9장. 인스펙션과 코딩

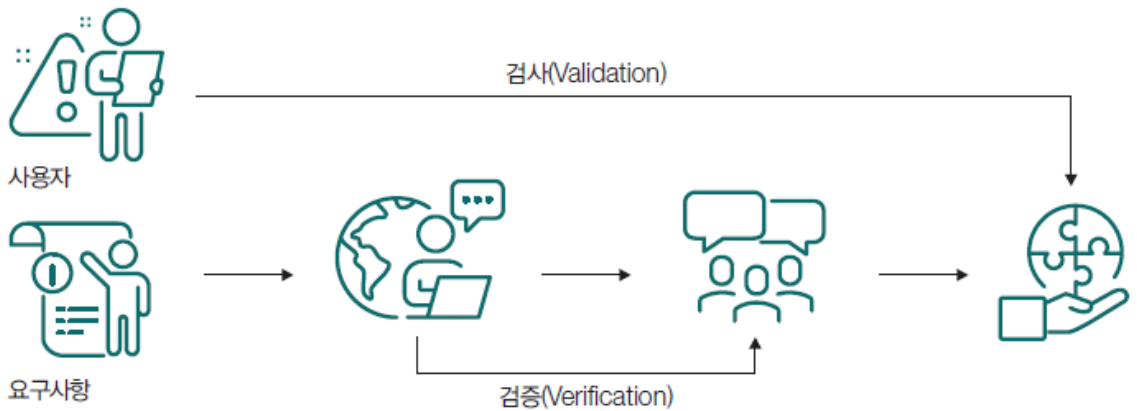
숙명여자대학교
SOOKMYUNG WOMEN'S UNIVERSITY

Contents

1. 인스펙션 개요
2. 프로그래밍 언어 유형
3. 좋은 코딩 기법
4. 코딩 가이드라인
5. 오픈 소스 기반 개발

Verification & Validation (1/2)

- Verification(검증)
  - 이전 단계의 결과물과 현재 단계의 결과물이 동일한지를 확인하는 활동
- Validation(검사)
  - 산출물이 사용자의 요구사항을 충족하는지를 확인하는 활동



Verification & Validation (2/2)

- V&V 정적 활동 :
  - 소프트웨어를 실행하지 않고 정적인 형태로 V&V 활동을 수행
  - 리뷰, 코드 리딩, 인스펙션, 프로그램 증명 등
- V&V 동적활동
  - 코드를 실행하면서 그 과정을 살펴보는 것
  - SW 테스팅

1. 인스펙션 개요 (1/5)

- 인스펙션(Inspection)
  - 체계적으로 정의된 절차를 기반으로 결함을 발견하기 위해 훈련된 엔지니어에 의해 수행되는 산출물의 동료 검토(Peer Review)를 의미
  - 제품 또는 응용 영역의 다수 전문가가 프로젝트 산출물의 기술적 정확성이 충족되는지를 확인 하는 모임
  - 일정한 절차와 사전에 정의된 양식을 이용하여 주어진 산출물에서 오류를 찾아내는 전문가 활동

1. 인스펙션 개요 (2/5)

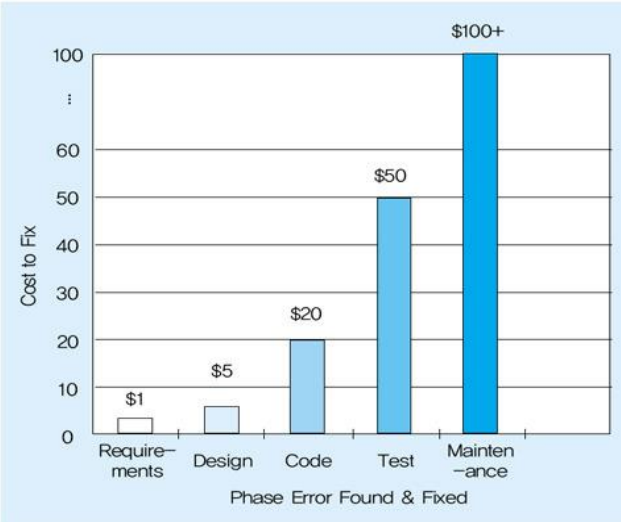
- 인스펙션의 목적
  - SW가 사양을 충족하는지, 지정된 품질 속성을 나타내는지 확인하기 위해
  - SW 개발의 각 단계에서 상세한 결함 분석을 수행하기 위한 기반 데이터를 확보하기 위해
  - 인스펙션을 통해 얻은 결과를 기반으로 조직에서 개선되어야 할 프로세스가 무엇인지 식별 가능
  - 예) 기본 설계서의 인스펙션에서 가장 많은 결함이 발견된다면 조직은 소프트웨어 아키텍처를 개발하는 지식 수준이나 기술이 낮다고 판단될 수 있음

1. 인스펙션 개요 (3/5)

- SW 개발 활동의 모든 단계에서 인스펙션을 수행
  - 프로젝트 계획서에 대한 인스펙션, 요구사항 인스펙션, 분석서 인스펙션, 소프트웨어 아키텍처 인스펙션, 코드 인스펙션, 테스트 결과 인스펙션 등
  - 각 산출물을 인스펙션하는데 사용할 체크리스트 준비
    - 이 체크리스트에는 인스펙션 수행자(즉, 인스펙터)가 참고할 산출물 오류 타입이 들어 있어야함
    - 오류 타입은 기존에 잘 알려진 오류들의 유형을 정의한 것으로, 인스펙션 직전에 수정·보완할 수 있음

1. 인스펙션 개요 (4/5)

- 인스펙션의 효과 : 결함 수정 비용의 감소
  - 유지보수 단계에서 발견한 결함의 수정비용은 요구사항 식별 단계에서 발견한 결함 수정 비용의 200배에 달함



결함 수정의 비용 그래프

  - 최대한 소프트웨어 개발 초기에 결함을 제거해야 전체적인 프로젝트 비용 절감 효과가 큼

1. 인스펙션 개요 (5/5)

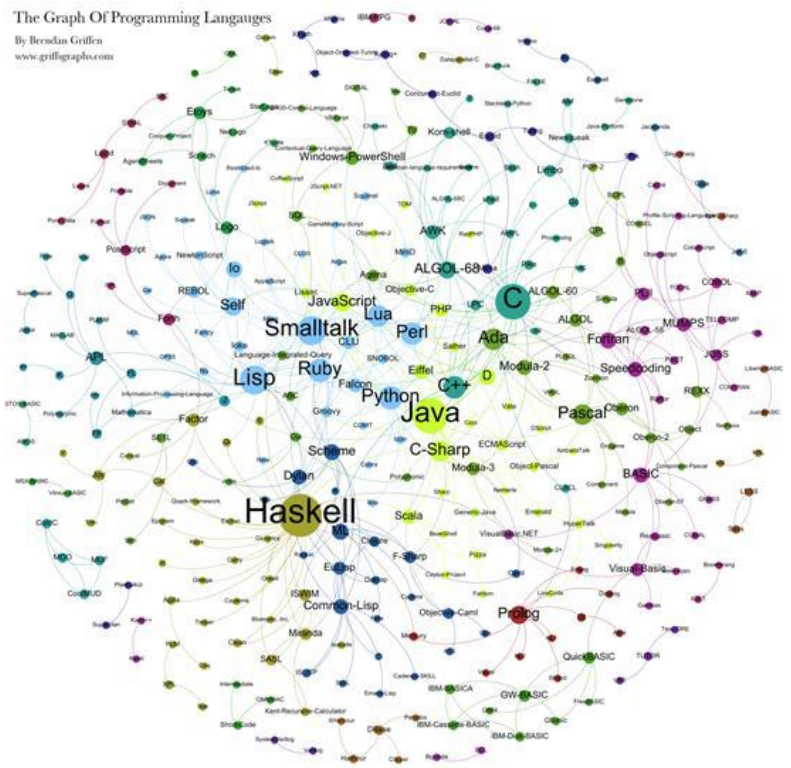
- 인스펙션 적용(A) 및 비적용 프로젝트(B)의 비용 분석 표
  - 인스펙션 수행의 큰 비용 절감 효과를 확인할 수 있음

활동	프로젝트 A (인스펙션을 수행한 경우)			프로젝트 B (인스펙션을 수행하지 않은 경우)		
	결함 수	PH	$	결함 수	PH	$
상위 설계 인스펙션	50	115	2,875	0	0	0
상세 설계 인스펙션	140	700	17,500	0	0	0
코드 인스펙션	110	1,700	42,500	0	0	0
단위 테스트	50	770	19,250	125	1,520	38,000
통합 및 시스템 테스트	100	2,490	62,250	190	4,290	107,250
소계	450	5,775	144,375	315	5,810	145,250
유지보수	50	13,000	325,000	185	31,900	797,500
합계	500	18,775	469,375	500	37,710	842,750

PH: Person Hour
$: 25$/hour

2. 프로그래밍 언어 (1/2)

- 프로그래밍 언어의 개수
  - TIOBE 2017년 현재 250여개
  - 위키피디아 700여개
  - FOLDOC(Free On-line Dictionary of Computing) 1,000여개
  - HOPL(History of Programming Languages) 8,945개

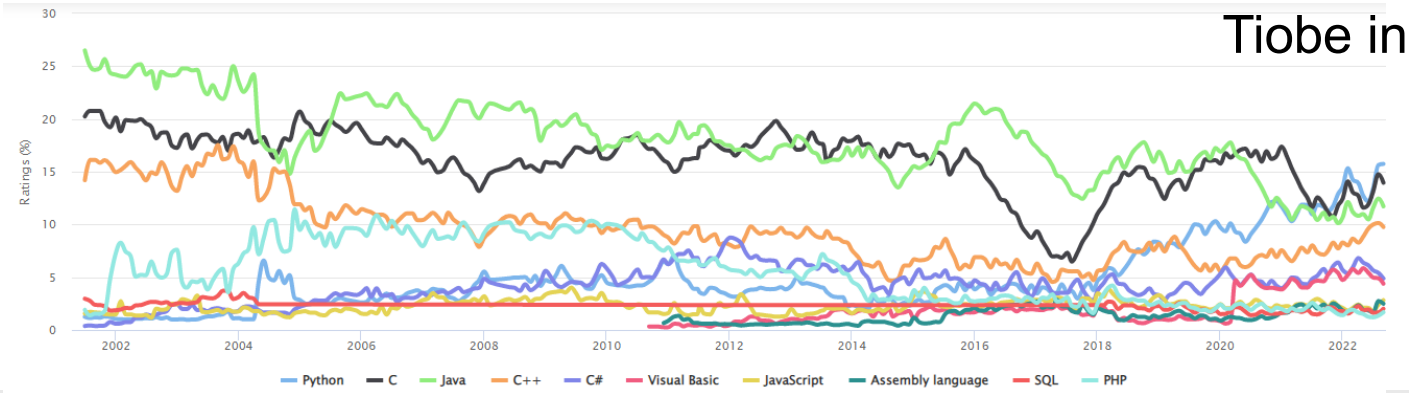


프로그래밍 언어 그래프

2. 프로그래밍 언어 (2/2)

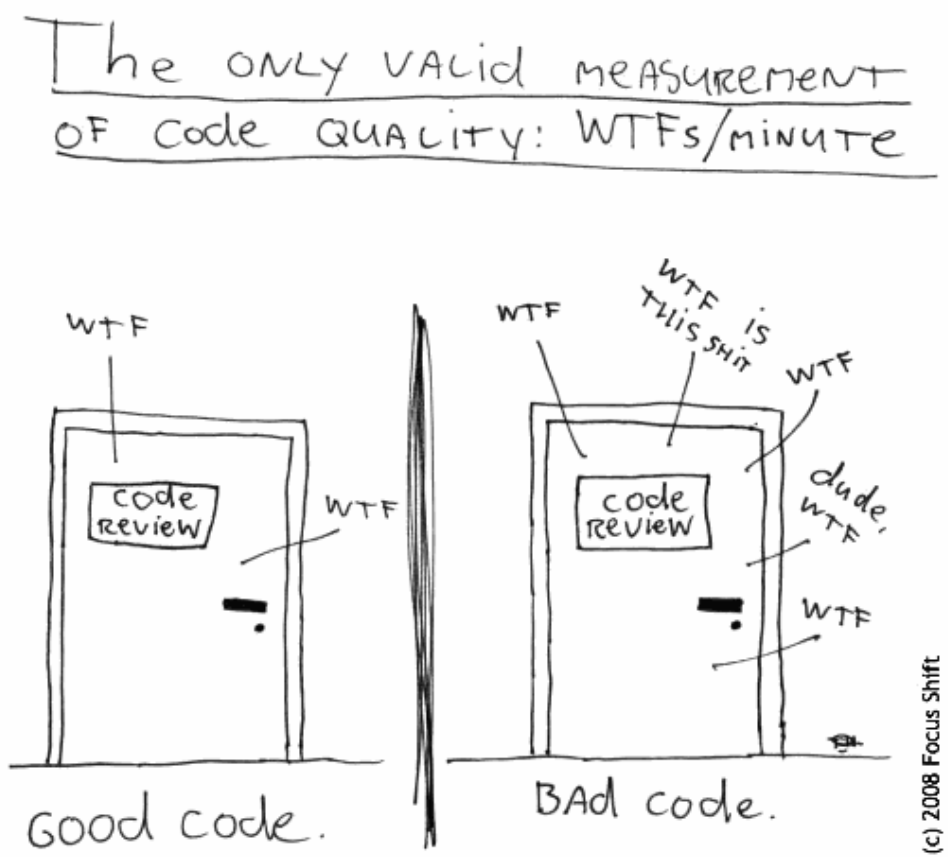
- 주요 프로그래밍 언어의 출현 연도
  - 1950년대 : Assembly Lang. 포트란, COBOL..
  - 1960년대 : BASIC, PL/1, Simula..
  - 1970년대 : Pasacl, C, Prolog, ML, Smalltalk, SQL..
  - 1980년대 : C++, Ada, Objective-C, Perl …
  - 1990년대 : Python, Ruby, Java, PHP, Delphi …
  - 2000년대 : C#, Scala, Go, Swift, F# …

Tiobe index



3. 좋은 코딩기법 (1/4)

- 좋은 코딩이란 ?



3. 좋은 코딩기법 (2/4)

Good code :

Code should be well-tested. Tests serve as an executable specification of the code and examples of its use.

Bad code :

Poor coding standard and style.

3. 좋은 코딩기법 (3/4)

- 좋은 소프트웨어 코드의 6가지 공통점
  - 쉽게 읽을 수 있다(가독성)
  - 주석이 잘 작성되어 있다.
  - 코드구조가 간결하다(코드를 간결하게 만들면 버그를 줄일 수 있음)
  - 변경에 탄력적이다(최소 한의 수정으로 미래에 필요한 사항을 수용할 수 있도록)
  - 활용을 위해 관리할 수 있어야 한다(어떤 코드든 삭제/무시 되는 일 없이 추후 활용될 수 있게 관리되어야 함)
  - 제 기능을 해야한다

3. 좋은 코딩기법 (4/4)

- 좋은 코드 작성을 위한 주요 규칙
  - 최적화보다는 가독성이 우선
  - 아키텍처를 우선 개발하라
    - 아키텍처 설계 없이 코드를 개발한 경우 대개 코드의 구조적인 문제 발생
  - 테스트 커버리지를 고려하라
  - 간단하고 단순하게 코딩하라
  - 주석을 작성하되 보조적으로 사용하라(좋은 코드는 주석이 없어도 이해가 되는 코드)
  - 가능하면 자동화 도구를 사용하라

4. 코딩 가이드라인 (1/2)

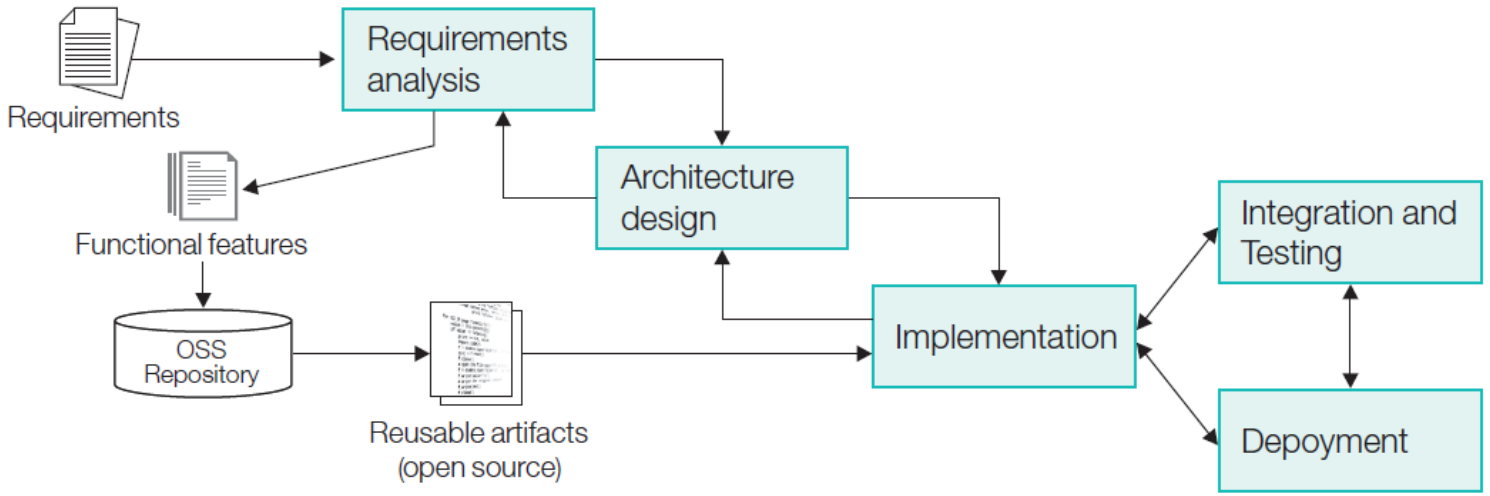
- MISRA-C 코딩 표준
  - Motor Industry Software Reliability Association for C language
  - 1998년 유럽의 자동차 연합에서 자동차 탑재 SW의 신뢰성 및 호환성을 높이겠다는 의도로 개발한 C 프로그램 코딩 표준
  - 우주/항공, 의료장비, 국방, 철도 등 다양한 산업에서 Best Practice로서 광범위하게 적용되는 가이드라인
  - MISRA-C:1998, MISRA-C:2004

4. 코딩 가이드라인 (2/2)

- 시큐어 코딩
  - 구현단계에서 해킹 등의 공격을 유발할 가능성이 있는 잠재적인 보안 취약점을 사전에 제거하여, 안전한 SW를 개발하기 위한 프로그래밍 기법
  - 2008년 CMU/SEI의 CERT Coordination Center에서 CERT C 코딩 표준 발표
    - 코딩규칙 별 제시된 심각성, 발생 가능성, 교정 비용에 대한 정량화된 척도 제시
  - 국내 : 2012년 5월, 행정안전부 발표
    - 정보 시스템 구축 운영 지침 개정안 행정 예고를 통해 시큐어 코딩 의무화 법안을 발표

5. 오픈소스 기반 개발 (1/4)

- 오픈 소스 기반의 소프트웨어 개발 프로세스



  - 라이선스 권한의 확인 검토가 필요
    - 공개된 코드를 사용할 수 있는지, 수정할 수 있는지, 수정하여 재배포할 수 있는지 등 확인
    - GPL, LGPL, BSD, Apache, MIT 등의 라이선스 형식

5. 오픈소스 기반 개발 (2/4)

- 오픈 소스 활용 시 주의 사항
  - 오픈 소스 검증 체계 구축
    - 라이선스 유형, 보안취약성 확인
  - 지속적인 오픈 소스 관리
    - 라이선스 준수 여부 및 새로운 취약점은 없는지 정기적으로 점검
  - 철저한 기술 검토
  - 오픈 소스 변경 사항 기록
    - 통합되는 오픈 소스에 대한 수정이 이루어졌다면 해당 변경 사항을 기록

5. 오픈소스 기반 개발 (3/4)

- 오픈 소스 개발 지원 도구
  - 개발 프로세스의 지원, 코드 통합, 문서화 지원 등 다양한 업무에 사용
  - Git : 소규모 프로젝트와 대규모 프로젝트를 모두 지원할 수 있도록 구축된 오픈 소스 기반의 분산 버전 제어 시스템
  - Eclipse IDE : Eclipse는 여러 언어를 지원하는, 포괄적인 Java 중심 통합 개발 환경
    - Java, JavaScript/TypeScript, C/C++, PHP, Rust 등을 지원하고, 다양한 플러그인을 설치하여 쉽게 활용 가능

5. 오픈소스 기반 개발 (4/4)

- Bootstrap : 웹에서 반응형 모바일 프로젝트를 구축하는 데 사용되는 프런트 엔드 구성 요소를 제공하는 라이브러리
  - 개발자는HTML, CSS, JavaScript 에서 작업할 수 있으며 jQuery 플러그인 등을 사용할 수 있음
  - 웹 페이지 개발을 단순화하기 위해 웹 프로젝트의 색상, 크기, 글꼴, 레이아웃 등을 쉽게 활용 할 수 있음.
- Apache NetBeans, Ruby on Rails, Apache Cordova, GNU Emacs, Brackets, Atom, Azure 등..