디자인 패턴 기반 소프트웨어의 테스트 가능성 분석

강영남° 최은만
동국대학교 컴퓨터멀티미디어공학과
{copyrei°, emchoi}@dgu.edu

Analysis for Testability of Software based on Design Pattern

Young Nam Kang° Eun Man Choi
Dept. of Computer Engineering, Dongguk University

요 약

잘 설계된 모든 객체지향 구조들은 패턴들로 가득 차 있다는 점에서 볼 때, 디자인 패턴은 상당히 유용하다. 특히 정확성, 강건성, 유연성, 재사용성, 효율성 측면에서 볼 때, 디자인 패턴은 충분히 가치가 있다. 이 논문에서는 디자인 패턴을 사용한 소프트웨어에서 테스트 가능성은 어떻게 달라지는지를 분석하고자 한다. 테스트 가능성을 측정하는 메트릭을 이용하여, 패턴이 적용된 소프트웨어와 적용되지 않은 소프트웨어에서의 메트릭을 분석한다. 측정된 값은 디자인 패턴을 사용하지 않은 소프트웨어에 비해, 사용한 소프트웨어에서 몇몇 메트릭이 낮은 값을 보였다. 이것은 디자인 패턴을 적용하는 것이 오류의 가능성이나 테스트 케이스의 수를 줄여 준다는 것을 의미한다. 또한 어떤 디자인 패턴이 적용되었는지를 알고 있을 때, 그 디자인 패턴에 맞는 테스트 케이스가 무엇인지 분석하였다.

1. 서 론

통계와 경험에 의하면 테스트는 소프트웨어를 개발하는 과정 중에서 시간과 비용이 가장 많이 드는 작업이다. 테스트 케이스 설계, 테스트 환경 구축, 결함의 위치를 파악하고 수정하기 위해서는 많은 시간과 비용이 필요하다. 이 같은 문제로 인하여 테스트 작업을 완벽하게 할 수는 없다[9].

소프트웨어 테스트는 매우 어려운 작업이기 때문에 그것을 합리적으로 할 수 있는 방법이 무엇인지 알아야 한다. 여기에 소프트웨어를 얼마나 쉽게 테스트할 수 있는지를 나타내는 테스트 가능성은 필수적으로 체크해야할 요소이다.

이제까지 테스트 가능성 분야의 연구는 data flow analysis를 사용한 소프트웨어 테스트 가능성을 측정한 연구[10], Observerability와 Controllability 개념을 적용하여 도메인 테스트 가능성의 연구[2] 등의 주로 테스트 가능성을 위한 디자인과 측정에 관한 것들이다.

따라서 이 논문에서는 디자인 패턴을 사용한 소프트웨어에서 테스트 가능성이 어떤 영향을 받는지 테스트 가능성 메트릭과 블랙 박스 기반 테스트 가능성을 통한 분석을 제시한다.

2. 관련 연구

2.1 테스트 가능성

IEEE에서 발행한 소프트웨어 관련 용어집에는 테스트 가능성(testability)을 "시스템 또는 컴포넌트가 테스트 기준을 확립하고 테스트를 쉽게 할 수 있는 정도" 그리고 "요구 사항에서 테스트 기준의 확립을 위해 사용된 용어의 정도"로 정의된다[3].

이상적인 환경에서 소프트웨어 엔지니어는 항상 테스트 가능성을 갖고 시스템을 설계해야 하며, 이것은 보다 쉽게 효과적인 테스트 케이스를 설계하도록 할 수 있다. James Bach는 다음과 같은 방법으로 테스트 가능성을 설명하였다. 소프트웨어 테스트 가능성은 컴퓨터 프로그램이 어떻게 쉽게 테스트되어지는지를 나타낸다. 테스트 작업은 매우 어렵기 때문에 그것을 합리적으로 행할 수 있는 방법이 무엇인지를 알아야 하며, 프로그래머들은 테스트 프로세스를 성공적으로 수행할 수 있는지 리스크 분석 차원에서 미리 분석할 필요가 있다.

여러 가지 다양한 측면에서 테스트 가능성을 측정하는데 사용할 수 있는 척도가 있으며, 테스트 가능성은 테스트 특정 집합이 적절하게 제품의 품질을 체크하고 결함을 식별할 수 있는지를 나타내는데 자주 사용된다. 그러나 이 의미는 테스트 가능성과는 차이가 있으며, 아래 표의 체크리스트는 테스트 가능한 소프트웨어를 이끌어내는 특성들의 집합을 제공해 준다.

표 1 James Bach가 제시한 테스트 가능성 요소[4]

제어성 (Controllability)	제어를 잘 하면, 테스팅은 자동화되고 최적화 될 수 있다.
관찰성 (Observability)	관찰하고 보는 것이 테스트될 수 있는 것이다.
이용성 (Availability)	테스트하기 위해서는 시스템을 이용해야 한다.
단순성 (Simplicity)	간단할수록, 테스트 작업이 쉽다.
안정성 (Stability)	변화가 적을수록, 테스팅에 혼란이 없다.
정보 (Information)	갖고 있는 정보가 많을수록, 빈틈없이 테스트 할 수 있다.

James Bach가 제안한 속성들은 테스팅이 가능한 소

프트웨어를 개발하는 소프트웨어 엔지니어가 사용할 수 있다.

쉽게 테스트 할 수 있다는 것은 테스트 집합이 작고 쉽게 생성되어야 하고, 테스트 집합이 중복되지 않아야 한다. 그리고 테스트 결과가 쉽게 해석되어야 하고 소프트웨어 오류는 쉽게 추적 가능해야 한다[2].

아래의 표는 Robert V. Binder가 제안한 6가지 테스트 가능성 요소[1] 중에서 구조적 테스트 가능성을 측정하기 위한 메트릭이다[7,8]. 이 값들이 높으면 테스트 케이스가 많아지거나 결함이 있을 확률이 높기 때문에, 그 소프트웨어가 테스트 가능성이 낮다는 것을 의미한다.

표 2 테스트 가능성 측정 메트릭

Encapsulation metric	
LCOM	하나의 메소드에 의해서만 사용되는 인스턴스 변수의 그룹의 수
PAP	public, protected 클래스에서 데이터 멤버의 비율
PAD	외부에서 public, protected 데이터 멤버로 접근하는 수
Inheritance metric	
NOR	프로그램에 의해 사용되는 구별되는 클래스의 수
FIN	부모 클래스의 수
NOC	자식 클래스의 수
DIT	상속 트리의 깊이
Polymorphism metric	
OVR	오버로드되지 않은 호출의 비율
Complexity metric	
WMC	클래스 안에서 구현된 메소드의 순환 복잡도의 합
RFC	클래스 안에 구현된 메소드의 수와 상속되지 않은 객체에 의해 사용된 메소드의 수의 합
CBO	한 클래스에서 다른 클래스의 메소드나 데이터 멤버에 접근하는 수

2.2 디자인 패턴

객체 지향 소프트웨어를 설계하는 것은 어렵다. 그리고 재사용성 있는 객체 지향 소프트웨어를 개발하는 것은 더욱 어렵다. 디자인 패턴은 "특정한 상황에서 일반적 설계 문제를 해결하기 위해 상호 교류하는 수정 가능한 객체와 클래스들"이라고 할 수 있다. 디자인 패턴을 이용하면 좋은 설계나 아키텍쳐를 재사용하기 쉬워진다[6].

2.2.1 Command 패턴

행위 패턴중 하나인 Command 패턴은 요청받은 오퍼레이션이 무엇이며, 이를 처리할 객체가 누구인지에 대한 아무런 정보 없이도 임의의 객체에게 메시지를 보낼 때 사용한다. 즉, 수행할 행위 자체를 파라미터화하고자 할 때 사용한다. 명령어 객체 자체에 실행 취소에 필요한 상태를 저장할 수 있기 때문에 수행과 취소를 무한 반복할 수 있다.

3. 사례 연구

보통 재사용을 위해 사용되는 디자인 패턴이 테스트 가능성에 어떤 영향을 주는지 알아보기 위해 임의의 패턴을 적용하여 실험해 본다.

똑같은 기능의 WordProcessor를 Command 패턴을 적용한 프로그램[5]과 패턴을 적용하지 않고 구현한 프로그램을 비교한다.

그림1은 Command패턴을 적용한 것으로 Client라고 볼 수 있는 WordProcessor에서 Command 인터페이스를 통하여 CutCommand, PasteCommand등의 클래스로 제어가 넘어가서 Client가 원하는 기능을 하게 된다. 패턴을 적용하지 않은 그림2는 WordProcessor에서 직접 모든 기능을 제어한다.

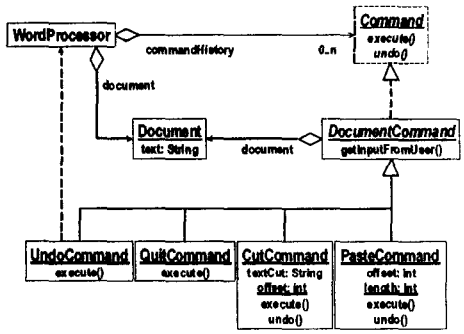

그림 1 Command 패턴을 적용한 WordProcessor

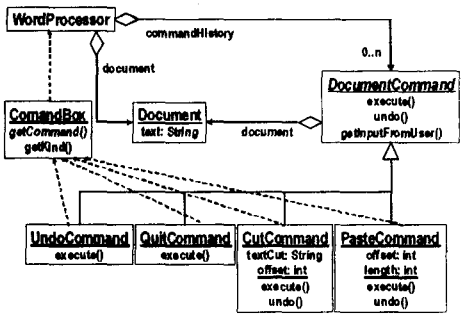

그림 2 패턴을 적용하지 않은 WordProcessor

3.1 테스트 가능성 메트릭 측정

위 예제를 이용하여 패턴이 적용된 프로그램과 그렇지 않은 프로그램의 테스트 가능성 메트릭을 측정한 결과이다.

표 3 메트릭 측정 결과

메트릭	패턴 적용	패턴 미적용
Encapsulation metric		
LCOM	3	4
PAP	0%	17%
PAD	0%	0%
Inheritance metric		
NOR	1	1
FIN	1	1
NOC	4	4
DIT	1	1
Polymorphism metric		
OVR	0%	7%
Complexity metric		
WMC	6	12
RFC	2	5
CBO	4	4

결과를 보면 패턴 미적용 시, 다른 메트릭에 비해 PAP, OVR, WMC, RFC가 많이 증가했다는 것을 알 수 있다. 이것은 패턴을 사용하지 않은 프로그램에서 부작용 또는 결함이 발생할 확률이 높고 더 많은 테스트 케이스를 필요로 한다는 것을 말해준다.

3.2 블랙박스 기반 테스트 가능성

각각의 디자인 패턴에 따라서 블랙박스 테스트 시, 유의해서 테스트해야 할 부분들이 있다. 즉, 블랙박스 테스트 시에는 어떤 패턴이 적용되었는지 알고 있다는 가정하에 테스트 가능성을 분석한다.

위 사례에서 사용된 Command 패턴은 행동의 실행과 취소를 구현하기 위해 많이 사용한다. 표 4와 같이 취소 프로세스 내에 오류가 누적될 가능성이 있으므로 이 부분을 테스트해야만 한다. 그리고 실행과 취소를 구현하기 위해 스택을 사용하는데 스택을 사용할 때 자주 발생하는 모든 오류를 체크해 봐야 할 것이다. 그림 3과 같이 현재 스택에 들어있는 오퍼레이션이 없을 경우, java.lang.NullPointerException과 같은 심각한 오류가 발생할 수 있다.

```
Pick from one of the following:
quit
undo
cut
paste

input command.
undo
java.lang.NullPointerException
        at UndoCommand.execute(UndoCommand.java:32)
        at WordProcessor.main(WordProcessor.java:88)
Exception in thread "main"
```

그림 3 Exception 발생

표 4 Undo 메소드 안의 오류

명령	인덱스	입력 문자열	저장된 문자열
paste	0	kyn	kyn
paste	3	storm	kynstorm
paste	5	pppp	kynstpppporm
undo			kynstorm
undo			kynst
undo			kynst
undo			cannot undo!!

4. 결론 및 향후 연구

디자인 패턴을 이용하여 재사용 가능한 객체 지향 소프트웨어를 개발할 수 있다. 디자인 패턴이 테스트 가능성에 어떤 영향을 주는지 알아보기 위해 Command 패턴을 사용하여 테스트 가능성 메트릭을 측정했다. 그리고 블랙박스 테스트 시, 어떤 패턴이 사용되었는지를 추측하여 그에 맞는 테스트 케이스를 생성했다. 위 실험 결과로 미루어보아, 디자인 패턴이 테스트 가능성을 향상시킨다는 결론을 내릴 수 있다.

향후 연구 과제로는 좀 더 다양한 디자인 패턴을 가지고 화이트 박스, 블랙박스 기반의 테스트 케이스를 생성하고, 각 패턴에 따른 테스트 케이스 생성 방법상의 차이점을 분석하는 연구가 필요하겠다.

5. 참고 문헌

[1] R. V. Binder, "Design for Testability in Object-Oriented Systems", *Communications of the ACM*, Vol. 37, No. 9, pp. 87-101, 1994.

[2] Roy S. Freedman, "Testability of Software Components", *IEEE Trans. on Software Engineering*, 1991.

[3] IEEE Std 610. 12-1990, *IEEE Standard Glossary of Software Engineering Terminology*, 1990.

[4] *Heuristics of Software Testability* by James Bach, Satisfice, Inc.

[5] Eric Braude, *Software Design: From Programming to Architecture*, Wiley&Sons, 2004.

[6] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, Mass. Addison-Wesley, 1995.

[7] Chidamber, S.R. and Kemerer, C.F. "A metrics suite for object-oriented design", *IEEE Trans. on Soft. Eng.* Vol. 20, No.6, pp.476-493, 1994.

[8] "OO tools aids software testing", *The Outlook*. Fall 1993, McCabe & Associates, Columbia, Maryland.

[9] 소프트웨어 테스트 전문 기술, 한국 정보통신기술협회, 2003.

[10] Pu-Lin Yeh; Jin-Cherng Lin, "Software testability measurements derived from data flow analysis", *Proceedings of the Second Euromicro Conference*, pp. 8-11, March 1998.