디자인패턴과 안티패턴을 이용한 품질 특성 분류에 관한 연구

김상영[0*] 황선명*, 김재웅,** 노병규***, 조규민***
대전대학교 컴퓨터공학과*, 공주대학교 멀티미디어정보·영상공학부**, 한국정보보호진흥원***
jayusop@zeus.dju.ac.kr[0]

A Study On Quality Characteristics Classification using Design Pattern and Anti Pattern

Sangyoung Kim[0*] Sunmyung Hwang*, Jaewoong Kim**, Byunggu Ro***, Gumin Cho***
Dept. of Computer Engineering, Daejeon University*
Dept. of Computer Multimedia, Kongju National University**
Korea Information Security Agency***

요 약

디자인 패턴과 안티 패턴의 원리는 혼합되어 사용되어진다. 디자인 패턴은 소프트웨어 품질에 대하여 공격적인 예방치료에 목적을 두며, 안티 패턴은 품질 이슈에 대하여 체계적인 진단을 수행하는 것이다. 품질에 대한 국제 표준으로는 ISO/IEC 9126, ISO/IEC 12119등이 있다. 최근에는 이러한 품질에 관련된 문제점들을 해결하기 위하여 디자인 패턴과 안티 패턴에 대한 연구과 활발히 이루어지고 있다.

본 논문에서는 현재 대표적으로 사용되어지는 디자인 패턴과 안티 패턴을 ISO/IEC 9126의 품질 특성별로 분류하고 이들간의 상호 연관성에 대하여 연구하였다.

1. 서 론

객체지향 소프트웨어를 설계하는 것은 어렵다. 그리고 재사용성 있는 객체 지향 소프트웨어를 개발하는 것은 더 어렵다. 클래스 인터페이스와 상속계층을 정의하고 그들 사이의 주요한 관계를 만드는 올바른 모양으로 클래스에서 적절한 객체와 인자를 찾아야 하는 어려움이 있다. 그렇기 때문에 설계를 하였다고 해도 나중에 직접적으로 또는 간접적으로 문제를 발생하거나 수정 보완을 해야 하는 일이 발생한다. 이러한 문제점을 해결하기 위하여 디자인 패턴과 안티 패턴은 중요한 역할을 한다.

본 논문에서는 ISO/IEC 9126의 품질특성별로 대표적인 디자인 패턴과 안티 패턴을 맵핑함으로서 패턴들이 품질에 끼치는 영향을 연구한다.

2. 관련 연구

2.1 디자인 패턴(Design Patterns)

Christopher Alexander는 "각각의 디자인 패턴은 기존환경 내에서 반복적으로 일어나는 문제들을 설명하고, 그 문제들에 대한 해법의 핵심을 설명하는 것이다. 이렇게 하면 똑같은 방법을 두 번 반복하지 않은 채 이 해법을 백만번 이상 사용할 수 있다"라고 하였다[6]. 보통 패턴은 일반적으로 다음의 4가지 요소를 정의한다[1].

- 이름(Name)
  한 두 단어로 설계 문제와 해법을 서술.
- 문제(Problem)
  언제 패턴을 사용하는가를 서술하며 해결할 문제와 그 배경을 설명.
- 해법(Solution)
  설계를 구성하는 요소들과 그 요소들 간의 관계, 책임 그리고 협력관계를 설명.
- 결과(Consequences)
  디자인 패턴을 적용해서 얻는 결과와 장단점을 서술.

디자인 패턴은 주어진 영역에서 반복되는 문제점들을 위한 해법이다. 프로그래머 입장에서 다음과 같은 다양한 이유로 인해 디자인 패턴을 사용한다[4].

- 검증된 디자인 패턴은 위험 요소를 줄여준다
  문제해결을 위해 검증된 청사진을 사용함으로서 성공확률을 증가시킨다.
- 디자인 패턴은 시간과 에너지를 절약시켜 준다
  효과적으로 시간을 사용하고 어려운 문제를 풀기 위해서, 다른 사람이 이미 노력하여 이루어 놓은 결과를 사용할 수 있다.
- 디자인 패턴은 기술과 이해도를 증진시켜 준다
  디자인 패턴을 통하여 문제 영역에 대한 이해도를 증가시키고, 복잡한 모델을 표현하기 위한 새로운 방법들을 배울 수 있다.

디자인 패턴을 받아들인다는 것은 코드를 작성하는 방식을 변화시킨다는 것을 의미하며 이것은 디자인 패턴을 공유하는 디자인 패턴 공동체에 참여한다는 것을 의미한다. 이렇게 함으로서 문제에 대하여 공동으로 문제 해결 방법을 찾을수 있으며 문제 해결을 위한 많은 좋은 소스들을 쉽게 손에 넣을 수 있다.

2.2. 안티 패턴(Anti Patterns)

프로젝트를 성공적으로 수행하기 위한 세가지 원칙은 다음과 같다[4].

- 전문가들이 사용하는 도구와 기법을 학습
  프로그래머로서 문제를 해결하는 최선의 기법들에 관해 배우고, 프로젝트를 개선할 수 있는 새로운 프레임워크와 도구들을 찾기 위해서 많은 컨퍼런스 참가.
- 최고 전문가들의 방식을 따름
  디자인 패턴을 사용함으로서 구조적 문제들을 성공적으로 해결할 수 있는 내용을 얻음.
- 이전의 실수로부터 학습

본 연구는 한국과학재단 목적기초연구 (R01-2001-000-00343-0(2002)) 지원으로 수행되었음.

좋은 계획을 세워도 실패할 수 있으나, 가장 중요한 것은 실패했을 때 어떻게 대처해야 하는지 하는 것.(안티패턴)
프로그래머의 입장에서 개발언어와 상관없이 많은 안티 패턴을 발견할 수 있으며, 대표적인 안티 패턴은 다음과 같다[4].
- 귀여운 축약
코드 사이에 있는 공백을 줄임으로서 코드를 최적화 했다는 생각을 할 수 있으나, 문제의 해결방법을 프로그램에 대하여 단순히 라인수를 줄임으로서 코드의 효율성을 높일 수는 없다.
- 최적화는 가독성의 저하
Crack 프로그래머들에게 매우 중요한 문제점이며, 대부분의 경우 최적화 보다는 코드 가독성이 매우 중요하다. 코드의 성능이 가독성보다 중요한 부분에서는 자세한 주석을 통하여 코드의 이해를 돕는 것이 중요하다.
- Cut & Paste 프로그래밍
프로그래밍에서 가장 범하기 쉬운 오류로서 복사해 붙이기 프로그래밍으로 일단 어떻게든 동작하는 프로그램을 만들어 내기는 쉽지만, 전체 프로그램을 복사하는 것은 매우 어렵다. 또한 복사된 코드를 원본만큼 철저하게 검사하는 오류를 범하기 쉬우며 실제로는 복사해 붙이 프로그램을 원본 프로그램보다 더욱 철저하게 검사해야 한다.
- 잘못된 알고리즘의 사용
같은 문제의 해법에서 결과는 같지만 어떻한 알고리즘을 선택하는냐에 따라 문제점이 발생할 수도 있다. 버블소트가 $O(n^2)$의 수행시간을 갖고 쉘 소트가 O(nlog(n))의 수행 시간을 같는 경우가 이러한 예이다.
- 적절하지 않은 클래스의 사용
객체지향 언어에서는 유사한 기능을 갖는 클래스를 구현하기 위해 테이블 방식을 사용하는 클래스와 배열 타입의 클래스 등 중에 하나를 선택하는 것이 가능하다. 이러한 경우 목적에 따라 매우 다른 수행 속도를 보인다. 데이터 집합에서 임의적인 접근을 목적으로 한다면 테이블을 이용하는 것이 좋으며 순차적으로 접근할 경우에는 배열의 형태가 좋다.

이러한 안티패턴을 연구하기 위한 기법은 다음과 같다.
- 문제 검색
문제점은 버그일 수도 있고, 잘못된 성능을 내는 알고리즘 또는 이해하기 힘든 알고리즘일 수 있다.
- 실패의 패턴 구축
품질제어(Quality Control)는 매우 전문적이고 가치있는 작업이다. 숙련된 엔지니어는 제작 과정을 주의 깊게 살피는 것 만으로 시스템 결함을 찾아냄으로서 개발 비용과 유지보수 비용을 절약할 수 있다. 소프트웨어 개발 과정은 시스템 오류를 발생할 수 있고, 패턴을 사용하는 기술과 연관되어 질 수 있으며 구성원들의 대화에서 문제점을 발견할 수 있다.
- 오류 코드 리팩토링
잘못된 코드를 리팩토링 하는 과정은 반듯이 거쳐야 하며 가능한 경우에는 디자인 패턴을 사용해야 한다.
- 해결방법 발표
다른 사람들에게 안티패턴을 어떻게 인지하였고 또 리팩토링 했는지를 알려야 한다.
- 프로세스 취약점 확인
프레임워크나 개발 도구의 잘못된 사용이 문제를 발생할 수 있다. 이러한 경우 교육을 통하여 해결할 수 있다.
- 프로세스 수정
단순한 문제에 대하여는 문제를 수정하면 되고, 복잡한 경우에는 위험 요소와 문제 해결시의 보상을 분석해 내고 문제를 수정하기 위한 지원을 받을 필요가 있다.

3. 디자인 패턴과 안티 패턴의 ISO 9126과 맵핑

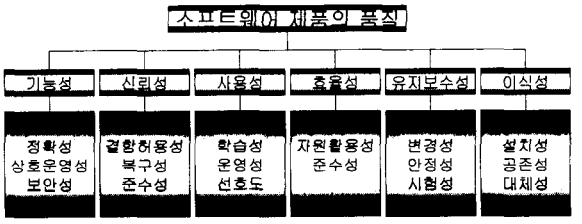
3.1 품질 기술 동향

소프트웨어 품질평가의 핵심 부분은 품질모델, 평가방법, 소프트웨어 측정 도구이다. 좋은 소프트웨어를 개발하기 위해 품질 요구가 명시되어야 하고 소프트웨어 품질보증 프로세스가 계획, 구현 및 통제되어야 한며 중간 및 최종 제품 모두가 평가되어야한다[7][8][10].
객관적인 소프트웨어 품질평가를 달성하기 위해 소프트웨어 품질속성이 확인된 메트릭스를 사용하여 측정되어야 한다 여기서 메트릭스는 측정을 위해 사용될 수 있는 양적인 스케일 및 방법으로 정의된다. 소프트웨어 품질을 평가하기 위해 <표 1>에 제시한 바와 같이 첫째로 평가 요구를 설정하고 다음 단계로 평가를 명세, 설계 및 수행한다. 각 세부 단계마다 국제 표준에서 정의한 품질 특성과 메트릭스를 참조한다.

<표 1> 소프트웨어 품질 평가 단계

구분	평가 프로세스	내 용	관련표준
1단계	평가 요구의 설정	평가의 목적 설정	
		제품의 유형설정	
2단계	평가의 명세	[redacted]	
		메트릭스에 대한 등급	
		평가를 위한 기준	
3단계	평가의 설계	평가 계획 작성	
4단계	평가의 수행	척도의 입수	
		기준치와의 비교	
		품질 모델의 명세	

ISO/IEC 9126-1은 개발자, 구매자, 품질보증 요원 및 독립된 평가자, 특히 소프트웨어 제품의 품질을 명세하고 평가하는 책임을 가진 사람에 의한 이용을 목적으로 하며 품질 메트릭스는 <그림 1>과 같이 분류할 수 있다.



<그림 1> ISO/IEC 9126 품질 메트릭스

3.2 안티패턴과 디자인 패턴의 품질 특성별 분류

Gof의 디자인 패턴에서는 생성패턴과 구조패턴 그리고 행위패턴으로 나누고 각각의 분류별로 총 23개의 디자인 패턴을 정의하는데 이것을 정리하면 <표 2>와 같이 표현할 수 있다[1].
Toshikazu Ando의 안티 패턴에서는 13개의 분류별로 다양한 안티 패턴 분류를 정의하고 있으며 총 33개의 안티 패턴을 정의한다. 이를 정리하면 <표 3>과 같이 표현할 수 있다[5].

<표 2 > Gof의 디자인 패턴

패턴 범주	패 턴
1. 생성패턴 (Creation Patterns)	1.1 Abstract Factory, 1.2 Builder, 1.3 Factory Method, 1.4 Prototype, 1.5 Singletion
2. 구조패턴 (Structural Patterns)	2.1 Adapter, 2.2 Bridge, 2.3 Composite, 2.4 Decorator, 2.5 Facade, 2.6 Flyweight, 2.7 Proxy
3. 행위패턴 (Behavioral Patterns)	3.1 Chain of Responsibility, 3.2 Command, 3.3 Interpreter, 3.4 Iterator, 3.5 Mediator, 3.6 Memento, 3.7 Observer, 3.8 State, 3.9 Strategy, 3.10 Template Method, 3.11 Visitor

<표 3> Toshikazu Ando의 안티 패턴

패턴 범주	패 턴
1.Beginner	1.1 Understand of Design patterns, 1.2 Copy&Paste, 1.3 Repeat-1, 1.4 Repeat-2
2.Share	2.1 Reusability of class and package, 2.2 Redundancy of the name of class
3.Object	3.1 singleton, 3.2 Proxy
4.Operate	4.1 Facsde, 4.2 Understand and Collaboration
5.Array	5.1 Array, 5.2 Command, 5.3 Iterator
6.Safety	6.1 Flyweight, 6.2 Proxy
7.Reuse	7.1 Adapter, 7.2 Bridge, 7.3 Decorator
8.Data	8.1 Memento-1, 8.2 Memento-2, 8.3 Abstract Factory
9.Framework	9.1 Observer-1, 9.2 Observer-2, 9.3 Exception, 9.4 Mediator
10.Action	10.1 Object Create, 10.2 Builder, 10.3 State
11.Tree	11.1 Tree
12.Interpreter	12.1 Interpreter
13.Understand	13.1 Indent, 13.2 Naming Convention, 13.3 etc

ISO/IEC 9126의 부 특성별로 디자인 패턴과 안티패턴을 분류 및 맵핑하여 <표 4> 와 같이 표현한다.

<표 4> ISO/IEC 9126 특성별 패턴 맵핑

주특성	부특성	디자인 패턴	안티 패턴
기능성	적합성	1.1 1.5 2.1 3.3 3.9	2.1 2.2 3.1 4.2 5.2 7.1 8.3 12.1
	정확성	1.1 1.2 1.5 2.1 3.1 3.2	1.3 2.1 2.2 3.1 7.1 8.3 9.3 10.1 10.2
	상 호 운영성	1.3 1.4 2.1 2.3 2.5 2.6 3.3 3.7 3.8 3.10	1.3 1.4 2.1 2.2 4.1 4.2 6.1 7.1 9.1 9.2 10.1 10.3 11.1 12.1 13.2
	보안성	1.5 2.2 2.4 2.7 3.1 3.2 3.4 3.5	3.1 5.2 5.3 6.2 7.2 9.4
신뢰성	성숙성	2.1 2.2 2.4 2.6 3.1 3.8 3.9	1.3 1.4 2.2 6.1 7.1 7.2 9.3 10.1 10.3
	결 함 허용성	1.5 2.1 2.4 2.6 3.1 3.7 3.8 3.11	1.3 2.2 3.1 6.1 7.1 9.1 9.2 9.3 10.1 10.3
	복구성	2.4 3.1 3.2 3.6 3.8	5.2 8.1 8.2 9.3 10.1 10.3
사용성	이해도	1.1 1.5 2.1 2.6 2.7 3.2	1.1 1.4 2.2 3.1 3.2 5.2 6.1 6.2 7.1 8.3 13.1
	학습성	1.1 1.4 2.1 2.6 3.1 3.3 3.7 3.8 3.9 3.11	6.1 7.1 8.3 9.1 9.2 10.1 10.3 12.1
	운영성	1.4 1.5 2.2 2.4 3.1 3.2 3.10 3.11	1.2 2.1 2.2 3.1 4.2 5.1 5.2 6.1 7.2
	선호도	1.2 2.2 2.6 3.11	5.1 7.2 10.1 10.2
효율성	시 간 반응성	1.3 2.3 2.4 2.5 3.1 3.2 3.8	1.3 1.4 4.1 4.2 5.1 5.2 10.1 10.3 11.1
	자 원 활용성	1.1 1.2 1.4 2.1 2.4 2.6 3.3 3.10 3.11	1.1 2.1 4.2 6.1 7.1 8.3 10.1 10.2 12.1
유지 보수성	분석성	2.6 3.1 3.2 3.3	1.1 1.3 1.4 2.2 5.2 9.3
	변경성	1.1 2.1 2.2 3.2 3.6 3.7 3.8 3.9 3.10 3.11	1.2 2.1 2.2 5.2 7.1 7.2 8.1 8.2 8.3 9.1 9.2 10.1 10.3
	안정성	1.5 2.4 2.6 2.7 3.2 3.5 3.8	1.1 1.2 1.3 14 2.1 3.2 5.2 6.1 6.2 9.3 9.4 10.1 10.3
	시험성	1.5 2.4 3.3	9.3 12.1
이식성	적응성	1.1 1.2 1.4 2.1 2.2 2.6 3.1 3.3 3.7 3.9 3.10 3.11	1.3 1.4 2.2 6.1 7.1 7.2 8.3 9.1 9.2 10.1 10.2 12.1
	설치성	1.4 1.5 2.2 2.6 2.7 3.2 3.7 3.8	3.1 3.2 5.2 6.1 6.2 7.2 9.1 9.2 10.1 10.3 13.2
	공존성	1.1 1.2 1.3 2.3 2.5 2.6 3.3 3.10 3.11	2.2 4.1 4.2 6.1 8.3 10.1 10.2 11.1 12.1 13.2
	대체성	1.1 2.6 3.1 3.5 3.9	1.1 1.3 1.4 2.2 6.1 8.3 9.3 9.4

4. 결론

본 논문에서는 디자인 패턴과 안티 패턴에 대하여 알아보았으며, ISO/IEC 9126의 특성별로 분류 및 매핑을 시킴으로서 이러한 패턴들이 어떻한 품질특성과 연관성이 있는지 알아보았다.

향후 연구로는 다양한 디자인 패턴과 안티 패턴을 적용하고 각각의 품질 특성별로 어느 정도의 영향이 있는지에 대한 연구를 할 예정이다.

참고문헌

[1] Erich Gamma, Richard Hel, Ralph Johnson, John Vlssides, "Design Patterns : Elements of Reusable Object-Oriented Software", Addison-Wesley, 2002.

[2] Ed Roman, Floyd Marinescu, "EJB Design Patterns : Advanced Patterns, Processes, and Idioms", John Wiley&Sons, 2002.

[3] Brandon Goldfedder, "The Joy of Patterns : Using Patterns for Enterprise Development", Addison-Wesley, 2002.

[4] Bruce A. Tate, "Bitter Java", Manning, 2002.

[5] Toshikazu Ando, "Anti Design Pattern", Tuttle-mori agency, 2002.

[6] Christopher Alexander, Sara Ishikawa, Murray Silverstein, Max jacobson, Ingrid Fiksdahl-King, and Shlomo Angle, "A Pattern Language", Oxford University Press, NewYork, 1997.

[7] ISO/IEC 9126-2: Software engineering - software product quality Part 2: External metrics.

[8] ISO/IEC 9126-3: Software engineering - product quality - Part 3: Internal metrics.

[9] ISO/IEC 9126-4: Software engineering - software product quality Part 4: Quality in use metrics.

[10] ISO/IEC 12119: Information technology software package - Quality requirement and testing.