM[2][2];`
	`  int d = aM[0][0]*aM[1][1]`
	`  -aM[0][1]*aM[1][0];`
	`  for(int i=0; i<2; i++)`
	`  for(int j=0; j<2; j++) {`
	`  if(i == j)`
	`  iM[i][j]=M[1-i][1-j]/d;`
	`  else`
	`  iM[i][j]=-aM[i][j]/d;`
	`  }`
	`}`

	`class DB() {`
	`  int n;`
	`  void insert(n);`
	`  void delete(n);`
	`}`
		`void Calculation() {`
		`  int x, y, r;`
		`  r = x + y;`
		`  r = x * y;`
		`}`

	`class Initializer() {`
	`  int data;`
	`  void init();`
	`  void clearData();`
	`}`
		`void InitVar() {`
		`  no_student = 0;`
		`  no_department = 0;`
		`  u_name = "Hongik Univ.";`
		`}`

	`class Figure() {`
	`  float getArea();`
	`  float getVolume();`
	`}`
		`void Degree(int point) {`
		`  switch(point/10) {`
		`  case 10: case 9: case 8:`
		`  printf("pass");break;`
		`  default:`
		`  printf("fail");break;`
		`}`

	`class Util() {`
	`  float getAbs(float f);`
	`  boolean initDB();`
	`  void printLog();`
	`}`
		`void Calculation() {`
		`  int b = 1, c = 2;`
		`  int a = b*c+10;`
		`  int y = a+b;`
		`}`

다르기 때문에 이에 맞는 코드 패턴이 필요하다. 예를 들어 Java에서는 클래스를 모듈 단위로 정의하였기 때문에 클래스와 클래스의 연관 관계에 따라 결합도가 결정된다. 반면에 C에서는 메소드와 메소드의 연관 관계에 따라 결합도가 결정된다. Java에서 응집도는 클래스와 메소드 사이의 관계에 따라 결정되며, C에서는 메소드 내의 프로세스 절차에 따라 결정된다. 이와 같은 차이점에 유의하며 코드 패턴을 정의한다.

표 1은 각각 Java와 C에서의 결합도, 응집도 코드 패턴을 나타낸다. 예를 들어, 시간적 응집도에서 Java는 Initializer 클래스가 가지는 init(), clearData() 메소드가 시간적인 관계를 갖고 있음을 나타낸다. 반면에 C는 InitVar() 함수 내의 프로세스가 시간적인 관계를 갖는다.

3. 결 론

각 언어에 따라 기본 모듈 단위 정의와 응집도, 결합도 점수화로 품질지표를 정의 하였다. 기존 C 프로그램 코드와 객체지향 Java 코드에서 응집도와 결합도에 대한 코드 패턴을 분석 하였다. 이는 기본적인 개발자의 나쁜 코드 습관을 가시화하여, 좋은 습관화로 유도 가능하다. 향후 SW Visualization 의 툴 체인과 적용 타깃에 해당 지표를 적용할 예정이다. 그리고 더욱 SW를 고품질화 하기 위해서 결합도와 응집도 이외의 또 다른 품질지표를 연구할 예정이다.

참 고 문 헌

[1] NIPA SW공학센터, "2013 SW 공학 백서", 2013. 4.
[2] NIPA SW공학센터, "SW개발 품질관리 매뉴얼(SW Visualization)", 2013. 12.
[3] 이종석, 우치수, "객체지향 시스템에서의 클래스 응집도와 결합도 메트릭", 정보과학회논문지:소프트웨어 및 응용, 제27권, 제6호, 2000