코드 인스펙션 사례
- Java/JSP/C(Pro*C) 코드 -

Soft4Soft

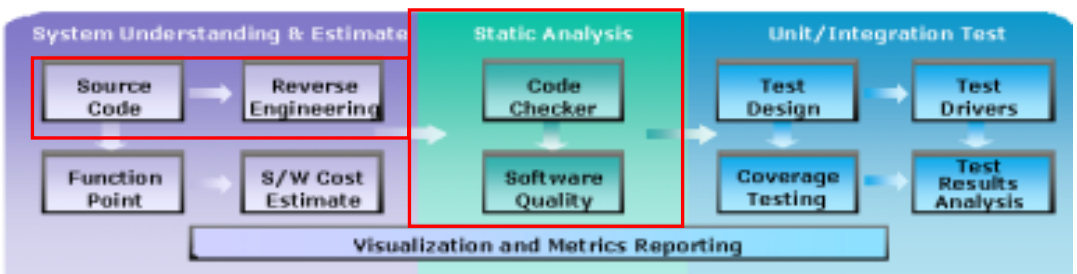
http://www.soft4soft.com

Contents

- 분석 내용 및 목적
- **Java** 품질 분석
  - 코드 검사 결과
  - **SW Architecture** 분석 결과
- **JSP** 품질 분석
  - 코드 검사 결과
- **C(Pro*C)** 품질 분석
  - 코드 검사 결과
- 코드 검사 컨설팅 사례
- **SW** 품질 관리 및 코드 검사 효과
- 코드 검사 활동 및 운영 방안

분석 내용 및 수행 목적

- 분석 내용
  - **Java** 코드
  - **JSP** 코드
  - **C(Pro*C)** 코드
- 수행 목적
  - 프로그램 분석: 구문 및 의미 에러 식별
  - **SW** 품질 분석: **SW** 크기 및 구조 복잡도 측정/평가
  - 코딩 검사: 가독성 표준 검사, 인터페이스 및 자원 등의 결함 검사
  - **SW Architecture** 분석: **SQL** 개발 표준 검사
- 수행 도구
  - **RESORT for Java/JSP/C(Pro*C)**



Java 코드 품질 검사

- **Code Inspection 결과** (필수 권고사항은 반드시 수정 또는 삭제)

구분	권고 사항 및 점검기능	위배 문장
가독성 및 유지보수저하	(선택) 명명 규칙(Package, method,global/local variable:소문자시작)	5/247/206/67
	(선택) 명명 규칙(class: 대문자 시작)	1
	(선택) 하드 코딩(조건절)	330
Dead Code (메모리누수)	(필수) 미사용 메소드	47
	(필수) 미사용 전역변수	25
	(필수) 미사용 지역변수	217
잠재적 에러	(선택) Switch 문(case의 break/comment)	77
	(필수) 빈 제어문 문장	39
	(선택) 빈 catch 및 finally 구문	319
	(필수) Finally 구문에서 return 사용	0
성능 저하	(필수) string 연산	222
	(선택) 반복문에서 선언문	266
	(선택) 반복문에서 메소드	759
	(선택) 중첩 try	18
	(선택) Consol 메시지	0
	(선택) SELECT 질의 *(all-column)	0
자료 무결성 저하	(필수) commit()-rollback() 호출	0
DB 자원관련 에러 (메모리 누수)	(필수) DB 자원 미해제	11
	(필수) DB 자원 재사용	1

Java 품질 측정

- **SW Architecture 분석 결과**
  - **SW 개발시, SW Architecture간의 독립성 및 일관성을 고려하여 SQL 개발표준을 준수하였는지를 평가**
    - **Business component 기반 SQL 코딩 : SQL 개발 표준**
      - **Insert/Update/Delete의 경우 Entity Class의 기능을 이용 SQL을 작성**
      - **Select(Read)의 경우 Template Engine 기반으로 SQL을 작성**
    - **화면 중심 SQL 코딩 : 유지보수 및 권한 부여 관리 어려움**
      - **Select(Read)와 Insert/Update/Delete를 혼용 사용으로 SQL을 작성**

품질 특성	진단 사항	부적합 품질
SQL 개발 표준 준수	Select와 Create/Update/Delete 문 혼용 사용	0 Class

JSP 코드 품질 검사

- **Code Inspection 결과** (필수 권고사항은 반드시 수정 또는 삭제)

구분	권고 사항 및 점검기능	위배 문장
가독성 및 유지보수저하	(선택) 명명 규칙(Package, method,global/local variable:소문자시작)	39/0/0/52
	(선택) 명명 규칙(class: 대문자 시작)	40
	(선택) 하드 코딩(조건절)	6
Dead Code (메모리누수)	(필수) 미사용 메소드	0
	(필수) 미사용 전역변수	0
	(필수) 미사용 지역변수	79
잠재적 에러	(선택) Switch 문(case의 break/comment)	0
	(필수) 빈 제어문 문장	28
	(선택) 빈 catch 및 finally 구문	19
	(필수) Finally 구문에서 return 사용	0
성능 저하	(필수) string 연산	10
	(선택) 반복문에서 선언문	18
	(선택) 반복문에서 메소드	29
	(선택) 중첩 try	0
	(선택) Consol 메시지	44
	(선택) SELECT 질의 *(all-column)	0
자료 무결성 저하	(필수) commit()-rollback() 호출	0
DB 자원관련 에러 (메모리 누수)	(필수) DB 자원 미해제	0
	(필수) DB 자원 재사용	0

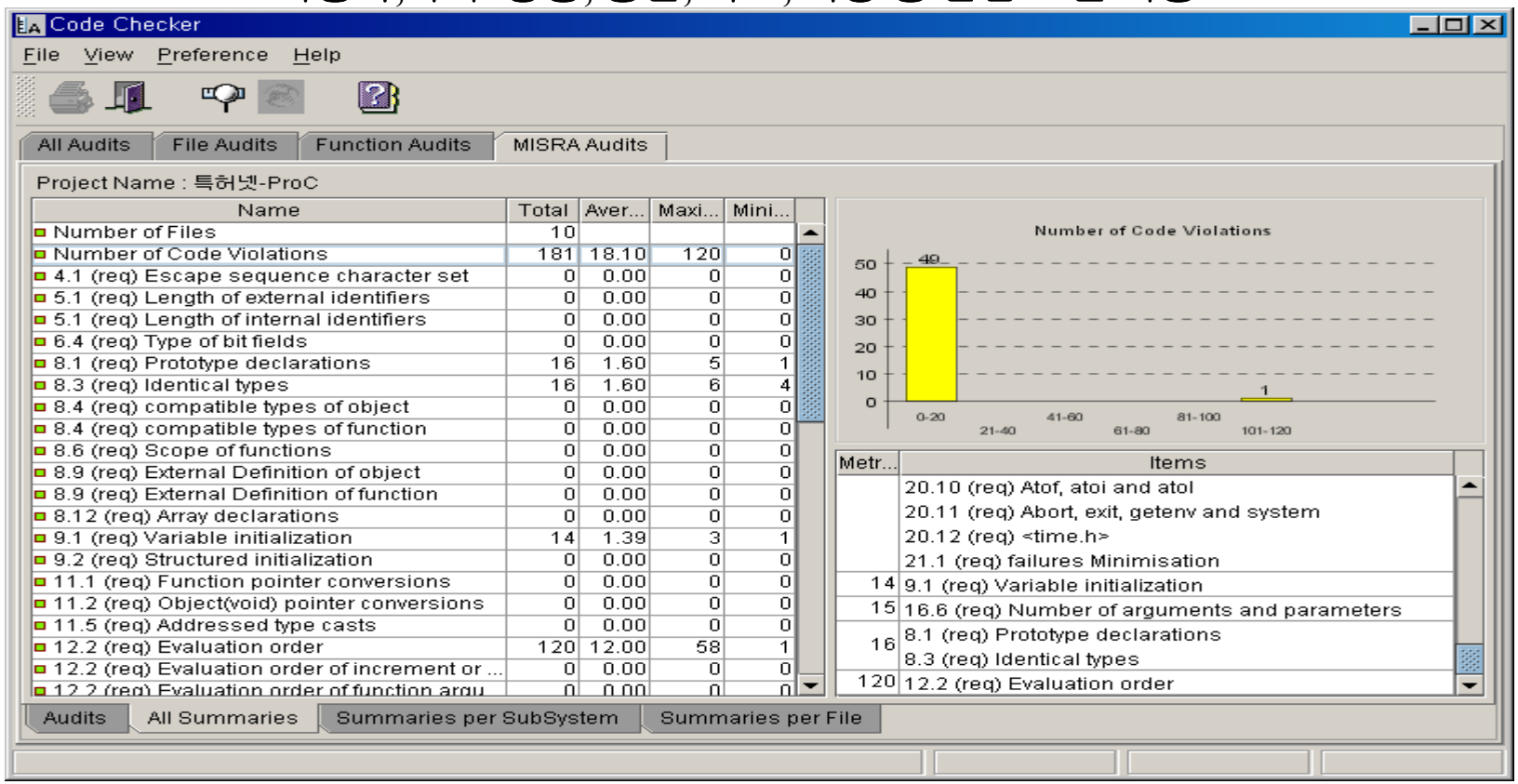
C 코드 품질 검사

- **Code Inspection 결과** (필수 권고사항은 반드시 수정 또는 삭제)

구분	권고 사항 및 점검기능	위배 문장
가독성 및 유지보수저하	(선택) 명명 규칙(file, global variable,function, local variable)	10/0/24/56
	(선택) 하드 코딩	301
Dead Code (메모리 누수)	(필수) 미사용 지역변수	14
잠재적 에러	(선택) Switch 문(case의 break/comment)	6
	(선택) 내.외부 식별자 동일	0
	(선택) 변수 초기화	14
	(선택) 포인터 초기화	13
데이터 에러	(선택) 정수 type cast	0
	(선택) 실수 type cast	0
	(선택) 포인터 type cast	0
제어 에러	(필수) 도달하지 않은 코드	0
	(선택) Sizeof 연산자	0
	(선택) Goto 문	0
저장관리 에러	(필수) 포인터 산술	0
	(필수) 포인터 뺄셈	0
	(필수) 동적 heap메모리 할당 사용 금지	0
인터페이스 에러	(필수) 파라미터 타입 선언과 정의 불일치, 반환 타입 불일치	16
	(필수) 파라미터 개수 선언과 정의 불일치	15
	(선택) 함수의 가변 인자 사용 금지	0
	(필수) Void 함수의 return	0

C 코드 품질 검사

- **Code Inspection 결과 화면 – MISRA-C: 2004**
  - C 프로그래밍(임베디드) 가이드라인-141 규칙
  - 자동차, 우주 항공, 통신, 의료, 국방 등 산업 표준 적용



C 코드 품질 검사

- **Implicit cast: long to int – MISRA 10.1 Rule**
  - 동일 타입이 아닌 유사한 타입들 간의 계산하는 경우
  - 계산된 값은 정보 유실 발생(우주선 SW라면, 그 피해는 천문학적임)

```
main() {
  unsigned long position;
  int          coordinate;
  coordinate = position >> 1;  // 형 변환 계산
}
```

- **Identifier Scope – MISRA 5.2 Rule**
  - 한 파일에서 전역 변수와 지역 변수를 동일 이름으로 사용하는 경우
  - 실행에 문제가 있는 것은 아니며, 프로그래머나 테스터가 두 개를 서로 혼돈하여 버그를 초래할 수 있으며, 이러한 것이 수백라인 떨어져 있다면 정말 엄격한 코드 검사과정에서도 이러한 것을 찾기는 쉽지 않음

```
#define FAILSAFE       (-1)
extern  unsigned long   err;                    //err 전역변수
unsigned long sysStartup(START tool) {
      unsigned long   err = FAILSAFE;       //err 지역 변수
}
```

코드 검사 컨설팅 사례 (I)

- 코딩 표준(30 규칙), 소스 검사 및 개발 진척도 관리 – 주 1회

코딩 표준
(30 규칙)

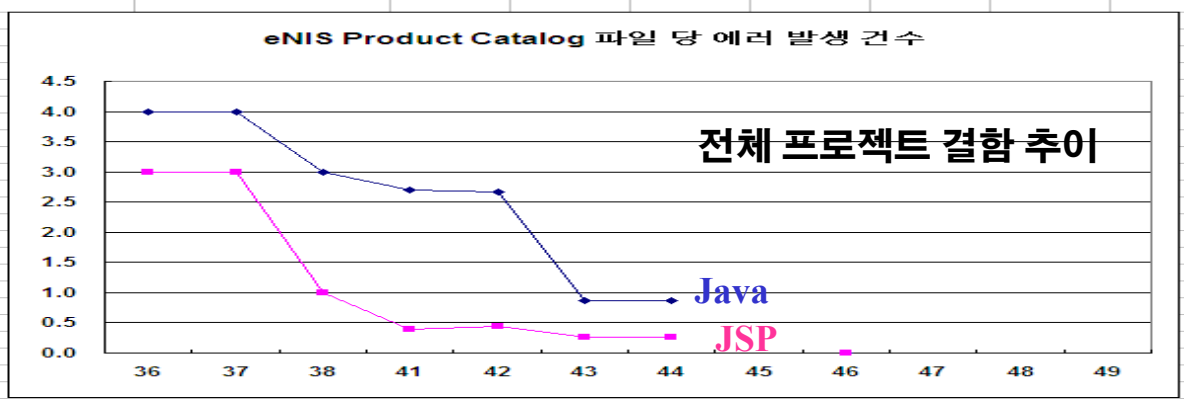
Scope	No		SUM	Ratio	eNIS											BMS	ERP	SEM			연동	MDM	Frame	eSVC
					NHS					YMS				BIZ	COMMON	SYS		LMS	VRP	RSC	EDR			
					PRD	SCE	TES	TPB	TRS	EQR	COA	SAQ	SPC	AGT											
File	1	File Comment	259	9.3%	2		9		12						171		1		2	45					10
	2	Class Declaration	1	0.0%															1						
	3	Import Declaration	9	0.3%			1		1											7					
Class	4	Class Comment	360	13.0%	2	3	18		42		1				171		7	24	2	71			5		6
	5	Failure Definition Variable	-	0.0%																					
	6	Constant Variable Naming	47	1.7%					2								18	15		6			5		1
Method	7	Method Comment	162	5.9%	8	11	47		19			8				8	5	2	2	58					
	8	Failure Definition Method	32	1.2%	3		4		2		1	1				1	1	13	1	5					
	9	[illegible]	-	0.0%														0							
	10	Method Naming	18	0.7%		1			1								2	10	1	2					1
	11	Method Name Length	-	0.0%														0							
	12	Switch Statement	17	0.6%														17							
	13	Additive Compound Assignment	51	1.8%			6				6			3			2	27		4			3		
	14	Floating Point Values	33	1.2%					1								29	1		3					1
	15	Empty Block Body	182	3.7%	1		31		47	3	6					3		4		7					
	16	[illegible] Finally Block	128	4.6%		2	59		25									1		58			3		
	17	[illegible] in the Finally Block	8	0.3%					8											4					
	18	Nested try Statement	35	1.3%		1	12		9								1	8					3		1
	19	[illegible] Synchronized Block	-	0.0%																					
	20	Static Variable Access	12	0.4%													8	6							
	21	Used Static Variables	24	0.9%													18	12					2		
	22	Failure Definition Local Variables	530	19.2%	75	10	268		54	4	38	2	7		6	4	22	12		22			18		6
	23	Collection Declarations	94	3.4%	1				4			24	2			1	48		5	5					4
	24	comment)-rollback()	4	0.1%																			4		
	25	Using a PreparedStatement instead of Statement	20	0.7%	1				2			6	8						2						1
	26	SELECT query with all-column	56	2.0%					9				1		46										
	27	Unbinding PreparedStatement	-	0.0%																					
	28	Ordering to release JDBC resources	60	2.2%	8	2	17		16		2						3	12							
	29	Debug Statement	230	8.3%			134		13	4			1		34		8	11	1	28			4		
	30	Using a log within the catch block	474	17.2%	7	12	57		189	1	11				166	2	3	25					73		2
		총위반수	2,758	100%	109	42	641	0	372	12	69	41	19	0	9	600	11	156	200	17	327	0	0	112	0	30
		총파일분석수	6,974		206	369	228	140	685	292	310	243	156	167	185	177	188	1,086	1,215	496	81	93	30	104	189	334
		분석에러 파일수	8		0	0	0	1	6	0	0	0	0	0	0	0	0	0	0	0	1	0		0	0	0
		총파일수	6,982		206	369	228	141	691	292	310	243	156	167	185	177	188	1,086	1,215	496	82	93	30	104	189	334
		총 클래스 수	7,037		206	369	228	140	686	292	310	243	156	167	185	177	188	1,086	1,268	501	82	93	31	104	191	334
		총 메소드 수	63,695		1,371	2,931	3,327	929	6,817	4,866	2,906	2,765	1,695	1,155	1,057	994	1,327	9,914	8,069	5,287	1,163	1,258	324	422	1,649	3,459
		파일당 위반수(평균)	0.4		0.5	0.1	2.8	0.0	0.5	0.0	0.2	0.2	0.1	0.0	0.0	3.4	0.1	0.1	0.2	0.0	4.0	0.0	0.0	1.1	0.0	0.1
		전주 파일당 위반 수	0.6		0.5	0.4	2.6	0.0	0.6	0.1	0.2	0.2	0.1	0.0	0.0	0.0	0.0	1.0	0.5	0.0	4.1	0.0	0.0	1.1	0.0	0.1
		전주대비 증가 파일당 에러수	-0.2		0.0	-0.3	0.2	0.0	-0.1	-0.1	0.0	-0.0	0.0	0.0	0.0	3.4	0.1	-0.9	-0.3	0.0	-0.1	0.0	0.0	-0.0	0.0	-0.0
		전주 총파일갯수	6,655		196	271	211	141	664	292	301	231	153	167	185	0	188	1,070	1,206	496	82	93	30	104	189	365
		전주대비 증가파일수	327		10	98	17	0	27	0	9	12	3	0	0	177	0	16	7	0	0	0	0	0	0	-31

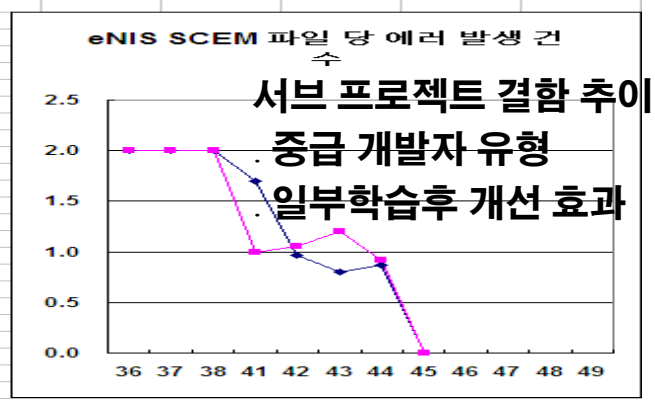
코드 검사 컨설팅 사례 (II)

- 파일당 결함 추이 및 개선 모니터링 – 주 1회

모듈별 파일 당 코드 에러 수

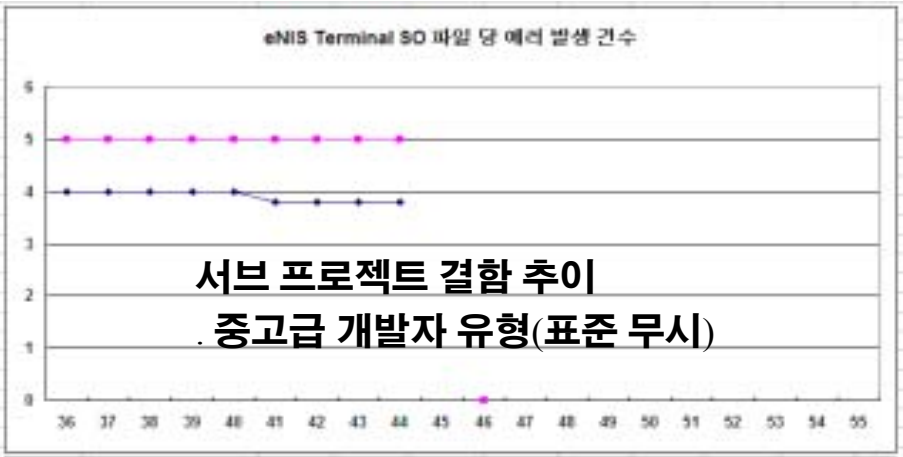
영 역		2006					2007						
		11/17	11/24	12/1	12/22	12/29	1/5	1/12	1/19	1/26	2/2	2/9	2/16
		36	37	38	41	42	43	44	45	46	47	48	49
eNIS	PRD(Java)	4.0	4.0	3.0	2.7	2.7	0.9	0.9		end			
	PRD(Jsp)	3.0	3.0	1.0	0.4	0.4	0.3	0.3		end			
	SCE(Java)	2.0	2.0	2.0	1.7	1.0	0.8	0.9	end				
	SCE(Jsp)	2.0	2.0	2.0	1.0	1.1	1.2	0.9	end				
	TES(Java)	4.0	4.0	4.0	3.8	3.8	3.8	3.8		end			
	TES(Jsp)	5.0	5.0	5.0	6.0	5.0	5.0	5.0		end			
	TRS(Java)	5.0	5.0	0.0	6.8	2.5	2.5	1.1					end
	TRS(Jsp)	4.0	4.0	1.0	0.6	0.4	0.4	0.2					end
	TPB(Java)	2.0	2.0	6.0	0.1	0.1	0.1	0.0	end				
	TPB(Jsp)	2.0	0.0	4.0	0.0	0.0	0.1	0.0	end				
	COA(Java)	8.0	2.0	16.0	0.6	0.7	0.5	0.3			end		
	COA(Jsp)	2.0	1.0	3.0	0.3	0.3	0.3	0.3			end		
	EQR(Java)	14.0	16.0	2.0	17.6	1.3	0.9	0.2					
	EQR(Jsp)	3.0	3.0	2.0	2.9	1.0	0.2	0.0					
	SAQ(Java)	1.0	1.0	0.0	0.3	0.7	0.2	0.2			end		
	SAQ(Jsp)	2.0	1.0	0.0	0.0	0.3	0.1	0.1			end		
	SPC(Java)	1.0	1.0	0.0	0.4	0.4	0.0	0.0			end		
	SPC(Jsp)	1.0	1.0	0.0	0.2	0.2	0.2	0.2			end		
	SAF(Java)				0.1	0.0	0.0	0.0					
	SAF(Jsp)				0.0	0.0	0.0	0.0					
	Biz 공통 (Java)	5.0	5.0	5.0	4.6	4.6	0.4	0.4					
	Biz 공통 (Jsp)	2.0		3.0	1.6	1.6	0.0	0.0					
	System 공통 (Java)	5.0	5.0	5.0	3.0	2.8	1.8	1.8					
	System 공통 (Jsp)	6.0	6.0	6.0	4.6	4.6	2.6	2.6					

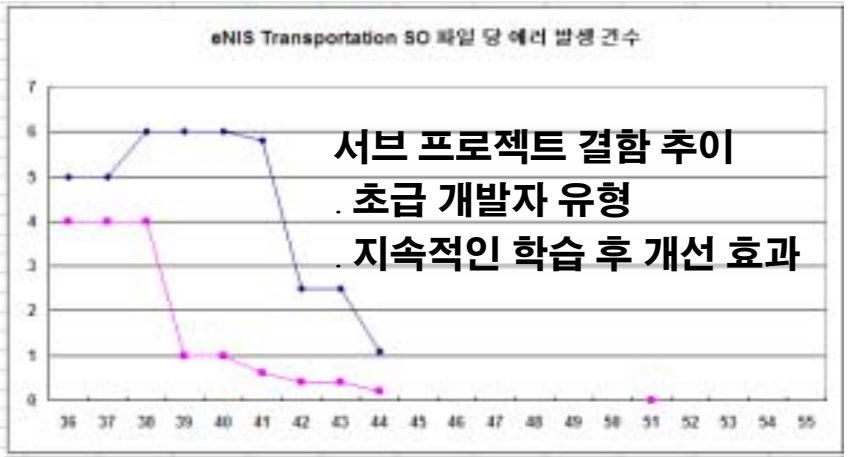


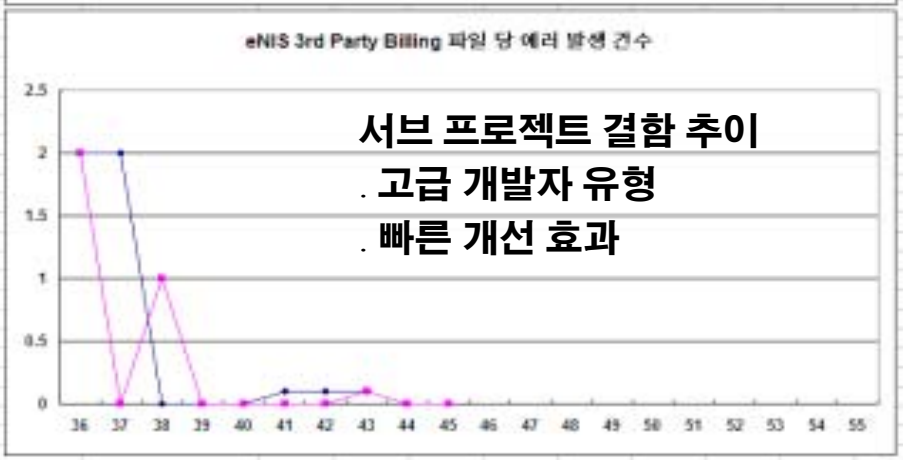


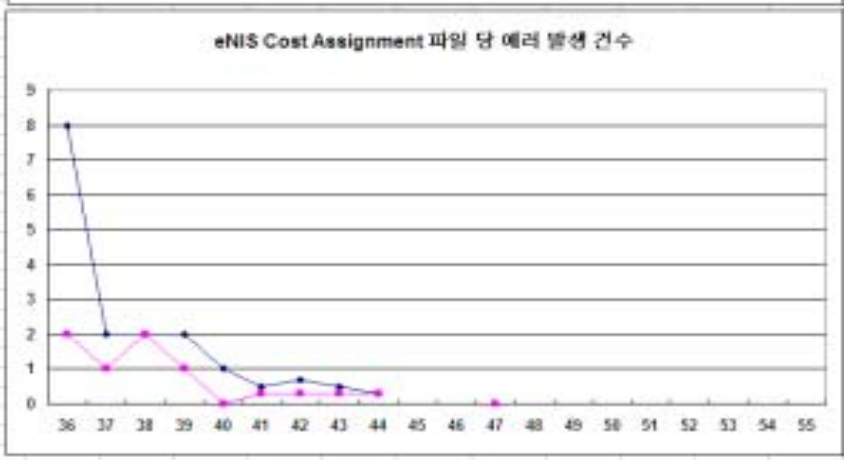
코드 검사 컨설팅 사례 (III)

- 파일당 결함 추이 및 개선 모니터링 – 주 1회









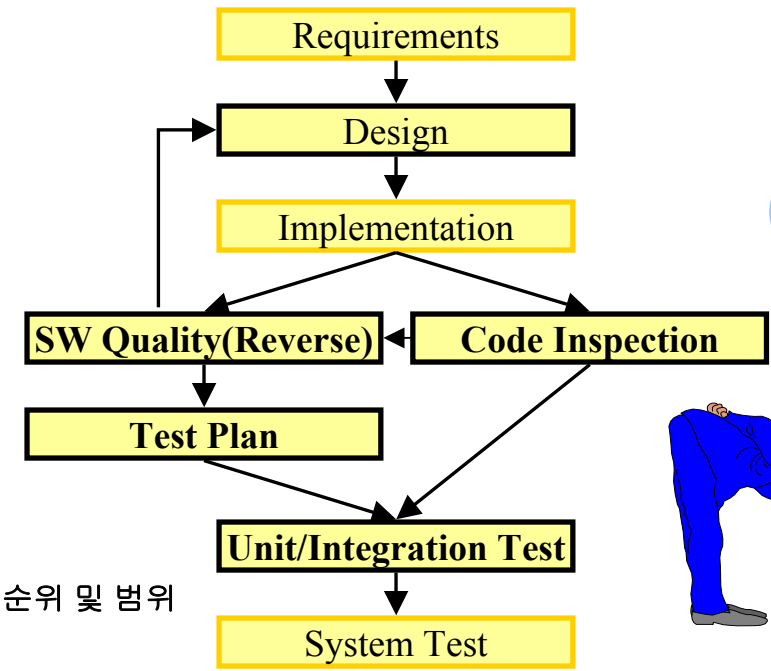
SW 품질 관리 및 코드 검사 효과

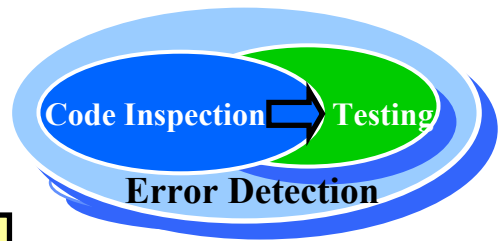
- SW 개발 표준화 및 에러 예방 -> SW 품질 향상
  - SW 구조의 단순화와 규격화되고 일관된 코딩 구조 유지
  - 개발 초기에 SW 구조 문제점 및 코딩 에러 등을 사전에 식별
- SW 개발 및 유지보수 비용 절감 -> SW 생산성 향상
- 단위/통합/성능 테스팅의 우선 테스트 범위 지원 -> SW 프로세스 성숙

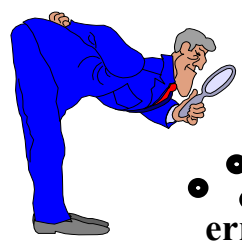
SW Lifecycle	Defect Ratios
Requirements	20%
Design	30%
Coding	35%
Documentation	5%
Bad Fixes	10%
Total	100%

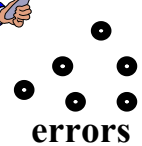
출처: J.E.Heiser, An Overview of Software Testing. IEEE TOSE 1997

고 위험성
구조 복잡성
유지보수성
성능성
테스팅 우선 순위 및 범위
…









규격화된 코드
코드 가독성
잠재적 에러
DB 인터페이스 에러
…

코드 검사 도구의 적용 효과 차이점

- 코드 검사 활용에 따른 코드 품질의 효과 차이점

품질/발주관리	코드 구현 중 적용	코드 구현 후 적용(검수)
코드 품질 검사	·품질 향상 –프로그램 잠재적 문제점 예방 ·생산성 향상 –검수 인력 및 비용의 절감	·품질 저하 –프로그램 잠재적 문제점 존재 ·생산성 저하 - 코딩 재작업 –개발 비용 및 기간 초과
발주 관리 제안	·RFP에 코드 검사 의무화	·SW 감리 시, 코드 검사 의무화

코드 검사 목적	코드 검사 효과	적용 시기 및 효과	
		개발 초기	개발 후
코딩 표준 준수	규격화된 우수한 코드 작성 일관된 코딩 구조와 스타일 유지 코드의 가독성 및 이해성 향상 코드의 유지 관리성 향상 및 비용 절감	●	◐
코드 에러 예방	SW 품질(성능) 향상 고급 코딩 기술과 프로그래밍 기능 사용 유도	●	◐

Legend:
● Fully efficiency
◐ Partially efficiency

If you cannot MEASURE it, you cannot IMPROVE it

㈜소프트4소프트

대전광역시 유성구 문지동 103-6 ICU 창업보육센터 T215호

Tel : 042-866-6632~3
Fax: 042-866-6626

구매 및 데모 문의 : sales@soft4soft.com
기술 및 일반 문의 : info@soft4soft.com

www.soft4soft.com