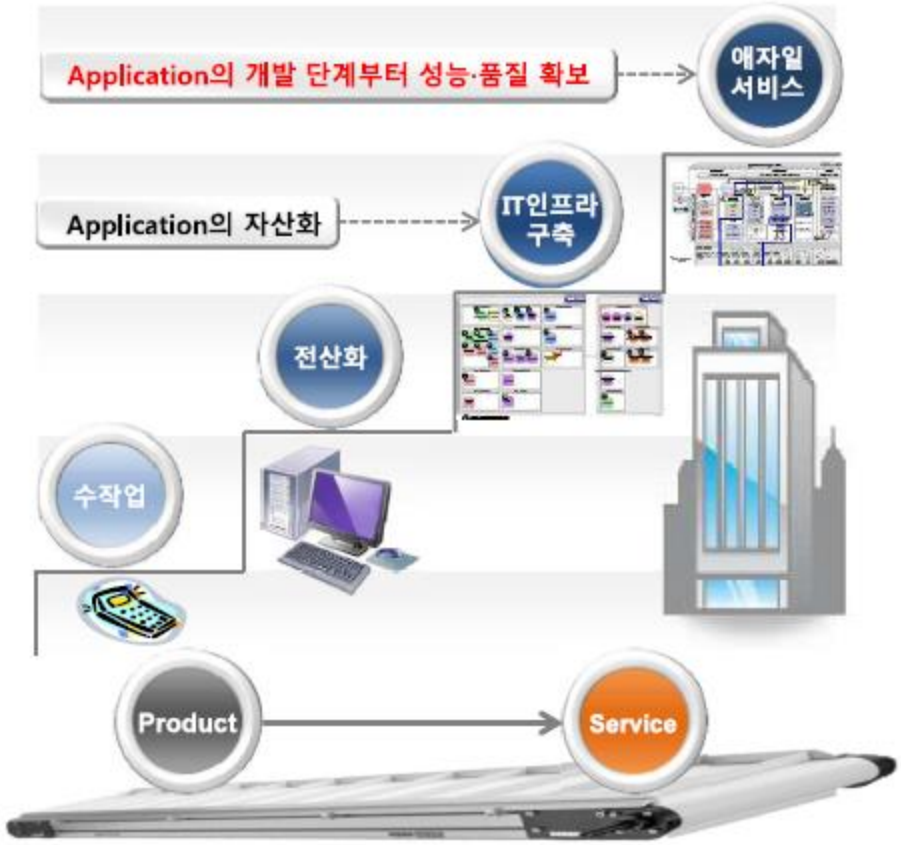
- 실시간 모니터링 기반의 선제적 품질관리 솔루션 -

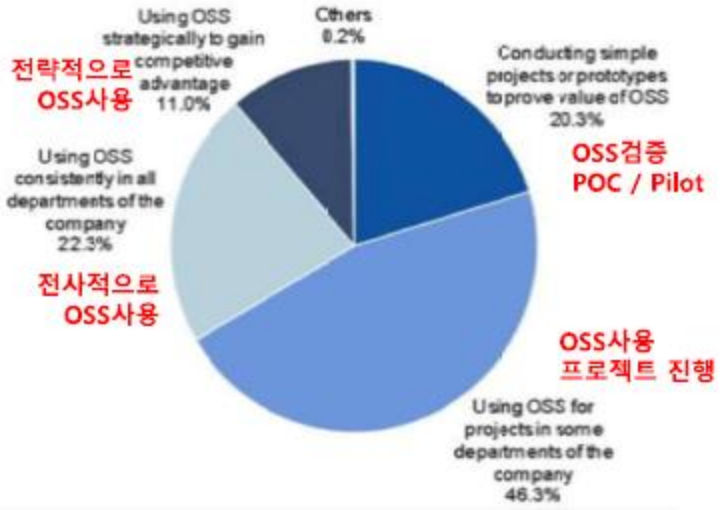
'Green Coding'은 낮은 품질의 애플리케이션 소스코드에서 결함 코드와 보안취약성 코드, 성능 저하 및 오류 원인을 제거하여 높은 품질의 'Clean Code'로 만드는 품질관리 기법입니다.

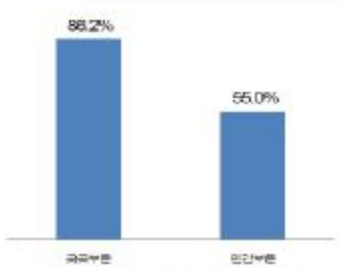
고객과 시장의 변화에 민첩한 기업을 위해 엔터프라이즈 애플리케이션의 개발 단계에서부터 품질, 성능, 보안 등에 대한 선제적인 품질 관리가 중요하고, 특히 비용 절감을 위해 도입한 OSS 기반에서 개발한 애플리케이션의 품질은 반드시 확보되어야 합니다.

엔터프라이즈의 진화와 애플리케이션의 자산화

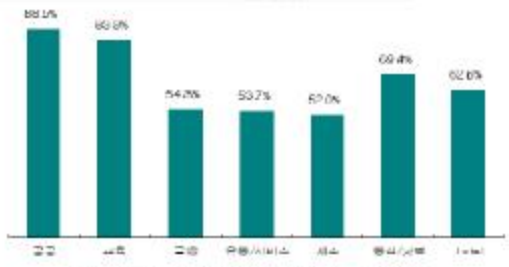


비용 절감을 위한 OSS의 도입 추세 확산





[공개소프트웨어 도입률]

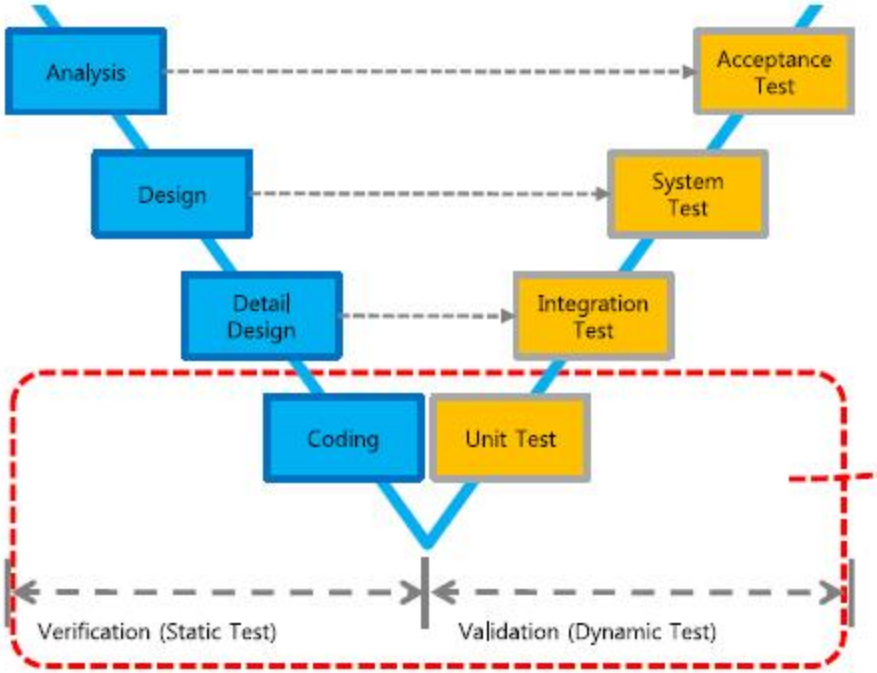


[업종별 공개소프트웨어 도입률]

1.2 개발 단계에서의 선제적인 품질관리 필요성

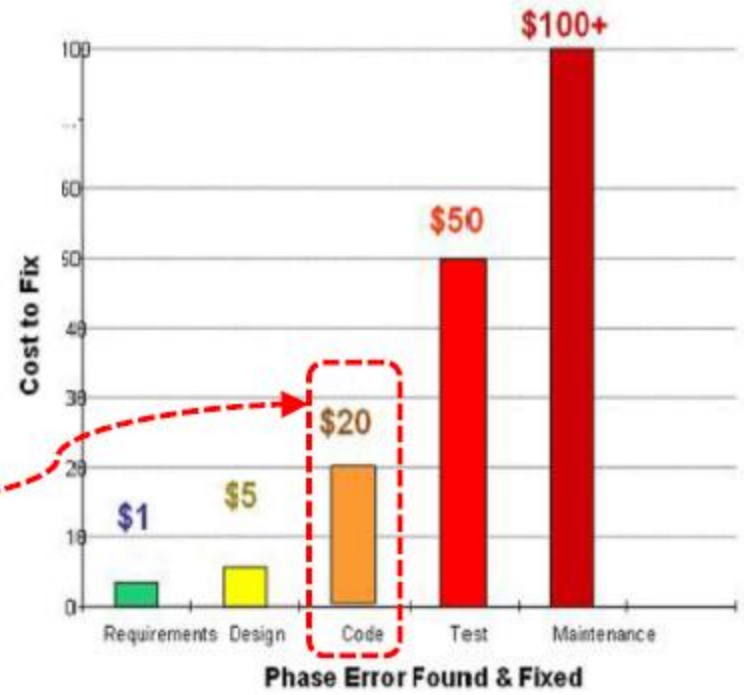
애플리케이션의 분석 단계에서 결함을 발견하고 제거하면 테스트 및 운영 단계에서보다 50배, 100배의 비용을 절감할 수 있으며, 따라서 현실적인 애플리케이션의 개발 단계에서부터 충분한 정적·동적 분석을 통한 지속적인 품질 관리가 중요합니다.

개발 단계와 V-model



* 출처 : Software Testing Techniques, Boris Beizer

개발 단계와 결함수정 비용



* 출처 : Software Verification & Validation, Steven Rakitin

IT 세계의 모든 SW 조직이 풀고자 하는 문제

세밀한 관리와 통제에도 불구하고, 프로젝트 마감일이 매번 연기되는데 그 원인은 무엇까?

낮은 품질비용으로 프로젝트 완료 및 유지보수를 할 수 없을까?

개발자의 개발활동에 대해 실시간으로 모니터링할 수는 없을까?

개발단계에서 개발자 스스로 소스코드에 대해 결함과 오류를 찾아 수정할 수는 없을까?

운영중인 소스코드에 대해서 품질 관리를 할 수 없을까?

형상관리서버에 Clean Code만 이관할 수는 없을까?

프로젝트마다 반복되는 동일 기능의 코드를 표준화할 수 없을까?

프레임워크 기반의 여러 가지 애플리케이션 제약, ILM 주석과 같은 필수 입력 항목 등에 대한 위반 여부를 체크할 수 있을까?

실시간 모니터링 기반의 선제적 프로젝트 관리

애플리케이션의 품질, 보안 관리

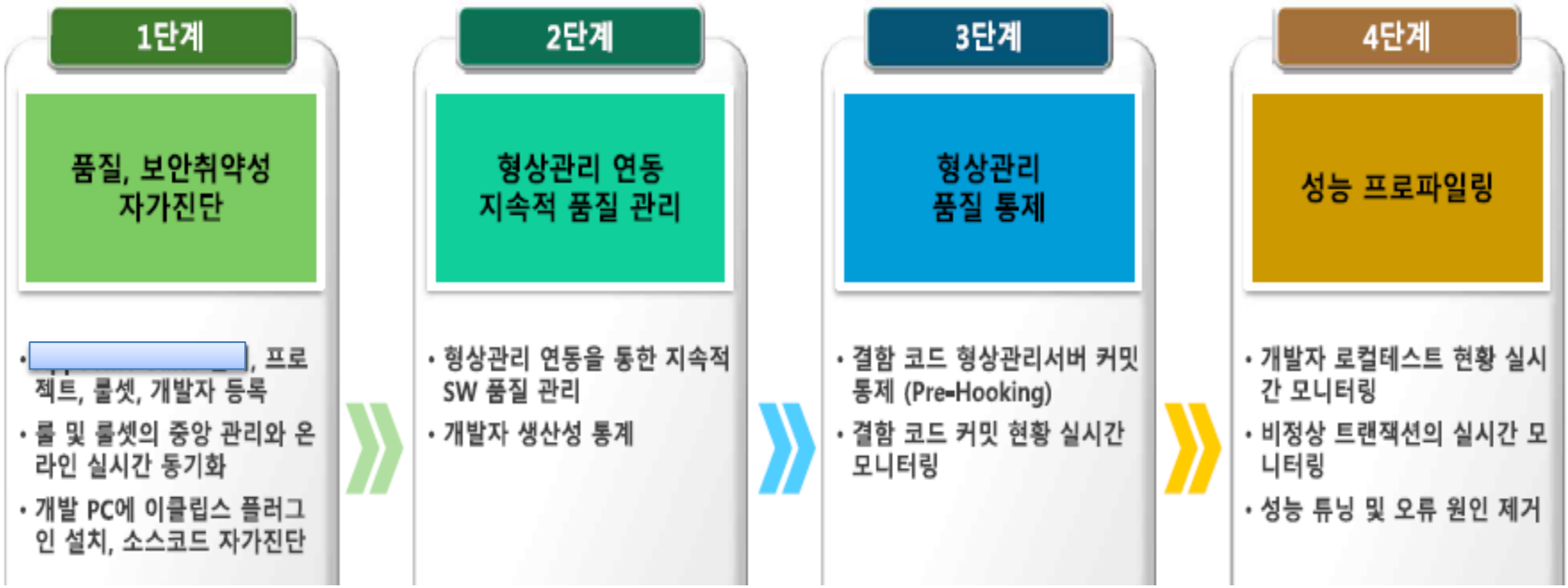
- SW 품질에 대한 국제표준(ISO/IEC 9126) 기반의 관리
- 행정 안전부의 "시큐어 코딩" 가이드 기반의 관리
- 프로젝트 표준 룰에 위배된 결함 코드의 커밋 통제

애플리케이션 성능 확보

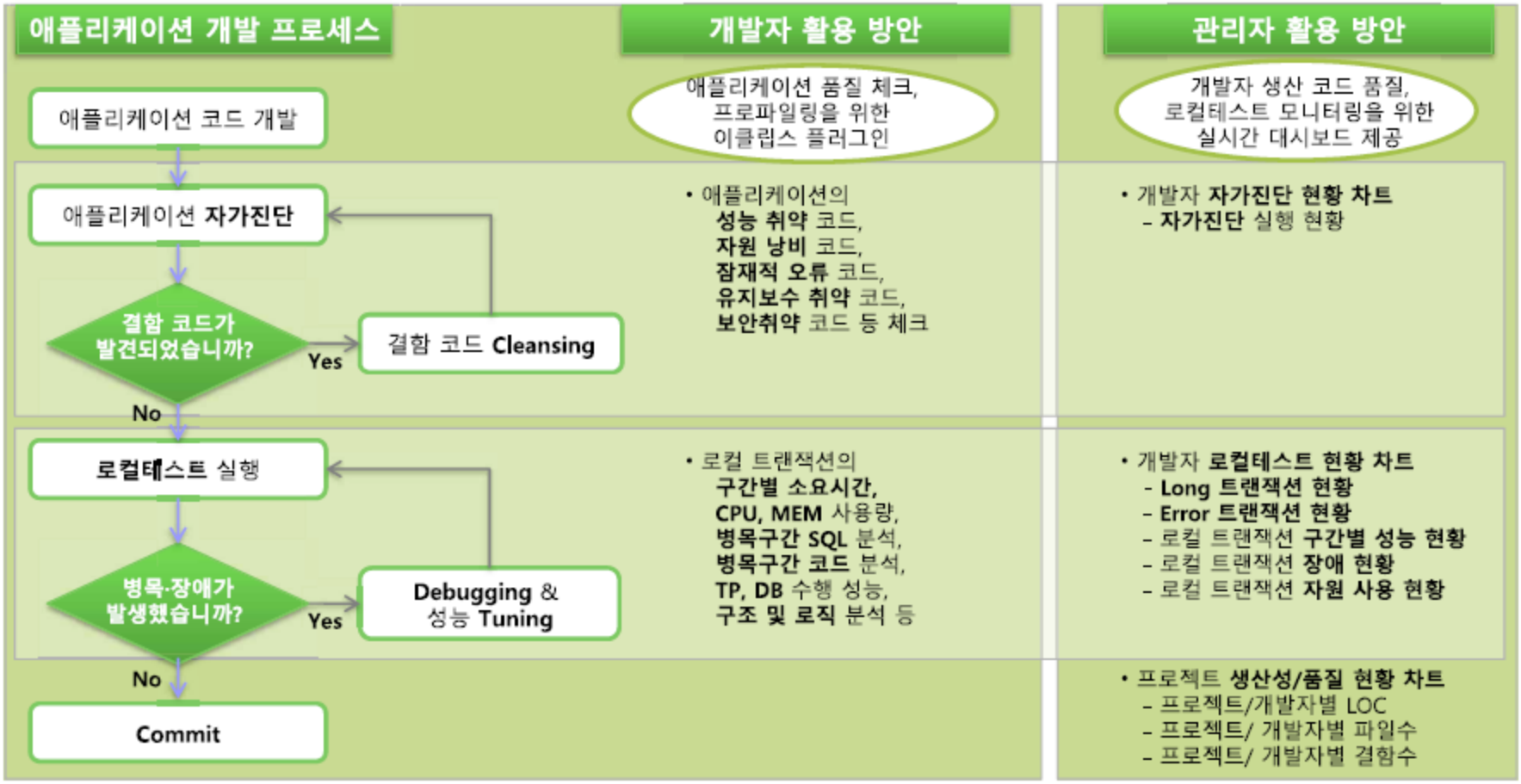
- 비정상 트랜잭션의 실시간 모니터링 및 구간 분석
- 트랜잭션이 사용한 CPU, MEM 등의 자원 사용량 분석
- 병목 및 오류 구간의 원인을 제거하여 성능 향상

개발 활동의 실시간 모니터링

- 프로젝트의 개발 진행 상황을 가시화하여 개발 일정 준수
- 개발자의 개발, 로컬 테스트, 커밋 현황을 투명하게 파악
- 개발자용 코드 인스펙션 및 로컬 트랜잭션 분석 플러그인 제공


1단계
품질, 보안취약성 자가진단
• , 프로젝트, 룰셋, 개발자 등록
• 룰 및 룰셋의 중앙 관리와 온라인 실시간 동기화
• 개발 PC에 이클립스 플러그인 설치, 소스코드 자가진단
2단계
형상관리 연동 지속적 품질 관리
• 형상관리 연동을 통한 지속적 SW 품질 관리
• 개발자 생산성 통계
3단계
형상관리 품질 통제
• 결함 코드 형상관리서버 커밋 통제 (Pre-Hooking)
• 결함 코드 커밋 현황 실시간 모니터링
4단계
성능 프로파일링
• 개발자 로컬테스트 현황 실시간 모니터링
• 비정상 트랜잭션의 실시간 모니터링
• 성능 튜닝 및 오류 원인 제거

애플리케이션 개발 프로세스

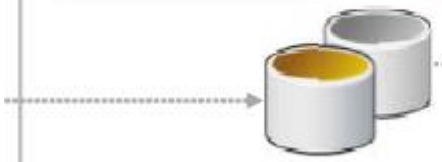


클라이언트(개발자 PC)

정적 분석 도구

동적 분석 도구

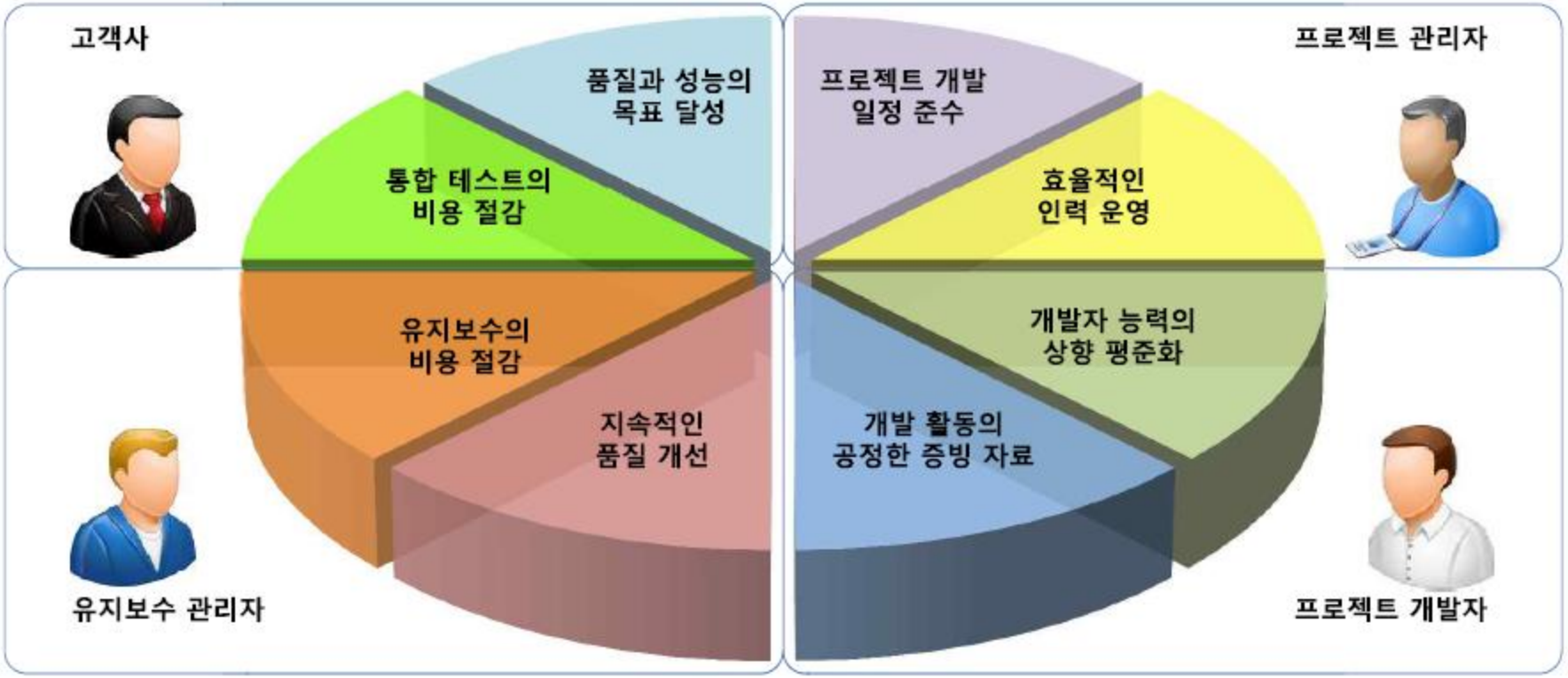
형상관리 서버



개발 활동 모니터링 대시보드

품질, 성능 정보

도입 효과


고객사
품질과 성능의
목표 달성
통합 테스트의
비용 절감
프로젝트 관리자
프로젝트 개발
일정 준수
효율적인
인력 운영
유지보수의
비용 절감
지속적인
품질 개선
유지보수 관리자
개발자 능력의
상향 평준화
개발 활동의
공정한 증빙 자료
프로젝트 개발자

도입 당위성

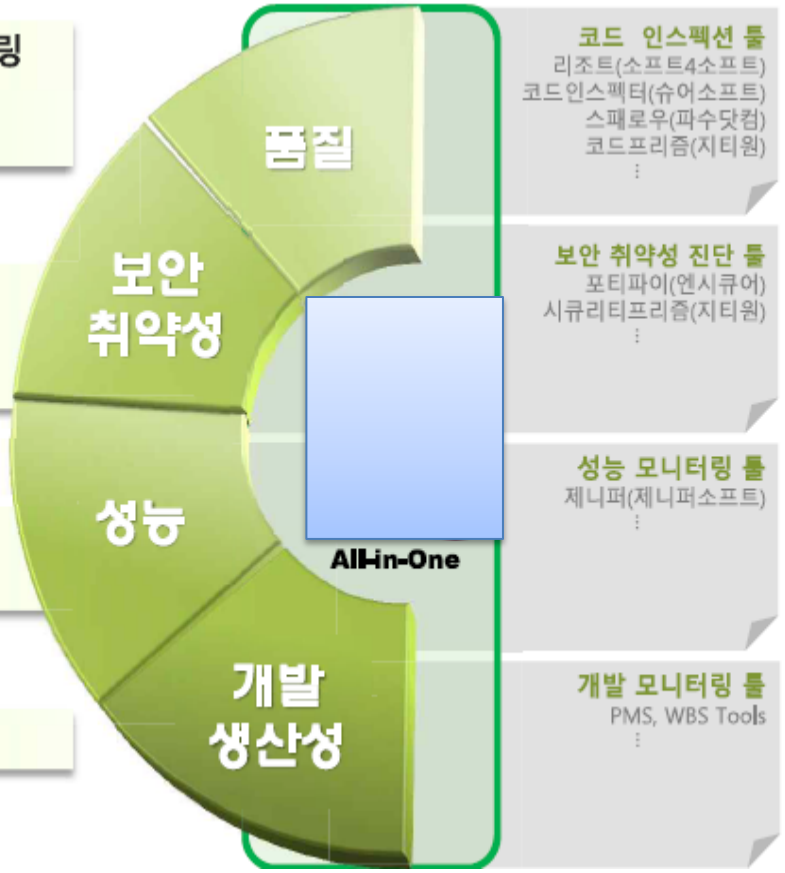
개발자의 기술능력 향상과 프로젝트 진행에 대한 소스레벨 모니터링
품질과 보안 취약점, 성능, 생산성 등을 실질적으로 제고할 수 있음

1. 프로젝트 투입 인력의 개발 상황 및 단위 테스트 모니터링
   - 코딩 파일 수, 코딩 라인, 자가진단 횟수
   - 개발 활동 체크 및 사전 개발 진척 파악

2. 문법, 보안, 성능에 대한 코드 레벨의 취약점 파악
   - 글로벌 표준 문법 및 기본 룰셋 제공
   - 꼭 피해야 하는 문법적, 보안적, 성능적 요소 디버깅

3. 각각 개발자의 표준 기술력 향상
   - 표준 코드 가이드

4. 소스코드의 컴팩트화, 클린화



어디까지 가볼 까?

애플리케이션 개발 모니터링 실시간 대시보드

애플리케이션 테스트 트랜잭션 현황

형상관리 서버의 커밋 현황

프로젝트 파일별 품질 현황

프로젝트 룰별 품질 현황

개발자 지원 도구

품질 자가진단 플러그인

로컬 트랜잭션 모니터링 프로세스

로컬 트랜잭션 구간별 분석 플러그인

0 로컬 트랜잭션 실시간 대시보드/사후분석

1단계 : 개발자 지원도구

자가진단
품질관리

- 애플리케이션 코딩 표준
- 애플리케이션 아키텍처

- 애플리케이션 **성능 취약** 코드
- 애플리케이션 **자원 낭비** 코드
- 애플리케이션 **잠재적 오류** 코드
- 애플리케이션 **유지보수 취약** 코드
- 애플리케이션 **보안취약** 코드

2단계

개발자 품질 자가진단
이클립스 플러그인