

소프트웨어 운영 및 테스트 실무

박민재

mjpark@daelim.ac.kr

수업 개요

- 산학협력강의
 - 융복합학습 구성
- 8월 26일 (오후) 첫 수업
 - 8월 28일 수업 보강
- 반학기제 수업
 - 각 수업은 1주일 단위
 - 정규/보강으로 구성
 - 추석/휴일 등으로 인한 보강계획 협의중
 - 오전 10시~, 오후 2시~
- 각 수업 공지
 - 학과 공지사항 및 단톡방 참고(주의)

강의 주차	월/일	성 명
1	8/28	박민재
2	8/28	박민재
3	9/4	박민재
4	9/4	박민재
5	9/11	오상일 (에스텍)
6	9/11	오상일 (에스텍)
7	9/18	박민재
8	9/18	박민재
9	9/25	원근주 (인제아이엔씨)
10	9/25	원근주 (인제아이엔씨)
11	10/2	김동환 (오토소프트)
12	10/2	김동환 (오토소프트)
13	10/9 10/16	박민재
14	10/9 10/16	박민재
15	10/23 (기말고사)	박민재

강의 주차	월/일	성 명
1	8/30	최종원 (아이엠테크놀로지)
2	8/30	최종원 (아이엠테크놀로지)
3	9/6	장재호 (토마토시스템)
4	9/6	장재호 (토마토시스템)
5	9/13 (보강일 필요)	박민재
6	9/13 (보강일 필요)	박민재
7	9/20	박민재
8	9/20	박민재
9	9/27	박민재
10	9/27	박민재
11	10/4 10/11	박민재
12	10/4 10/11	박민재
13	10/18	박민재
14	10/18	박민재
15	10/25	박민재

- 각 산업체 전문가 기반 학습
 - 각 교육 내용에 대해 이해
 - 회사에서 진행중인 업무 SW 산업 등에 대한 이해
 - 이해된 내용을 기반으로
- 소프트웨어 운영 및 테스트 실무
 - 소프트웨어 프로세스에 대해 이해 (소프트웨어 인증 등)
 - 소프트웨어 품질 관리 활동에 대한 이해
- 소프트웨어 개발 실무
 - Problem Based Learning
 - 문제 중심 학습
 - 현재 진행중인 작품활동 마무리
 - 작품 기반 가이드 문서 제출/작성
 - 참여형 티칭 가이드북 작성
 - 본 프로젝트(작품활동)을 위해, 활용할 수 있는 문서 작성
 - 개발 가이드/설치 가이드/운영 가이드/트러블 슈팅 가이드 등 필요한 문서를 선택하여 작성
 - 작성 문서를 서로 평가 (10월 10일까지 제출)

수업 개요

- 교육자료사이트- <http://cafe.naver.com/dlumjpark>

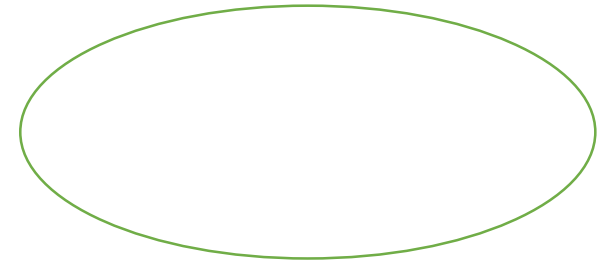
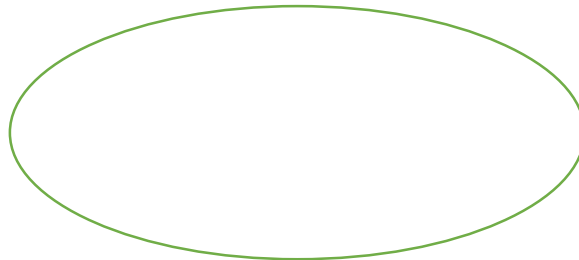
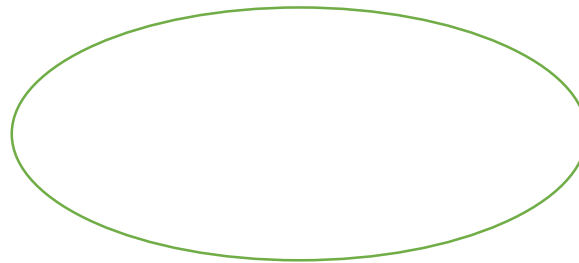
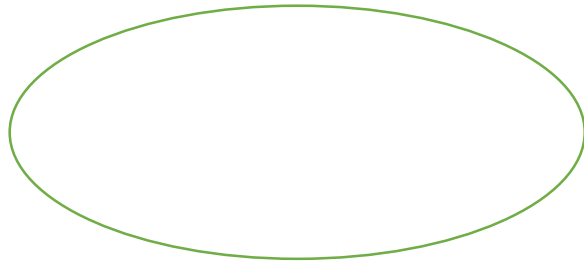
좋은 소프트웨어(프로그램 만들기)

- 각자 프로젝트를 진행하며, 무슨 활동을 했는지 이야기 해보자

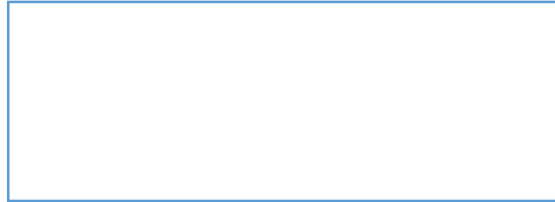
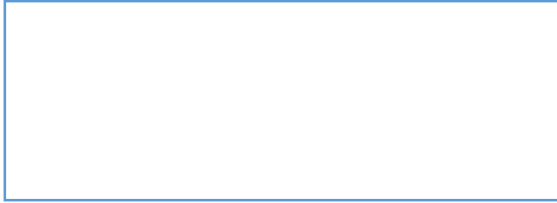
코딩

테스팅

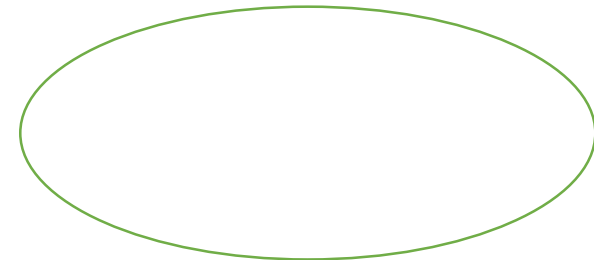
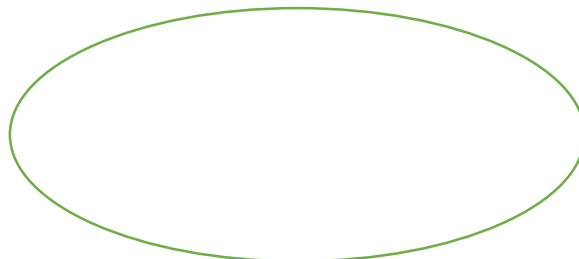
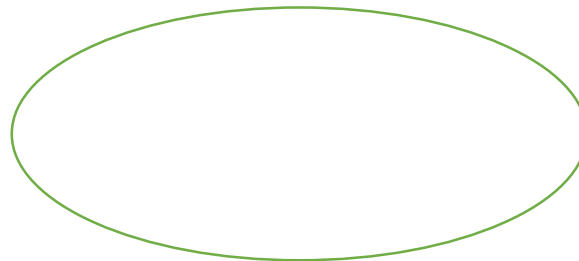
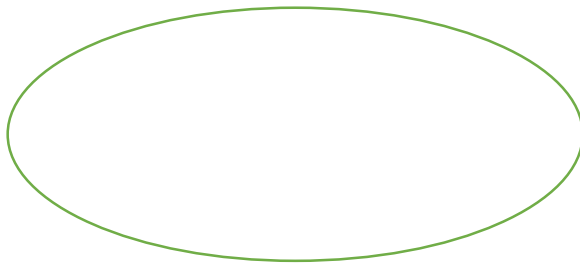
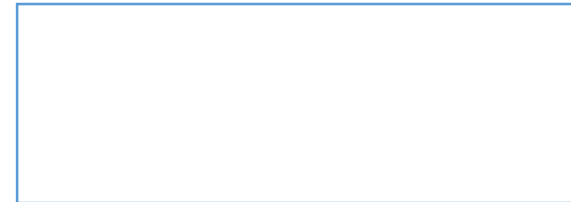
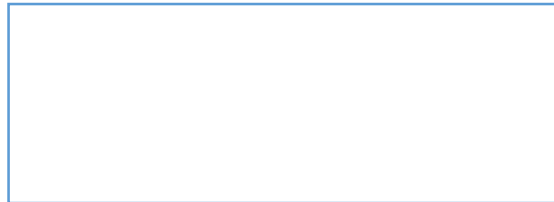
Github 사용



- 소프트웨어 품질?
 - 어떻게 하면 좋아지고, 왜 좋아질까?



Github 사용



- 출처: 중앙일보

“나로호, 자동 발사장치 소프트웨어 결함”

[중앙일보] 입력 2009.08.21 03:09 수정 2009.08.21 10:36 | [종합 2면](#) [지면보기](#) ▶



- 출처: 조선일보

[단독인터뷰]"BMW 화재, 하드웨어 아닌 소프트웨어 결함 가능성" 박용성 BMW 리콜 TF 결함조사 반장

전성필 기자 박성우 기자

▶ 본문듣기



입력 2018.08.29 14:16 | 수정 2018.08.29 17:47

BMW 화재, 근본 원인은 'EGR 작동 빈도 SW 설정'

정부 책임자 SW 문제 첫 인정

"하드웨어만 문제"라는 BMW 측 주장 뒤집어

냉각기 크랙에서 유출된 알콜 성분 냉각수도 조사 중

'BMW 차량의 발화 원인'을 조사중인 박용성 교통안전공단 기획조정실장이 29일 "BMW 차량에 결함을 발생시킨 근본 원인은 배기가스 재순환장치(EGR) 작동 빈도를 과도하게 늘린 소프트웨어(SW) 설정"이라고 분석했다. BMW 측은 차량 화재 원인으로 '하드웨어 결함'이라고 주장해왔다.

Google에 의해 종료된 광고입니다.

이 광고 신고

이 광고가 표시된 이유 ⓘ

Software Engineering (소프트웨어 공학)

- 소프트웨어 요구사항: 소프트웨어 요구 사항의 추출, 분석, 명세, 검증. 소프트웨어 요구공학(Software Requirements Engineering)분야가 독립적으로 존재함
- 소프트웨어 설계: 보통 전산 지원 소프트웨어 공학 (CASE) 도구로 이루어지고, UML과 같은 표준 형식을 사용.
- 소프트웨어 개발: 프로그래밍 언어로 소프트웨어를 구축.
- 소프트웨어 시험
- 소프트웨어 유지 보수: 소프트웨어 시스템은 때때로 처음 완료된 후 긴 시간이 지난 후에 문제를 일으켜 향상시켜야 할 필요가 있음.

Software Engineering (소프트웨어 공학)

- 소프트웨어 형상 관리: 소프트웨어 시스템은 매우 복잡하므로, 그 형상(버전과 소스 제어)이 표준화되고 구조적인 방법으로 관리 받아야 함.
- 소프트웨어 공학 관리: 프로젝트 관리에 매우 밀접하나, 다른 관리 분야와는 다른, 소프트웨어 고유의 미묘한 뉘앙스가 있음.
- 소프트웨어 개발 프로세스: 소프트웨어를 구축하는 과정에 관하여 실무 종사자들 사이에서는 열띤 논쟁이 오가고 있으며 주요한 패러다임은 애자일 프로세스와 폭포수 프로세스임.
- 소프트웨어 공학 도구(CASE)
- 소프트웨어 품질

Software Engineering (소프트웨어 공학)

V · T · E		Software engineering	[hide]
Fields	Computer programming · Requirements engineering · Software deployment · Software design · Software maintenance · Software testing · Systems analysis · Formal methods		
Concepts	Data modeling · Enterprise architecture · Functional specification · Modeling language · Orthogonality · Programming paradigm · Software · Software archaeology · Software architecture · Software configuration management · Software development methodology · Software development process · Software quality · Software quality assurance · Software verification and validation · Structured analysis		
Orientations	Agile · Aspect-oriented · Object orientation · Ontology · Service orientation · SDLC		
Models	Developmental	Agile · EUP · Executable UML · Incremental model · Iterative model · Prototype model · RAD · UP · Scrum · Spiral model · V-Model · Waterfall model · XP	
	Other	SPICE · CMMI · Data model · ER model · Function model · Information model · Metamodeling · Object model · Systems model · View model	
	Languages	IDEF · UML · USL · SysML	
Software engineers	Victor Basili · Kent Beck · Grady Booch · Fred Brooks · Barry Boehm · Peter Chen · Danese Cooper · Ward Cunningham · Tom DeMarco · Edsger W. Dijkstra · Delores M. Etter · Martin Fowler · Adele Goldstine · Margaret Hamilton · C. A. R. Hoare · Lois Haibt · Mary Jean Harrold · Grace Hopper · Watts Humphrey · Michael A. Jackson · Ivar Jacobson · Alan Kay · Nancy Leveson · Stephen J. Mellor · Bertrand Meyer · David Parnas · Trygve Reenskaug · Winston W. Royce · James Rumbaugh · Mary Shaw · Peri Tarr · Elaine Weyuker · Niklaus Wirth · Edward Yourdon		
Related fields	Computer science · Computer engineering · Project management · Risk management · Systems engineering		
<div><div></div><div>Category</div></div> · <div><div></div><div>Commons</div></div>			

품질 용어 중 가장 주요하게 묻는 차이 중 하나



- Validation
 - 확인?
- Verification
 - 검증?

Validation과 Verification의 차이의 이해

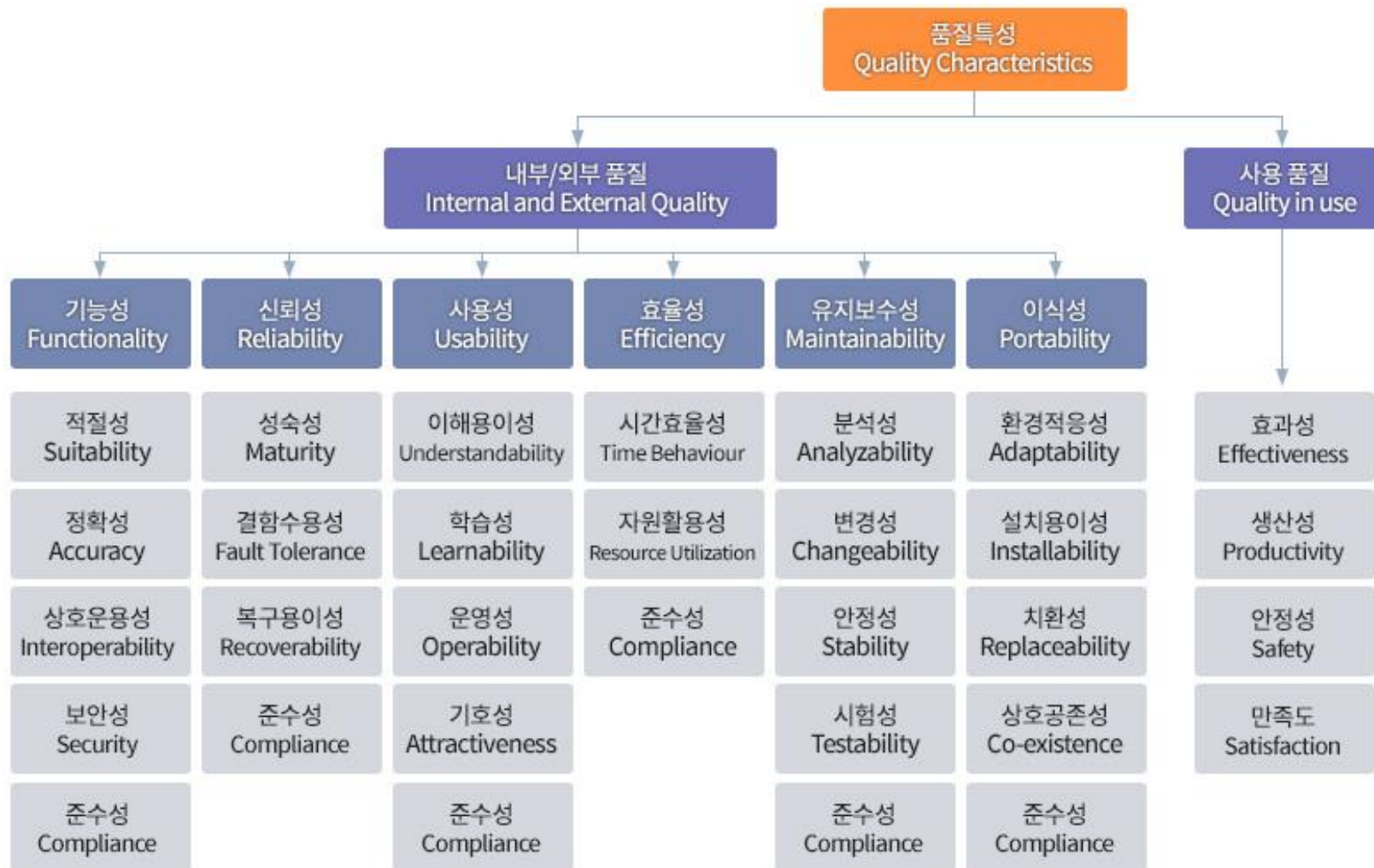
- Validation-유효성 (우리가 필요했던 프로그램인가? 요구사항에 맞는 프로그램인가? Are we building the right product?)
- Verification-검증 (올바르게 제작되었는가? 개발방법론이 제대로 평가되고 진행되었는가? Are we building the product right?)

Criteria	Verification(검증)	Validation(확인)
정의	최종 산출물이 아닌 중간 단계 산출물이 정의된 요구사항을 만족하는지 평가하는 과정.	개발 과정의 최종 단계에서 개발된 소프트웨어 제품이 비즈니스 요구 사항을 만족하는지 평가하는 과정.
평가 관점	'개발 요구 명세서의 내용대로 소프트웨어가 개발되고 있는가?'	'사용자가 요구하는 소프트웨어를 개발하고 있는가?'
평가 항목	계획서, 요구사항 정의서, 설계문서, 프로그램 코드, 테스트 케이스... 등등	소프트웨어 제품
평가 활동	Review, Walk throughs, Inspections.	testing

소프트웨어 품질: ISO/IEC 9126

- 소프트웨어 품질의 특성을 정의하고 품질 평가의 Metrics를 정의한 국제 표준
- 사용자 관점에서 본 소프트웨어의 품질 특성에 대한 표준
 - ISO (International Organization for Standardization) – 국제 표준화 기구
 - IEC (International Electrotechnical Commission) – 국제 전기기술협회
- 사용자, 평가자, 시험관, 개발자 모두에게 소프트웨어 제품의 품질을 평가하기 위한 지침의 마련 필요
- 평가대상 소프트웨어의 품질을 직접 측정하기 위해 필요한 평가 Metrics의 준비
- 소프트웨어의 품질을 객관적이고 계량적으로 평가할 수 있는 기본적 틀 필요

- 기능성
 - 요구된 기능이 소프트웨어에 있는가?
- 신뢰성
 - 얼마나 신뢰할만 한가?
- 사용성
 - 사용하기 쉬운가?
- 효율성
 - 얼마나 효율적인가?
- 유지보수성
 - 소프트웨어를 변경하기 쉬운가?
- 이식성
 - 다른 환경에 얼마나 쉽게 이식할 수 있는가?

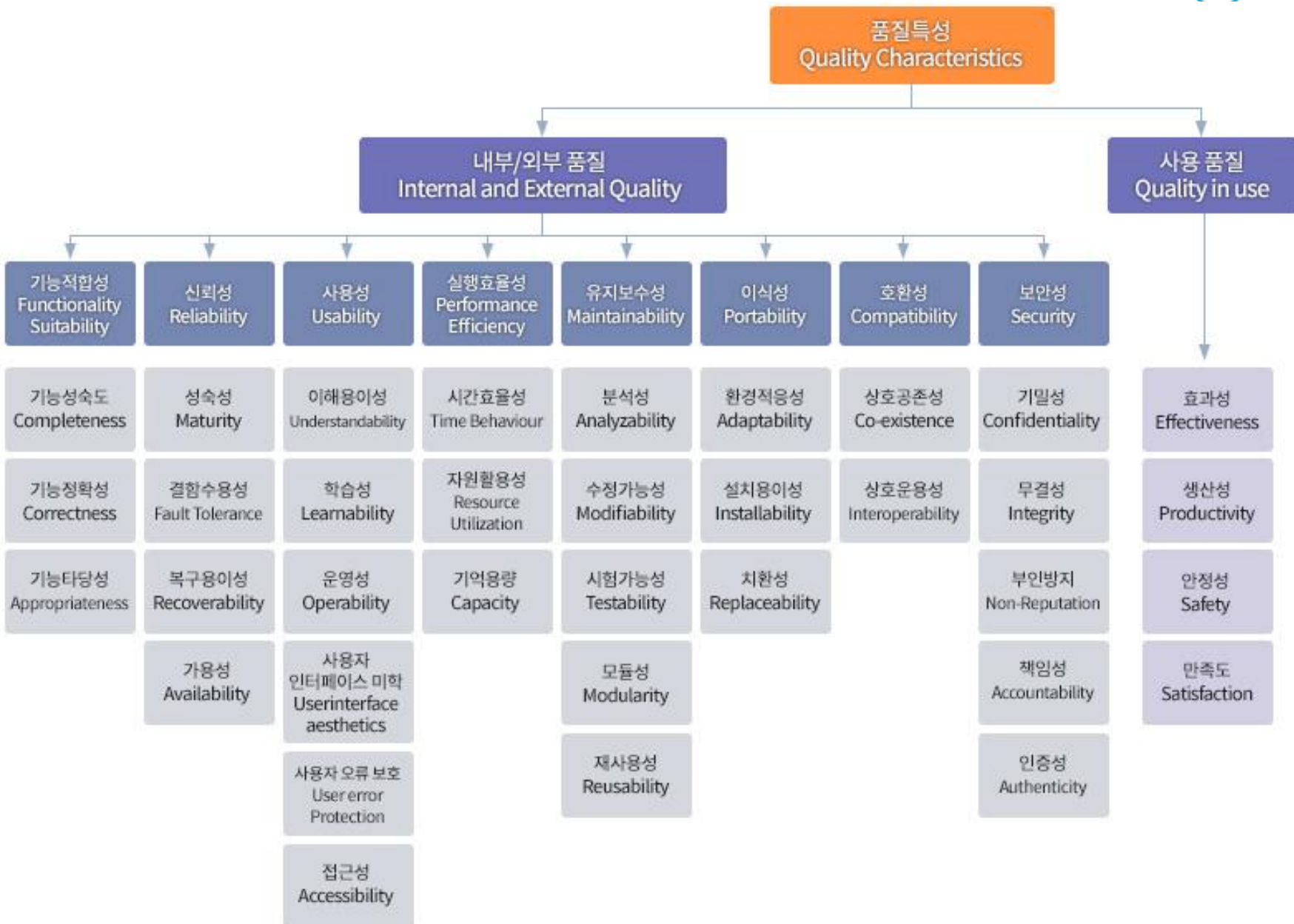


- 외부지표 (External Metrics)와 내부지표 (Internal Metrics)
- **외부지표 (External Metrics)**
 - 실행 가능한 SW, 시스템을 시험, 운영 또는 관찰을 통하여 시스템을 구성하고 있는 일부분으로부터 추출된 소프트웨어 제품의 측정에 사용
 - 사용자, 평가자, 시험자 및 개발자에게 시험 수행이나 운영 중에 소프트웨어 제품에 대한 품질을 평가
- **내부지표 (Internal Metrics)**
 - 설계, 코딩도중에 실행할 수 없는 SW 제품(명세서, 원시코드 등)에 대하여 적용
 - 설계 상 요구되는 외부품질을 성취하기 위하여 ISO 9126-3에 규정
 - 사용자, 평가자, 시험자 및 개발자가 소프트웨어 제품의 품질을 평가할 수 있도록 도움
 - SW 제품이 완성되기 이전단계에서 외부적으로 발생할 수 있는 문제점들을 사전에 검증

• 사용품질지표 (Quality in use Metrics)

- 사용자 관점의 품질
- SW 자체가 가지고 있는 속성이라기 보다는 SW를 활용한 결과에 대한 측정
- 사용자와 SW 품질특성이 결합된 효과를 표현
- 사용자 편의성, 기호성 등과 같은 사용성(Usability)에 한정된 것이 아닌 전반적인 품질 특성에 의해 영향
 - 효율성(Effectiveness): 주어진 목표달성을 위한 정확성(Accuracy), 완전성(Completeness)을 측정
 - 생산성(Productivity, Task Efficiency): Task 효율성과 관련되어 소요된 자원을 측정
 - 안전성(Safety): 특정 위험의 발생 가능서의 수준을 측정
 - 만족도(Satisfaction): 제품의 사용에 대한 속성을 측정

- 기존의 ISO/IEC 9126에서 ISO/IEC 25010으로 개정
- 주 특성이 기존 6개에서 8개로 증가
 - 기존: 기능성, 신뢰성, 사용성, 유지보수성, 이식성, 효율성
 - 변경: 기능 적합성, 실행 효율성, 호환성, 사용성, 신뢰성, 보안성, 유지보수성, 이식성
- 부특성 일부 증가
 - 기존 27개 -> 31개 증가
 - 일부 항목 삭제
 - 각 주 특성의 준수성(Compliance) 항목은 전체 삭제



- ISO/IEC 25010 품질 특성 모델
- 개발된 최종 산출물인 소프트웨어 제품이 사용자의 의도대로 기능을 수행하는지 평가



① 일반 개요(ISO/IEC 14598-1)

- 소프트웨어 제품의 품질 평가를 수행하기 위한 일반적인 개요로 범위와 용어를 정의
- ISO/IEC 14598과 ISO/IEC 9126의 관계 설명
- 개발자/평가자/구매자가 사용할 수 있는 평가 프로세스와 평가 모듈에 대한 로드맵 제공

② 계획과 관리(ISO/IEC 14598-2)

- 품질 평가 척도를 프로젝트 성격에 따라 선정하고 적용하기 위한 제품 품질 측정 계획의 준비와 구현에 대해 다루고 있다.
- 제품 평가 기능을 지원하기 위한 요구 사항과 안내 지침을 포함한 전체적인 사항을 제공

③ 개발자를 위한 프로세스(ISO/IEC 14598-3)

- 소프트웨어 개발 단계에서 알아야 할 개발자 준수 사항으로, 개발자가 개발 과정 및 최종 소프트웨어 제품의 품질 평가에 사용할 수 있는 방법을 제공

④ 구매자를 위한 프로세스(ISO/IEC 14598-4)

- 소프트웨어 제품을 구매하기 위한 계획을 수립할 때 사용하는 구매자용 프로세스
- 구매자가 상용 소프트웨어 제품을 구매하는 과정에서 소프트웨어 제품의 품질 평가에 사용할 수 있는 방법 제공

⑤ 평가자를 위한 프로세스(ISO/IEC 14598-5)

- 품질 전문가를 위한 프로세스로, 평가자가 개발 과정 및 최종 소프트웨어 제품의 품질 평가를 수행할 때 사용

⑥ 평가 모듈(ISO/IEC 14598-6)

- 개발자/구매자/평가자가 소프트웨어 제품의 품질을 평가할 때 사용할 수 있도록 평가 모델에 대한 기본적인 가이드와 이론적인 모델 제공
- 평가 모듈을 개발하여 문서화하고 검증하는 지침 제공

- ISO/IEC 12119 information technology-software packages-quality requirements and testing
 - 패키지 소프트웨어의 일반적인 제품 품질 요구 사항 및 테스트를 위한 국제 표준 규격
 - 규격 내용: 품질, 지침, 세부 인증
 - 요건 사항: 명확화, 유사 문서 정의, 변경(가능)성, 환경 명세, 보안성
 - 구성: 제품 설명서, 사용자 문서, 실행 프로그램
 - 첫 번째 평가 대상:
 - - 패키지 소프트웨어
 - - 제품 문서, 사용자 문서, 프로그램/데이터에 명시된 요구 사항
 - 두 번째 평가 대상: 패키지 SW와 최종 제품 및 중간 산출물을 포함하는 수주 개발 SW
 - 셋째 평가 대상: 패키지 SW와 최종 제품 및 개발/유지보수 과정을 포함한 수주 개발 SW
 - 테스트 대상: 소프트웨어 제품 명세서, 사용자 매뉴얼, 사용자 요구 테스트 결과서 등
 - 주요 테스트 평가 항목: 기능성, 신뢰성, 사용성, 이식성 등
 - 평가 척도: ISO 9126-2, ISO 9126-3(품질 메트릭), ISO 12119(테스트 방법)

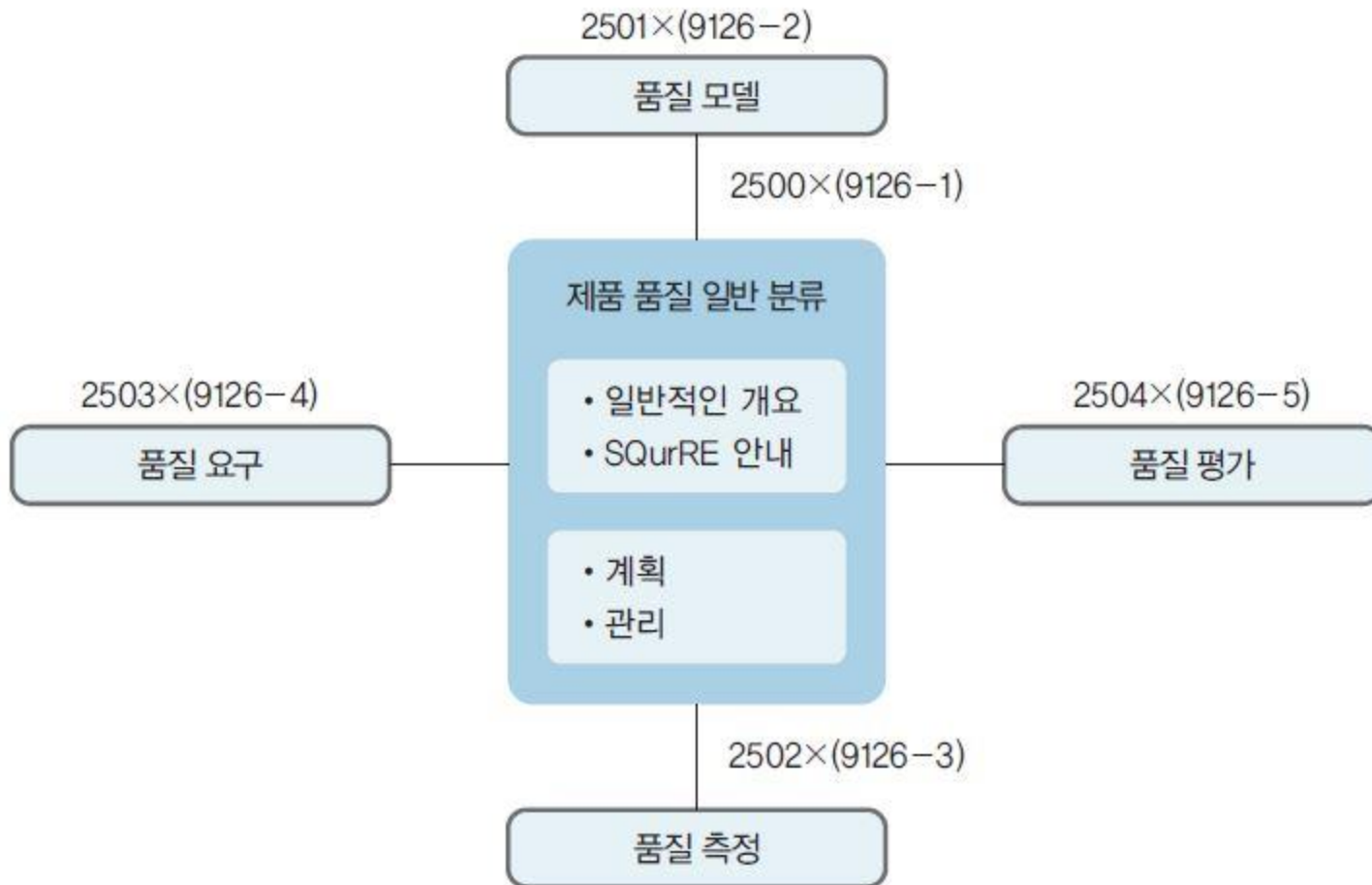
ISO/IEC 25000

- ISO/IEC 25000, SQuaRE Software Quality and Requirement Evaluation

- 사용자들에게 유용하도록 여러 표준 문서를 통합하고 재구성하여 만든 국제 표준 문서
- 소프트웨어 품질 평가 모델로부터 시작해 전체적인 품질 평가를 위한 표준 방안 제시

모델 구분	구성	내용
2500×(9126-1)	제품 품질 일반 분류	SQuaRE에 대한 개요, 전체에 대한 계획과 관리
2501×(9126-2)	소프트웨어 품질 모델	품질 모델 및 품질 사용 안내
2502×(9126-3)	소프트웨어 품질 측정	기본 메트릭스, 내/외부 메트릭스, 사용 중 품질 메트릭스, 평가 모듈의 문서화
2503×(9126-4)	소프트웨어 품질 요구	품질 요구 사항
2504×(9126-5)	소프트웨어 품질 평가	품질 평가 프로세스에 관한 개요 및 개발자, 구매자, 평가자 관점의 품질 평가 프로세스

ISO/IEC 25000 구성도



- 법적근거
 - 소프트웨어산업 진흥법 14조(품질인증)
 - 소프트웨어산업 진흥법 시행령 제10조(품질인증의 실시)
 - 소프트웨어 품질인증의 세부기준 및 절차(과학기술정보통신부 고시)
- ISO 국제 표준을 기준으로 SW의 기능성, 신뢰성, 효율성, 사용성, 유지보수성, 이식성, 성능, 상호운용성 연동성 및 적합성을 시험/테스트하여 인증을 부여

PRODUCT
(제품 자체의 품질)

개발방법과 절차가 정확해도 제품자체의 품질을 보증하지 못함

대표적 적용모델

- ISO 25051 (Quality Requirement and Testing)
- ISO 9126 (Software Product Quality)
- ISO 14598 (Software Product Evaluation)

PROCESS
(개발 공정의 품질)

소프트웨어 조직의 개발방법과 절차가 정확하면 고품질 제품생산

대표적 적용모델


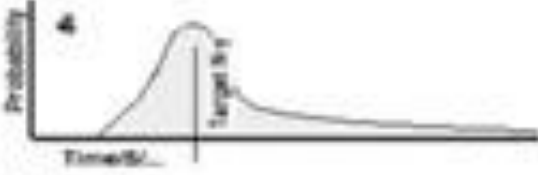

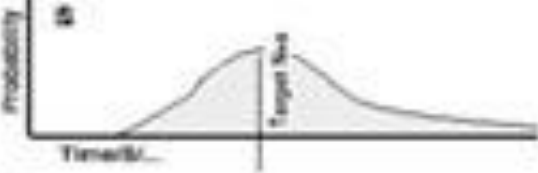

- CMMI (Capability Maturity Model Integration)
- ISO 15504 (SPICE-SW Process Improvement & Capability determination)

Software Process의 이해

- 흔히,
- CMM / CMMI
- ISO~~~
- SP
- 이러한 용어로 이야기 하는 것이 Software Process

- Capability Maturity Model
- 소프트웨어를 개발, 유지보수하는 과정에서 이루어지는 모든 일련의 과정을 효율적이고 효과적으로 개선하는 프로세스적인 접근 방법

CMM의 이해

단계	프로세스 특징	예상되는 성과
5	최적 프로세스 개선에 초점을 맞춤	
4	관리 프로세스가 측정되고 통제됨	
3	정의 조직차원의 프로세스 정의	
2	반복 성공한 경험이 다음 프로젝트에서 반복	
1	초기 프로세스가 비공식적이고 예측할 수 없음	

- Level 1 : 초기(Initial)
 - 1단계는 초기 단계로 가장 낮은 성숙도 수준을 의미하며 프로세스가 거의 정의되어 있지 않아 프로젝트의 성공은 **개인의 노력에 따라 달라지게 됩니다**. 많은 조직들이 이 단계에 해당한다고 볼 수 있습니다.
- Level 2 : 반복(Repeatable)
 - 2단계는 프로젝트 레벨에서 프로세스가 재사용되고 있음을 의미합니다. **성공적인 프로젝트에 활용됐던 유사한 프로세스를 다시 사용**하는 것입니다.
- Level 3 : 정의(Defined)
 - 3단계는 2단계에서 한발 더 나아가 **조직 레벨의 표준 프로세스가 존재함**을 의미합니다. 즉, 조직에서 수행되는 모든 프로젝트는 소프트웨어를 개발, 관리하기 위해 조직의 표준 소프트웨어 프로세스를 사용하게 됩니다. 이때 조직의 표준 프로세스는 각 프로젝트의 특성에 맞게 수정되어 사용됩니다.
- Level 4 : 관리(Managed)
 - 4단계에서는 **소프트웨어 프로세스와 품질에 대한 상세한 측정**이 이뤄집니다. 모든 프로젝트에 대한 중요한 소프트웨어 프로세스 활동의 생산성과 품질이 정량적으로 측정되고 프로세스 관리 측면의 강화가 이뤄지는 단계입니다.
- Level 5 : 최적(Optimizing)
 - 가장 상위의 단계인 5단계에서는 **지속적인 프로세스 개선**이 이뤄지게 됩니다. 신기술을 결합해 프로세스의 최적화가 이루어지고 전 조직에 최적화된 프로세스가 다시 적용되는 최상위 단계입니다.

CMM의 이해

- CMM 인증기관인 미국 카네기멜론 대학 소프트웨어기술연구소가 2006년부터 CMM 인증을 중단하고, CMMI만 인증
- CMM은 엄밀히 말하면 SW-CMM(Capability Maturity Model for Software)
 - 바꾸어 말하면 소프트웨어뿐만 아니라 다른 분야의 CMM도 있음을 의미
 - 시스템공학을 위한 SE-CMM
 - 점차적으로 복잡해지는 소프트웨어 환경에 효과적으로 대응하기 위하여 개인의 능력을 개발하고 동기부여를 강화하며 조직화하는 수준을 측정하고 개선하기 위한 모델인 P-CMM(People CMM)
 - 소프트웨어 획득 과정을 중점적인 대상으로 하여 성숙도를 측정하고 개선하기 위한 모델인 SA-CMM(Software Acquisition CMM)
- 다양한 CMM 모델로 인한 포괄적 틀 필요
 - CMMI로 진화

CMM이냐 CMMI이냐가 중요한가?

- 중요하지 않다.
- 소프트웨어 프로세스의 능력에 따라 소프트웨어 부문의 품질과 생산성이 좌우됨
- 기업 및 기관의 경쟁력이 정보 기술 활용 능력에 따라 좌우되는 정도가 심화되는 추세를 감안한다면 프로세스 능력 향상을 위한 노력은 매우 중요하며 CMM을 적용하는 것은 소프트웨어 프로세스 능력 향상을 위한 시작이라는 점을 이해해야 함

- 첫째, 프로세스의 **실질적 개선에 초점을 뒀다** 한다.
 - CMM의 어느 단계에 속하는가 하는 것은 CMM의 공인 평가자(Lead Assessor)가 평가한 결과일 뿐입니다.
 - SEI가 공식적으로 평가 받은 조직의 품질 수준에 대해 전적으로 보증하는 것이 아닙니다.
 - 이는 CMM은 기업 스스로 프로세스 수준을 측정하고 개선할 수 있는 틀을 제공하는 데 주안점을 두고 있기 때문입니다.
- 국내 일부 기업들의 경우, 상위 수준의 CMM 단계를 인증하는 것에만 무게를 두고 있는 경우가 있는데 이는 바람직하지 못하다는 의미입니다.
- 자체 기업의 소프트웨어 프로세스 요건에 적합한 모델을 선택하고 이를 적용함으로써 실질적으로 프로세스 능력을 향상시키기 위해 노력해야 합니다.
- 수준에 대한 평가는 프로세스 능력 향상의 결과로서 받아들일 수 있어야 진정으로 프로세스 수준 향상의 효과를 볼 수 있습니다.

- 둘째, 가능한 빨리 시작하자
 - SEI의 통계조사에 의하면 CMM의 특정 단계에서 다음 단계로 진화하는 데는 1년에서 2년 정도의 기간이 소요된다고 합니다.
 - 그것은 새로운 프로세스를 정의하여 조직에 정착시키는 것은 교육과 훈련 등의 변화 관리가 필요하기 때문입니다.
 - 따라서 늦게 시작하면 시작할수록 경쟁 기업과의 격차는 더욱 벌어지게 됩니다.
 - 일단 가능한 범위 내에서 소프트웨어 프로세스 개선 노력을 시작해야 합니다. 소프트웨어프로세스를 전담하는 조직(SEPG, Software Process Engineering Group)을 구성하여 준비를 시작하는 것도 대안이 될 수 있습니다.

- 셋째, **지속적인 추진이 중요합니다.**
 - 소프트웨어 프로세스 개선은 한 번의 노력으로 마무리되지 않습니다.
 - 프로세스를 정의하고 적용하고 결과를 측정하여 피드백하는 정기적인 노력이 필요합니다.
 - 즉 소프트웨어 프로세스 개선은 일회성 프로젝트로 효과를 볼 수 있는 것이 아니라
는 것입니다.
 - 물론 새로운 프로세스를 정의하기 위한 프로젝트를 수행하는 것은 필요합니다.
 - 프로젝트를 통해 새로운 프로세스 체계를 만들었다 하더라도 이를 적용하고 문제
점을 발견하는 과정이 충분하게 수행되지 못한다면 원하는 결과를 기대할 수 없습
니다.
 - 얼마나 알고 있느냐 보다는 얼마나 행하느냐가 중요한 것입니다.

CMMI Staged Maturity Levels



- CMMI: Capability Maturity Model Integration
능력 성숙도 모델 통합
- **본래 CMM이라는 것이 있었고, 이것을 개선**
- 직역하면 능력성숙모델통합?
- (Project) CMMI
 - 프로젝트를 하는 능력이 얼마나 성숙 되었는지를 체크(측정)하기 위한 모델
- SI (System Integration) 프로젝트
- 정보시스템 개발 프로젝트
- 통합(I)는 CMM을 만든 사람들이 그들의 방법론을 통합한다는 의미
- 즉, CMMI는 정보시스템 개발 프로젝트를 수행하는데 얼마나 잘 할 수 있는지를 측정하는 모델임

- CMM 인증기관인 미국 카네기멜론 대학 소프트웨어기술연구소가 2006년부터 CMM 인증을 중단하고, CMMI만 인증
- CMM은 엄밀히 말하면 SW-CMM(Capability Maturity Model for Software)
 - 바꾸어 말하면 소프트웨어뿐만 아니라 다른 분야의 CMM도 있음을 의미
 - 시스템공학을 위한 SE-CMM
 - 점차적으로 복잡해지는 소프트웨어 환경에 효과적으로 대응하기 위하여 개인의 능력을 개발하고 동기부여를 강화하며 조직화하는 수준을 측정하고 개선하기 위한 모델인 P-CMM(People CMM)
 - 소프트웨어 획득 과정을 중점적인 대상으로 하여 성숙도를 측정하고 개선하기 위한 모델인 SA-CMM(Software Acquisition CMM)
- 다양한 CMM 모델로 인한 포괄적 틀 필요
 - CMMI로 진화

레벨 1 : 개인 플레이

- 프로젝트 팀의 실력대로 개발을 수행하는 단계
- 미리 정의된 절차가 전혀 없기 때문에 개개인의 실력대로 일을 함
- 절차가 없다고, 분석, 설계, 구현.. 등의 과정을 거치지 않는 것은 아님
 - 기본적으로 분석하고, 설계하고 개발하고 테스트는 함
- CMMI의 방법론 소개시, CMMI와 같은 방법을 안한다고, 우리가 절차가 없는 것은 아닌데? 라고 하는 경우가 있음

레벨 2 : 프로젝트 플레이

- 정의된 프로세스가 있어서 이를 따르면서 프로젝트를 수행
- 프로젝트별로 각각 실력대로 정의된 프로세스 를 가지고 진행
- 1,2단계에서 '실력대로'라는 말을 쓰는 것은 다른 프로젝트에서는 적용되지 않을 수 있다는 것을 의미
- PM과 구성원의 실력대로 만들어진 절차에 따라 진행하기 때문
- 정의된 절차가 있음
 - 2단계에 있는 경우 어느 프로젝트는 성공하고 어느 프로젝트는 실패하는데 이를 조직적으로 개선하지 못하는 단계

레벨 3 : 팀 플레이

- 레벨 3는 정의된 프로세스가 조직 차원에서 규정
- 조직 내에서 공통된 관점으로 일을 함
- 사실 조직화 되었다는 점이 매우 중요
 - **CMMI 레벨 3를 전사조직으로 인증받았다는 것은 대단히 큰 의미**
- 실력이 제 각각이 아니며, 특별히 처지는 프로젝트는 없어진다고 봐도 무방
- 전 조직원이 동일한 시각으로 프로젝트를 수행
 - 팀웍이 있다는 것을 의미
 - 불확실성도 어느 정도 제거되고 있다고 의미
- 레벨 3에서는 조직화된 절차를 프로젝트에 맞추어 조정하여 진행
 - 모든 프로젝트를 똑같이 수행한다는 것이 아니라 필요없다고 판단되는 것은 하지 않는다는 것을 의미
 - 여기서 중요한 것은 필요성 여부를 조직적으로 판단한다는 것
 - PM이 임의로 절차를 정하게 되면 레벨2

레벨 4 : 데이터 기반 플레이

- 레벨 4는 데이터로 관리하는 단계
- 절차는 조직화 되었으며, 프로젝트별로 여러 경우가 생김
 - 이러한 데이터를 집계하여 분석
 - 특이사항을 제거
 - 프로젝트를 예측
- 정량화된 데이터로 어떤 단계에서 어떤 위험이 있고, 이 정도로 진행되는 경우라면 결과는 이렇게 나올 것이라고 예측
 - 위험을 회피
- 과거 데이터에 기반하는 것이고 기술적 차이, 구성원의 차이를 반영하지 못하므로 보조적인 관리 지표가 될 수 있음

레벨 5 : 꿈의 플레이

- 레벨 5는 정량화된 데이터에 근거하여 개선점을 지속적으로 찾아 스스로 발전할 수 있게 되는 단계
- 정성적인 목표가 아니라 정량적인 데이터로 문제점을 찾아내고 개선하고 혁신하는 단계
- 이를 스스로 하는 꿈의 단계