타가시 메소드를 이용한 소프트웨어 변경 영향도 측정 기법 설계

권예진*, 박용범**
*단국대학교 전자계산학과
**단국대학교 컴퓨터공학과
e-mail:{kwon6838*, ybpark**}@dankook.ac.kr

The design of Taguchi method for software change impact measurement techniques

Yejin Kwon*, Youngbom Park**
*Dept of Computer Science, Dankook University
**Dept of Computer Science, Dankook University

요 약

소프트웨어의 잦은 변경은 작은 기능 변경에 한정되어 있다고 하더라도 해당 기능을 수정하고 그로 인해 발생할 수 있는 파급효과에 대해 의도하지 않은 비용이 발생하기도 하며 심지어 소프트웨어에 심각한 영향을 끼치게 된다. 따라서 본 논문에서는 소프트웨어 변경 영향도를 측정하기 위해 동적 코드 분석 방법과 정적 분석 방법을 혼합하여 사용하였다. 특히 객체지향 언어인 JAVA에서 각 객체들 사이의 의존도와 관계를 분석하는 방법과, 실제 프로그램 수행 과정 중에 나타나게 되는 객체들 사이의 의존성을 분석할 수 있는 Taguchi method를 이용한 테스트 케이스 추출 방법을 제안하였다.

1. 서론

소프트웨어가 개발되고 상용화되어 사용되고 있다고 하더라도 실제 소프트웨어를 유지하고 보수하는 데에는 많은 비용이 소모된다. 소프트웨어의 잦은 변경은 작은 기능 변경에 한정되어 있다고 하더라도 해당 기능을 수정하고 그로 인해 발생할 수 있는 파급효과에 대해 의도하지 않은 비용이 발생하기도 하며 심지어 소프트웨어에 심각한 영향을 끼치게 된다[1]. 특히 객체지향 시스템에서는 객체들 사이가 복잡한 관계로 얽혀 있기 때문에 이러한 변경 요구에 대한 영향도 분석이나 변경으로 인한 파급효과를 분석하는 일은 매우 중요하다. 또한 객체지향 소프트웨어에서 복잡한 관계는 변경 영향도를 측정하기 위한 효율적인 테스트 전략을 구성하는데 어려움이 따른다[2]. 따라서 객체지향 소프트웨어 변경 요구에 대한 분석은 소프트웨어를 변경하고 유지하는데 매우 중요한 단계로 활발하게 연구되고 있다.

변경 영향도 분석은 “변화의 잠재적인 결과를 식별하거나 변경 사항을 달성하기 위해 수정해야 하는 니즈를 추정하는 것(Identifying the potential consequences of a change, or estimating what needs to be modified to accomplish a change)”을 의미한다[3]. 소프트웨어의 변경 영향도 분석 방법은 의존성 분석(dependency analysis)과 추적가능성 분석(traceability analysis)으로 분류된다. 의존성 분석은 프로그램 구성 요소들 사이의 의존성 관계들을 자세하게 검사하는 것을 의미하며 소스 코드 단위의 낮은 수준에서 자세한 의존성 분석을 수행한다. 추적가능성 분석은 코드보다 상위 수준인 요구사항이나 소프트웨어 설계, 아키텍처 등에서 의존성 분석을 수행한다[4, 5]. 본 연구에서는 소스코드 단위에서 실제 프로그램 수행 과정중에 나타나게 되는 의존성 분석을 위해 타가시 메소드를 이용한 변경 영향도 분석 방법을 제안하였다.

소스 코드 수준에서 변경 영향도를 분석하는 방법은 다양하게 연구되어 왔다. 대표적으로 콜 그래프를 이용하여 변화에 미치는 영향을 예측하는 방법이 있으며, 이는 상대적으로 저렴하지만 그 예측 결과가 부정확할 수 있다[5]. 정적 분할(Static slicing) 방법은 소프트웨어의 안전한 변경 영향도를 예측할 수 있는 방법이지만 모든 프로그램의 동작에 집중할 경우 시스템 운영의 예상 프로필이 너무 방대해 져서 부정확한 예측을 할 가능성이 있다[6]. 동적 분할(Dynamic slicing) 방법은 특정 프로그램이 실행 될 때 운영 프로파일을 이용하여 분석하며, 많은 유지 보수 작업에 유용하게 쓰일 수 있다. 하지만 결과 영향 평가에서 안전성 보장이 어렵다. 정적 및 동적 분할(Static and Dynamic slicing) 기법은 데이터 위존성, 컨트롤 의존성, 에일리어스(alias) 의존성 등을 분석해야 하기 때문에 콜 그래프를 기반으로 하는 분석 방법에 비해 상대적으로 많은 비용이 발생한다[7].

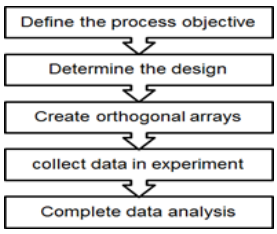
본 논문에서는 소프트웨어 변경 영향도를 측정하기 위해 동적 코드 분석 방법을 기반으로 하여 변경 영향도 분석 과정을 설계하였다. 특히 객체지향 시스템에서 실제 프로그램 수행 과정 중에 나타나게 되는 객체들 사이의 의존성을 분석할 수 있는 Taguchi method를 이용한 테스트

케이스 추출 방법을 제안하였다.

2. 관련 연구

2.1. 타가시 메소드(Taguchi Method)

Taguchi Method는 실제 실험의 견고한 설계를 통해 과정의 변화를 측정하고 그 측정 결과를 통해 변화를 감소할 수 있도록 하는 방법이다. Taguchi Method의 목적은 저렴한 비용으로 고품질의 결과를 생산하게 만드는 것에 있다. Taguchi Method는 Genichi Taguchi에 의해 개발되었으며, 서로 다른 매개 변수를 통해 프로세스가 얼마나 잘 작동하는지 성능의 특성을 평균과 분산에 미치는 영향에 대해 조사하기 위해 실험을 설계하는 방법을 제공한다. Taguchi method에서는 직교 배열(orthogonal array)의 수학적인 방법과 신호 대 잡음비(signal to noise ratio)라는 품질 평가 개념을 이용하여 다양한 사용환경에서 원하는 품질을 유지할 수 있는 제품이나 생상 공정을 적은 실험 횟수를 통하여 설계할 수 있다. Taguchi method는 1980 대 초에 미국에 도입되어 미국 자동차 업계의 품질 관리와 경쟁력 강화에 지대한 기여를 하였다[8]. Peace가 제안한 Taguchi method의 실험 절차는 아래의 (그림 1)과 같다.



(그림 1) Taguchi method process

Taguchi method의 실험 절차는 첫 번째로 프로세스의 목표를 정하고 목표에 맞는 측정 계획을 세우게 된다. 또한 실험 설계에서 설계 매개 변수를 확인하고 매개변수는 쉽게 제어할 수 있는 성능에 영향을 미치는 변수로 정의한다. 그 후에 정의된 설계에 따라 직교 배열을 만든다. 만들어진 직교 배열을 가지고 성능 측정에 영향을 미치는 데이터를 수집하기 위해 완성된 배열에 있는 실험을 수행하고 전체 데이터를 분석하게 된다[9]. 간단한 Taguchi method 수행 절차는 그림 1과 같이 나올 수 있으며 실험 결과나 분석 단계를 추가하여 각 데이터에 맞는 실험을 설계할 수 있다.

2.2. 변경 영향도 분석(Change Impact Analysis)

변경 영향도 분석은 “변화의 잠재적인 결과를 식별하거나 변경 사항을 달성하기 위해 수정해야하는 니즈를 추정하는 것(Identifying the potential consequences of a change, or estimating what needs to be modified to accomplish a change)”으로 정의된다[3]. 변경 영향도 분석은 하위 수준인 코드 부분에서 변경 요구에 의한 영향도를 측정하는 코드 분석 방법과 요구사항이나 아키텍처와 같은 상위 레벨에 따라 분석하는 방법이 있다. 최근 한 연구에 따른 변경 영향도 분석의 연구 분야를 나눈 그림은 아래의 그림 2와 같다[10].

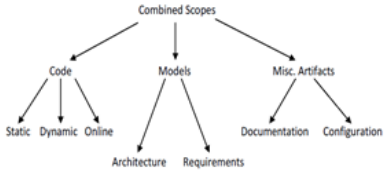


그림 2. Scopes of impact analysis

프로그램의 소스코드를 기반으로 하는 변경 영향도 분석은 정적 분석과 동적 분석, 온라인 분석으로 나눠지며, 하위 수준에서 실제 프로그램의 동작 과정이나 메소드 콜 관계나 메모리 추적과 같은 세부적인 영향도 분석이 가능하다. 모델을 기반으로 하는 영향도 분석은 아키텍처와 요구사항 분석으로 구분되고, 상위 수준에서 소프트웨어 개발 프로세스 전반에 걸친 분석이 가능하다. 그 외에 소프트웨어 개발 과정에 나타나게 되는 문서를 기반으로 하는 영향도 분석이나 구성 요소들 사이의 연관 관계를 기반으로 분석하는 기법 등이 있다.

3. Taguchi method를 이용한 영향도 분석

3.1. 분석 범위 정의

프로그램의 소스코드를 기반으로 하는 변경 영향도 분석은 정적 분석과 동적 분석, 온라인 분석으로 나눠지며, 하위 수준에서 실제 프로그램의 동작 과정이나 메소드 콜 관계나 메모리 추적과 같은 세부적인 영향도 분석이 가능하다. 모델을 기반으로 하는 영향도 분석은 아키텍처와 요구사항 분석으로 구분되고, 상위 수준에서 소프트웨어 개발 프로세스 전반에 걸친 분석이 가능하다. 그 외에 소프트웨어 개발 과정에 나타나게 되는 문서를 기반으로 하는 영향도 분석이나 구성 요소들 사이의 연관 관계를 기반으로 분석하는 기법 등이 있다.

3.2. 분할 동적 측정

Taguchi method를 이용한 변경 영향도 분석을 하기 위해서는 Taguchi method에 따른 파라미터와 레벨이 필요하다. 따라서 본 논문에서는 실제 객체지향 프로그램에서 메소드를 변경의 기준 단위로 보고 동적 분석을 통해 파라미터 설정과 레벨의 값을 토대로 테스트 케이스를 추출하고자 한다. 먼저 메소드 단위의 동적 측정 방법을 그림으로 표현하면 아래의 (그림 3)과 같다.

(그림 3) 분할 동적 측정 방법

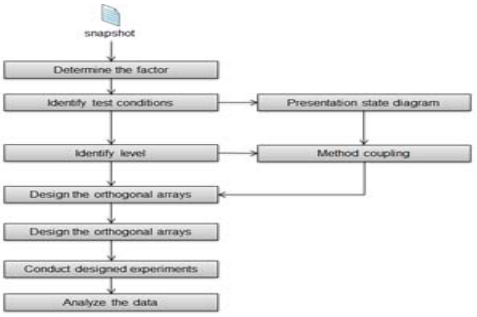
Taguchi method를 이용한 변경 영향도 분석을 하기 위해서는 Taguchi method에 따른 파라미터와 레벨이 필요하다. 따라서 본 논문에서는 실제 객체지향 프로그램에서 메소드를 변경의 기준 단위로 보고 동적 분석을 통해 파라미터 설정과 레벨의 값을 토대로 테스트 케이스를 추출하고자 한다. 먼저 메소드 단위의 동적 측정 방법을 그림으로 표현하면 아래의 (그림 3)과 같다.

객체지향 프로그램에서 프로그램이 동작 과정 중에 객체가 생성되고 객체들 간의 메소드가 수행되는 과정을 그림으로 간단하게 표현하면 그림 3과 같다. 기존의 동적 측정 방법은 전체 시스템에서 메모리 사용이나 리소스 추적을 하며 많은 비용을 발생시키고, 객체지향 시스템에서 객체들간의 관계를 정확하게 예측하기 어렵다. 이에 따라 본 연구에서는 실제 변경이 요구되는 메소드 'mb3'가 존재한다면 'mb3'가 속한 객체가 생성되는 시점과 'mb3' 메소드가 끝나고 값을 반환하는 시점의 리소스들의 스냅샷을 이용하여 영향도 분석을 하고자 한다. 전체 시스템이 동작하는 모든 과정을 추적하게 되면 많은 비용이 소모되고, 메소드 호출 관계가 복잡해져 실제 어떤 메소드가 호출되고 종료되었는지를 확인하기가 어려워 진다. 따라서 메소드가 동작하기 위한 어트리뷰트 값이 생성되는 시점과 메소드가 끝나고 종료되는 시점으로 분할하여 해당 부분에서 생성되어 있는 객체들 간의 관계를 통해 영향도를 분석한다. 객체가 생성되는 시점의 파라미터의 값들과 객체가 생성된 시점에 존재하는 다른 객체들의 리소스 정보와 변경이 요구되는 메소드가 끝나는 시점의 반환되는 값을 Taguchi method의 입력 파라미터 값과 결과 값으로 지정하여 직교 배열을 생성하고 해당 직교 배열에 따른 테스트 케이스를 추출하고, 영향도 분석을 한다.

3.3. Taguchi Method를 이용한 영향도 분석 설계

Taguchi method를 이용한 변경 영향도 을 위한 전체 프로세스를 표현하면 그림 4와 같다.

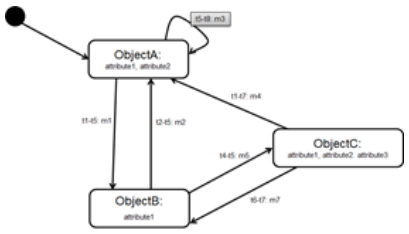
특정 메소드의 변경 요청이 있을 경우 해당 하는 메소드가 속한 객체의 생성 시점과 메소드가 끝나는 시점의 스냅샷을 통해 주요 요소를 정의하고 테스트를 위한 조건들을 정의한다. 이 때 스냅샷을 찍은 구간 동안의 메소드들의 관계와 값을 스테이스 다이어그램으로 표현하고, Taguchi method의 설정을 위한 레벨을 정의하기 위해 메소드들 간의 콜 관계를 정의할 수 있는 메소드 커플링을 이용한다. 파라미터와 레벨의 값을 정의하고 나서 Taguchi method에서 제공하는 배열 선택기에 따라 직교 배열을 선정하고 설계한다. 해당 직교 배열은 각각의 테스트 케이스로 도출 될 수 있으며, 테스트 케이스를 통해 해당 부분에 대한 테스트를 수행한다. 또한 결과로 도출된 결과 데이터를 분석하여 최종적인 영향도 분석을 수행할 수 있게 된다.



(그림 4) Taguchi method를 이용한 영향도 분석 설계

4. State Diagram을 이용한 Taguchi method condition 설정

동적 측정을 위해 전체 프로그램을 세세하게 감시하지 않고 일부 변경이 필요한 메소드를 위주로 분할하여 동적 측정을 하였기 때문에 분석해야 하는 부분은 감소하게 되었다. 따라서 해당 동적 측정을 한 결과를 토대로 변경이 필요한 메소드가 속한 객체의 생성 시점에서부터 메소드가 끝나는 시점까지의 state diagam을 통해 Taguchi method의 condition을 설정하게 된다.



(그림 5) 분할된 프로그램의 state diagram

예를 들어 그림 5와 같이 프로그램의 분할된 일부분만을 통해 state diagram을 추출하였다고 가정하였을 경우 실제 변경해야 하는 부분인 메소드 'm3'를 Taguchi method를 통해 method condition을 설정 할 경우 아래의 <표 1>과 같이 나타나게 된다.

<표 1> Taguchi method condition set

parameter	attribute1:ObjectA attrebute2:ObjectA
level	the number of method that using attribute1, the number of method that using attribute2

<표 1>과 같이 파라미터와 레벨이 정의되면 Taguchi metho를 이용한 직교 배열 설정이 가능하다. Taguchi method 직교 배열 선택기에 따라 실험 케이스를 추출할 수 있게 된다.

Taguchi method에서 정의된 직교 배열 선택기에 따라 테스트 케이스를 추출하며, 추출된 테스트 케이스의 종합적

인 결과로 인해 변경이 필요한 메소드의 영향도를 분석할 수 있게 된다.

5. 결론

지금까지 Taguchi method를 이용한 영향도 분석 방법에 대해 설계하고 세부 과정을 서술하였다. 기존의 영향도 분석에서 수행되었던 기법과는 달리 전체 시스템이 아닌 실제 변경이 필요한 최소 기준 단위인 메소드의 수행 과정에 나타나게 되는 단계를 State diagram으로 표현하고, 각 상태에 따른 입력과 결과 값을 통해 Taguchi method의 직교 배열을 만들어 테스트 케이스를 추출하는 과정을 설계하였다. 본 연구에서는 실제 동적 측정 과정에서 필요한 리소스를 최대한 줄이고, 전체 시스템의 동작 과정에서 나타나는 메소드 콜 관계나 객체들 사이의 관계의 복잡도를 줄이고자 변경이 필요한 메소드가 속한 객체의 생성 시기와 메소드가 반환되는 시점을 기준으로 하여 분할하고 분할된 부분만을 분석하였다. 또한 정적 분석을 통해 나타낼 수 없는 실제 시스템의 동작 과정에서의 콜 관계들을 시각화하고 분석이 용이하도록 설계하였다.

향후 연구로는 실제 객체지향적으로 프로그래밍 된 소프트웨어의 분할 동적 측정을 통해 전체 시스템이 아닌 일부분의 영향도 분석을 수행하고, Taguchi method를 이용한 테스트 케이스 추출과 실험을 통해 결과의 분석이 필요하다. 또한 전체 시스템의 동작 과정이 아닌 변경이 필요한 메소드를 기준으로 일부만을 측정한 결과이기 때문에 정밀한 증명과 전체 시스템 영향도의 비교 분석이 필요하다.

참고문헌

[1] J. L. Lions. "ARIANE 5, Flight 501 Failure, Report by the Inquiry Board. European Space Agency", July 1996.

[2] Michelle L. Lee(Li Li), “CHANGE IMPACT ANALYSIS OF OBJECT-ORIENTED SOFTWARE”, Fall Semester 1998.

[3] S. A. Bohner and R. S. Arnold, “Software Change Impact Analysis.Los Alamitos”, CA, USA: IEEE Computer Society Publications Tutorial Series, 1996.

[4] Shawn A. Bohner and Robert S. Arnold, “An Introduction to Software Change Impact Analysis”, IEEE Computer Society Press

[5] S. Bohner and R. Arnold. “Software Change Impact Analysis”. IEEE Computer Society Press, Los Alamitos, CA, USA, 1996.

[6] F. Tip. "A survey of program slicing techniques." Journal of Programming Languages, 3:121-189, 1995.

[7] James Las, Gregg Rothermel, “Whole Program Path-Based Dynamic Impact Analysis”, Proceedings of the 25th International Conference on Software Engineering(ICSE), 2003.

[8] Zalewski, “Design of experiments via taguchi methods: orthogonal arrays”, The Michigan Chemical Process Dynamics and Controls Open Text Book, 2006.

[9] Peace, Glen Stuart, "Taguchi methods: A hands-on approach", Addison-Wesley (Reading, Mass.), 1993.

[10] Steffen Lehnert, "A Review of software change impact analysis", Technische Universitat Ilmenau Ilmenau, 2011