Software Quality Management

Woo Young Moon, Ph.D.
Professional Engineer Information Management
PMP, CFPS

Mobile 010-8709-1714
Wooyoungmoon@soongsil.ac.kr

Software Quality?

- 2005년 ComputerWorld (Hildreth,S. “Buggy Software: Up from a Low Quality Quagmire 저품질 수렁에서 벗어나기”, 2005, https://www.computerworld.com/article/2557403/buggy-software--up-from-a-low-quality-quagmire.html)
  - "나쁜 소프트웨어는 컴퓨터를 사용하는 거의 모든 조직을 괴롭히며 컴퓨터 가동 중지 시간 동안 근무 시간 손실, 데이터 손실 또는 손상, 판매 기회 손실, 높은 IT 지원 및 유지 관리 비용, 낮은 고객 만족도를 초래

- 2006년 InfoWorld (Foster, E. “Quality Culprits,” 2006, https://www.infoworld.com/article/2647071/the-sorry-state-of-software-quality.html)
  - 품질 문제가 개선되지 않았다는 보고서 ” the sorry state of software quality” 소프트웨어 품질의 유감스러운 상태: 품질은 나빠진다는 점은 동의하지만 누가 책임을 지는지에 대해서 동의하지 않음.

- 오늘날 소프트웨어 품질은 여전히 문제이지만 누가 책임을 져야 할까요?
  - 고객은 부주의한 개발 관행이 저품질 소프트웨어로 이어진다고 주장하면서 개발자를 비난합니다.
  - 계속되는 변경과 소프트웨어가 완전히 검증되기 전에 소프트웨어를 납품하도록 강요하는 불합리한 납기로 고객(및 기타 이해 관계자)을 비난.

Quality

- American Heritage Dictionary는 품질을 다음과 같이 정의.
  - “a characteristic or attribute of something, 어떤 것의 특성이나 속성”

- 소프트웨어의 경우 세 가지 종류의 품질이 발생.
  - Quality of Design: 요구 사항, 사양 및 시스템 설계의 충족 정도.
  - Quality of Conformance: 주로 구현에 초점을 맞춘 문제로 요구사항과 성능 목표를 충족하는 정도.
  - User Satisfaction= compliant product + good quality + delivery within budget and schedule (준수 상품+ 좋은 품질 + 예산 및 일정 내 배송)
    - 품질은 제품, 서비스, 경험 등 어떤 것의 우수성 또는 우월성의 정도라고 직관적으로 정의
    - 고품질 제품은 의도한 기능을 효과적이고 효율적으로 수행하고 신뢰할 수 있고 내구성이 있으며 결함이나 오류가 없는 제품

What is Software Quality?

- Quality
  - The degree of conformance to explicit or implicit requirements and customer expectations.
    - 명시적 또는 암시적 요구 사항 및 고객 기대에 대한 준수 정도
  - An effective software process applied in a manner that creates a useful product that provides measurable value for those who produce it and those who use it.(The Business Value of Quality)
    - 제품을 생산하는 사람과 사용하는 사람에게 측정 가능한 가치를 제공하는 유용한 제품을 만드는 방식으로 적용되는 효과적인 소프트웨어 프로세스

Effective Software Process

- **효과적인 소프트웨어 프로세스**
  - **고품질 소프트웨어 제품을 구축하려는 모든 노력을 지원하는 인프라를 구축.**
  - **프로세스의 관리 측면은 품질 저하의 주요 원인인 프로젝트 혼란을 방지하는 데 도움이 되는 견제와 균형.**
  - **소프트웨어 엔지니어링 관행을 통해 개발자는 고품질 소프트웨어를 구축하는 데 중요한 문제를 분석하고 견고한 솔루션을 설계.**
  - **마지막으로 변경 관리 및 기술 검토와 같은 지원 활동은 소프트웨어 엔지니어링 관행의 다른 부분만큼 품질과 많은 관련이 있음.**

Useful Product

- 유용한 제품
  - 최종 사용자가 원하는 콘텐츠, 기능 및 특징을 제공.
  - 그러나 중요한 것은 이러한 자산을 신뢰할 수 있고 오류 없는 방식으로 제공한다는 것.
  - 유용한 제품은 항상 이해 관계자가 명시적으로 언급한 요구 사항을 충족.
  - 또한 모든 고품질 소프트웨어에서 기대되는 일련의 암시적 요구 사항 (예: 사용 용이성)을 만족.

(Adding) Value

- 가치 추가
  - 고품질 소프트웨어는 소프트웨어 제품의 생산자와 사용자 모두에게 가치를 더함으로써 소프트웨어 조직과 최종 사용자 커뮤니티에 혜택을 제공.
  - 소프트웨어 조직은 고품질 소프트웨어가 더 적은 유지 보수 노력, 더 적은 버그 수정, 더 적은 고객 지원을 필요로 하기 때문에 부가 가치 제공.
  - 사용자 커뮤니티는 애플리케이션이 일부 비즈니스 프로세스를 가속화하는 방식(빠르게 제공하는 방식) 으로 편리한 기능을 제공하기 때문에 부가 가치 제공.
  - 최종 결과.
    - (1) 더 많은 소프트웨어 제품 수익
    - (2) 애플리케이션이 비즈니스 프로세스를 지원할 때 더 나은 수익성
    - (3) 비즈니스에 중요한 정보의 가용성 향상

Quality의 실용적인 관점은?

- 초월적 관점
  - 품질은 즉시 인식할 수 있지만 명시적으로 정의할 수는 없다고 주장.
  - 개인의 주관적인 판단과 경험, 소비자의 기대와 만족도를 결정
- 사용자 관점
  - 최종 사용자의 특정 목표 측면에서 품질. 제품이 이러한 목표를 충족하면 품질을 보여줌.
- 제조자의 관점
  - 제품의 고유한 특성 측면에서 품질을 정의합니다. 제품이 사양을 준수하면 품질을 보여줌.
- 제품 관점
  - 품질이 제품의 고유한 특성(예: 기능 및 특징)과 연결될 수 있음을 시사.
- 가치 기반 관점
  - 고객이 제품에 대해 얼마를 지불할 의향이 있는지에 따라 품질을 측정.
  - 실제로 품질은 이러한 모든 관점과 그 이상을 포함.

Quality Dimensions

- Quality Dimensions.
  - 제품이나 서비스의 품질을 평가할 때 사용되는 다양한 요소들을 의미.
  - 이러한 요소들은 고객이 제품이나 서비스를 평가하고 구매하는 데 있어서 중요한 역할.
- Eight Dimensions (David Garvin, “Competing on the Eight Dimensions of Quality”, Harvard Business Review, 1987)
  - 성능 품질 : 소프트웨어가 최종 사용자에게 가치를 제공하는 방식으로 요구 사항 모델의 일부로 지정된 모든 콘텐츠, 기능 및 특징을 제공하나?
  - 기능 품질: 소프트웨어가 처음으로 최종 사용자를 놀라게 하고 기쁘게 하는 기능을 제공하나?
  - 신뢰성: . 소프트웨어가 오류 없이 모든 기능? 필요할 때 사용할 수 있나? 오류가 없는 기능을 제공하나?
  - 적합성: 소프트웨어가 애플리케이션과 관련된 로컬 및 외부 소프트웨어 표준을 준수하나? 사실상의 설계 및 코딩 규칙을 준수하나? 예를 들어, 사용자 인터페이스가 메뉴 선택 또는 데이터 입력에 대해 승인된 디자인 규칙을 준수하나?
  - 내구성: 소프트웨어를 유지 관리(변경)하거나 수정(디버깅)할 수 있습니까? 변경으로 인해 시간이 지남에 따라 오류율이나 신뢰성이 저하됩니까?
  - 서비스 용이성: 허용 가능한 짧은 시간 내에 소프트웨어를 유지(변경)하거나 수정(디버깅)할 수 있습니까? 지원 직원이 변경하거나 결함을 수정하는 데 필요한 모든 정보를 얻을 수 있습니까?
  - 미학: 우리 대부분은 미적 실체가 특정 우아함, 고유한 흐름, 정량화하기 어렵지만 그럼에도 불구하고 독특한 흐름 및 분명한 "존재감"을 가지고 있다는 데 동의할 것입니다.
  - 지각: 어떤 상황에서는 품질에 대한 인식에 영향을 미치는 일련의 편견이 있다. 예를 들어 과거에 낮은 품질을 제공한 업체의 소프트웨어에 대한 편견, 뛰어난 평가를 받은 업체의 소프트웨어에 대한 편견 등.

Measuring Quality

- 품질 측정
  - 일반적인 Quality Dimensions및 factors는 구체적인 용어로 응용 프로그램의 품질을 평가하는 데 적합하지 않음.
  - 프로젝트 팀은 각 애플리케이션 품질 요소가 만족된 정도를 평가하기 위해 일련의 목표 질문을 개발해야 함.
  - 소프트웨어 품질에 대한 주관적인 척도는 개인적인 의견에 지나지 않음.
  - 소프트웨어 metrics은 품질의 일부 징후에 대한 간접적인 측정을 나타내며 소프트웨어 품질 평가를 정량화하려는 시도임.

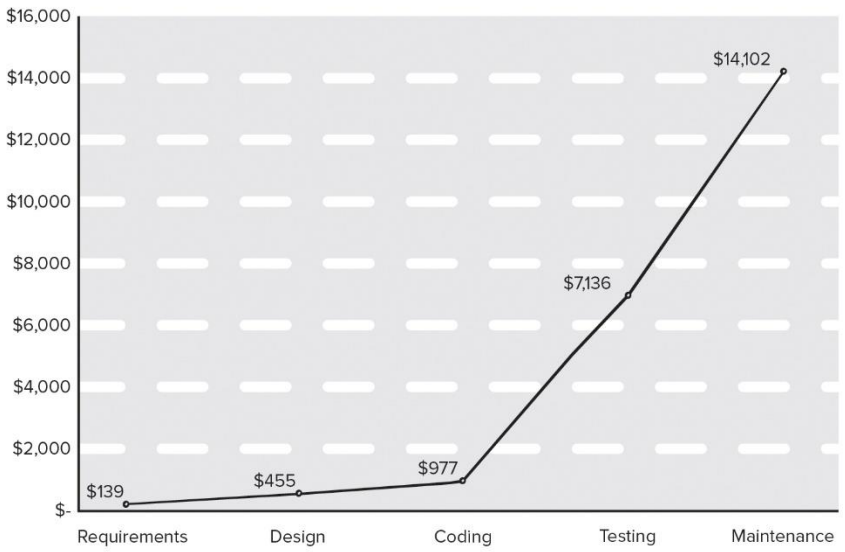
The Software Quality Dilemma

- Design by Contract(Venners, B. 2003)
  - 품질이 좋지 않은 소프트웨어 시스템을 생산하면 아무도 그것을 사고 싶어하지 않기 때문에 손해를 봄.
  - 반면에 절대적으로 완벽한 소프트웨어를 구축하기 위해 무한한 시간, 극도로 많은 노력, 막대한 돈을 들인다면 완성하는 데 시간이 너무 오래 걸리고 생산하는 데 비용이 너무 많이 듬. 어쨌든 폐업.
  - 시장 진입을 놓쳤거나 단순히 모든 자원을 소진.
  - 그래서 업계 종사자들은 제품이 평가와 같이 즉시 거부되지 않을 만큼 충분히 훌륭하지만 너무 많은 완벽주의와 너무 많은 작업의 대상이 되어 너무 오래 걸리거나 완료하는 데 비용이 너무 많이 들지 않는 magical middle ground를 찾기 위해 노력.

"Good Enough" Software

- 충분히 좋은 소프트웨어
  - 최종 사용자가 원하는 고품질 기능(function and feature) 을 제공하지만 동시에 알려진 버그가 포함된 더 모호하거나 특수한 다른 기능(function and feature)을 제공.
- "충분히 좋다"에 대한 논쟁.
  - "충분히 좋다"는 것은 일부 애플리케이션 도메인과 일부 주요 소프트웨어 회사에서 작동할 수 있다는 것이 사실.
    - 결국 회사가 대규모 마케팅 예산을 가지고 있고 충분한 사람들이 버전 1.0을 구매하도록 설득할 수 있다면 그들을 독점하는데 성공한 것.
  - 작은 회사에서 일한다면 이 철학에 주의하십시오. "충분히 좋은/만족스러운 "(버그가 있는) 제품을 제공하면 회사의 평판이 영구적으로 손상될 위험.
    - 나쁜 소문으로 인해 판매가 급감하고 회사가 망할 수 있기 때문에 버전 2.0을 제공할 기회를 얻지 못할 수도 있음.
  - 특정 애플리케이션 도메인(예: 실시간 임베디드 소프트웨어, 하드웨어와 통합된 애플리케이션 소프트웨어)에서 작업하는 경우 회사가 비용이 많이 드는 소송에 노출될 수 있음.

Cost



Source: Boehm, Barry and Basili, Victor R., "Software Defect Reduction Top 10 List," IEEE Computer, vol. 34, no. 1, January 2001.

- The graph plots requirements, design, coding, testing, and maintenance on the x axis and the relative costs on the y–axis.
- The relative costs to repair a defect are as follow: requirements at $139, design at $455, coding at $977, testing at $7,136, and maintenance at $14,102.
- 오류 또는 결함을 찾고 수리하는 상대적 비용은 prevention에서 detection, Internal failure, external failure 비용으로 이동함에 따라 극적으로 증가.

Cost of Quality

- 예방 비용(Prevention Cost) .
  - 품질 계획(quality planning)
  - 공식 기술 리뷰(formal technical review)
  - 테스트 장비(test equipment)
  - 훈련 (training)
- 내부 실패 비용(Internal Failure Cost) .
  - 재작업 (rework)
  - 수리 (repair)
  - 고장 모드 분석 (failure model analysis)
- 외부 실패 비용(External Failure Cost)
  - 불만 해결 (complaint resolution)
  - 제품 반품 및 교환(product return and replacement)
  - 헬프 라인 지원 (help line support)
  - 보증 업무(warranty work)

Cost of Quality

- 평가 비용(Appraisal Costs) .
  - 품질을 평가하고 결함을 식별하기 위해 수행하는 활동에 대한 비용
  - 기술 검토 수행 (conducting technical reviews)
  - 데이터 수집 및 측정 지표 평가 (data collection and metrics evaluation)
  - 테스트 및 디버깅 (testing and debugging)
  - 예시:
    - 코드 리뷰, 정적 코드 분석, 단위 테스트, 통합 테스트, 시스템 테스트, 성능 테스트, 보안 테스트 등 평가 비용은 곧바로 결함을 방지하거나 수정하는 비용은 아님
    - 장래에 발생할 수 있는 더 큰 문제를 방지하는 예방적 성격의 비용이며, 결과적으로 전체적인 개발 비용을 절감하는 데 도움.

Quality and Risk

- "People bet their jobs, their comforts, their safety, their entertainment, their decisions, and their very lives on computer software. It better be right."

  "사람들은 자신의 직업, 안락함, 안전성, 엔터테인먼트, 결정, 그리고 자신의 삶을 컴퓨터 소프트웨어에 걸고 있습니다. 그리고 이것이 옳길 바란다"

- Example)
  - 2000년 11월 한 달 동안 파나마의 한 병원에서 다양한 암 치료를 받는 동안 28명의 환자가 감마선 과다투여.
  - 그 후 몇 달 동안 이 환자 중 5명이 방사선 중독으로 사망했고 다른 15명은 심각한 합병증을 일으킴.
  - 이 비극의 원인은?
  - 미국 회사에서 개발한 소프트웨어 패키지는 각 환자에 대한 수정된 방사선 선량을 계산하기 위해 병원 기술자에 의해 수정.
  - 추가기능을 제공하는 소프트웨어를 개조한 3명의 파나마 내과의사들은 2급살인혐의로 기소되었으며, 미국 기업은 2개국에서 심각한 소송에 직면.

과실 및 책임

- 과실 및 책임, 이런 이야기는 너무 일반적입니다. 정부 또는 기업체는 주요 소프트웨어 개발자 또는 컨설팅 회사를 고용하여 요구 사항을 분석한 다음 일부 주요 활동을 지원하는 소프트웨어 기반 "시스템"을 설계 및 구성.
- 시스템은 주요 기업 기능(예: 연금 관리) 또는 일부 정부 기능(예: 의료 관리 또는 국토 안보)을 지원할 수 있음.
- 작업은 양측의 최선의 의도로 시작되지만 시스템이 제공될 때쯤에는 상황이 나빠짐.
- 시스템이 늦고 원하는 기능을 제공하지 못하고 오류가 발생하기 쉬우며 고객 승인을 충족하지 못함.
- 소송이 계속.

반복되는 대형 공공 SW사업 오류 원인은?

| 관습화된 불공정 관행과 여전히 낮은 SW산업에 대한 인식

컴퓨팅 | 입력 :2022/11/03 09:22 수정: 2022/11/03 14:12

사회보장정보시스템의 전산 오류가 지속되며 현장의 고통이 가중되고

사중와해야 할 대규모 업무를 수기 작성하는 등 공무원의 업무 피로도는 극에 달하고 있으며, 불규칙한 급여와 연금 지급으로 취약계층은 생계의 위협까지 받고 있다.

대규모 공공 시스템에 장애가 발생한 것은 어제오늘 일이 아니다. 코로나19 백신 사전예약 시스템은 한동안 먹통 현상을 겪었으며, 원격수업 역시 접속 불가 이 발생했다.

Low Quality Software

- Low Quality Software는 개발자와 최종 사용자 모두에게 위험을 증가시킴.
- 시스템이 늦게 배송되고 기능을 제공하지 못하며 고객 기대치를 충족하지 못하는 경우 소송이 발생.
- Low Quality Software는 해킹하기 쉽고 일단 배포된 애플리케이션의 보안 위험을 증가시킬 수 있음.
- 설계 단계에서 품질(보안, 신뢰성, 의존성)에 초점을 맞추지 않고 안전한 시스템을 구축할 수 없음.
- Low Quality Software는 아키텍처 결함과 구현 문제(버그)를 포함할 가능성이 있음.

Impact of Management Decisions

- 관리적인 의사결정의 영향
  - 추정 결정(Estimation Decisions)
    - 비합리적인 납기 추정으로 인해 팀은 제품 품질 저하로 이어질 수 있는 지름길을 택하게 됨.
    - 고품질의 소프트웨어를 추구하는 활동은 생략,제품 품질은 저하
    - 납기가 무리인 경우 제품 품질의 저하를 하지 않기 위한 노력 – 더 많은 시간이 필요한지 설명, 할당된 시간에 만들 수 있는 하위 집합의 기능 제시
  - 일정 결정 (Schedule Decisions)
    - 프로젝트 일정을 만들 때 작업 종속성에 주의를 기울이지 못함
  - 위험 지향적 의사 결정(Risk-oriented decisions)
    - 위험을 모니터링하는 메커니즘을 구축하기보다 발생하는 각 위기에 대응하면 제품 품질이 저하될 수 있음.
    - 잘못될 것이 없다는 개념을 전제로 개발계획을 수립하거나 잘못되어가는 것에 대한 대처방법이 없는 것.

How to achieve software quality?

- 소프트웨어 품질은 우수한 프로젝트 관리 및 견고한 엔지니어링 실행의 결과
  - 소프트웨어 공학 + 프로젝트 관리 기술 + 품질관리 조치 + 소프트웨어 품질보증
- 소프트웨어 공학 방법
  - 해결해야 할 문제에 대한 이해
    - 설계 품질 및 적합성(요구 사항, 설계, 구현)
  - 문제에 부합한 설계 중 아키텍처 결함 제거
- 품질 관리 (Quality Control)
  - Series of inspections, reviews, and tests (full life cycle)
- 품질보증 (Quality Assurance)
  - 사전 결정을 내리는 데 필요한 관리 데이터를 제공하는 데 사용되는 감사 및 보고 절차

How to achieve software quality?

- 프로젝트 관리
  - 의사결정의 영향

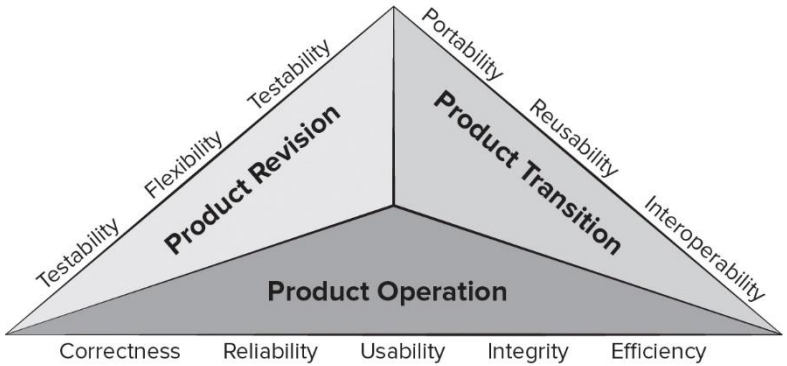
    1) Project Plan : 프로젝트 매니저가 납기 달성 여부를 verify하기 위해 estimation을 사용하는 경우

    2) 계획 종속성(schedule dependency)을 이해하고 팀은 최소 노선 사용(shortcuts) 유혹에 저항할 경우

    3) 위기 계획이 실행되어 문제가 혼란을 야기하지 않을 경우
  - 프로젝트 계획에는 품질과 변경 관리를 위한 명백한 기술이 포함

McCall Quality Factors

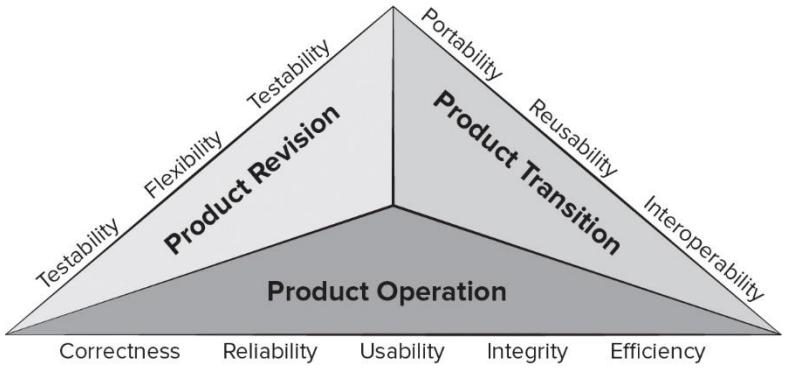
- McCall, Richard, Walters (Factors in Software Quality, 1977)



- A pyramidical illustration displays McCall's quality factors.
- The faces of the pyramid are labeled: product revision, product transition, and product operation.
- Product operation verifies: correctness, reliability, usability, integrity, and efficiency.
- Product transition verifies: portability, reusability, and interoperability.
- Product revision verifies: testability, flexibility, and testability

McCall Quality Factors

- McCall, Richard, Walters (Factors in Software Quality, 1977)
  - 품질인자 (Quality Factor)
    - 소프트웨어 품질을 관리적인 측면에서 표현, 품질 목표 설정하는데 적용
  - 품질기준 (Quality Criteria) :
    - 품질인자 세분화, 최종소프트웨어 제품의 품질을 측정하거나 품질여부를 결정할 수 있도록 하는 속성
  - 품질척도 (Quality Metric)
    - 품질 기준을 정량적으로 평가하기 위한 수단



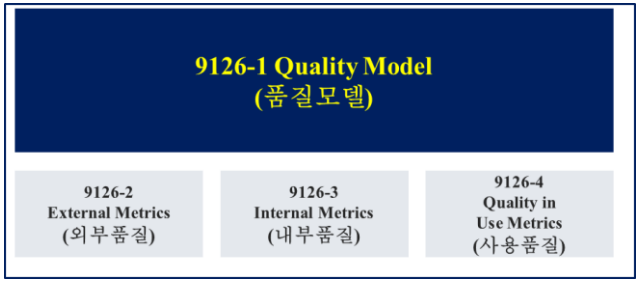
McCall Quality Factors

- **McCall, Richard, Walters (Factors in Software Quality, 1977)**

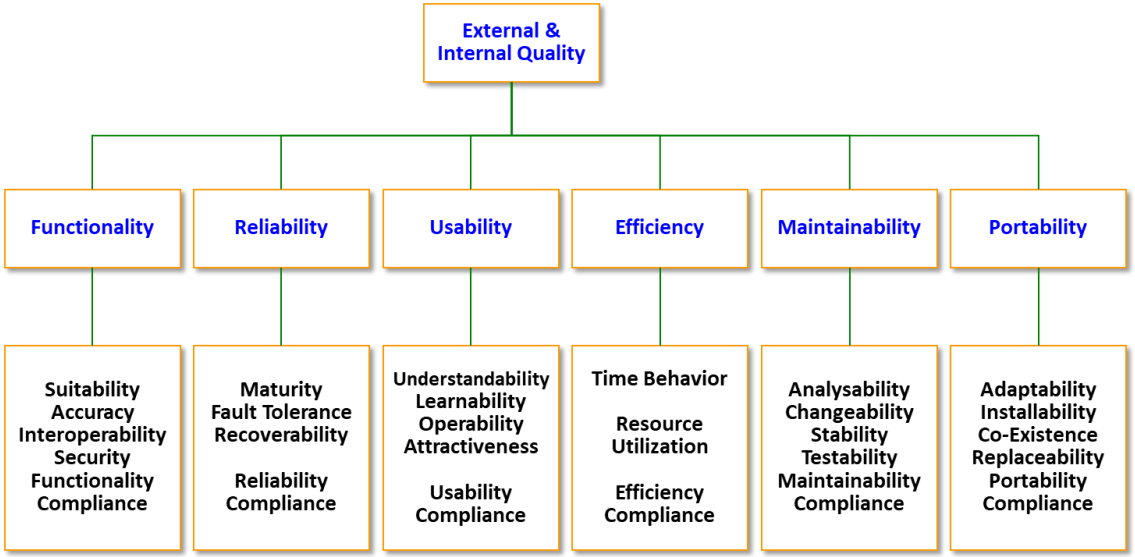
품질인자 (Q.Factor)	품질기준 (Q.Criteria)	정의 (Definition)
Product Operation 제품 운용	정확성 신뢰성 사용성 무결성 효율성	사용자 요구사항 충족, 설계되고 구현된 성질 오류없이 정확하고 일관된 결과, 의도된 기능 기능을 수행하기 위한 컴퓨터 리소스 /코드 양 안전을 위한 비인가자로부터의 SW/DATA보호 학습, 입력 출력 등 사용하기 용이한 성질
Product Revision 제품 개정	유지보수성 시험성 유연성	오류를 찾아 수정하는데 드는 비용, 성질 의도하는 기능을 수행하는지 입증하는성질 프로그램의 변경/보완하기 쉬운정도
Product Transition	이식성 재사용성 상호운용성	새로운 시스템으로 이동하기 쉬운 성질 다른 응용분야에서 재사용할 수 있는 범위 다른 시스템과 결합하는데 드는 노력정도

ISO 9126 Quality Factors

- ISO/IEC 9126 모델
  - 품질모델, 외부품질, 내부품질, 사용품질



  - 내부/외부 품질
    - 기능성, 신뢰성, 사용용이성, 효율성, 유지보수성, 이식성



Quality of Attributes-II

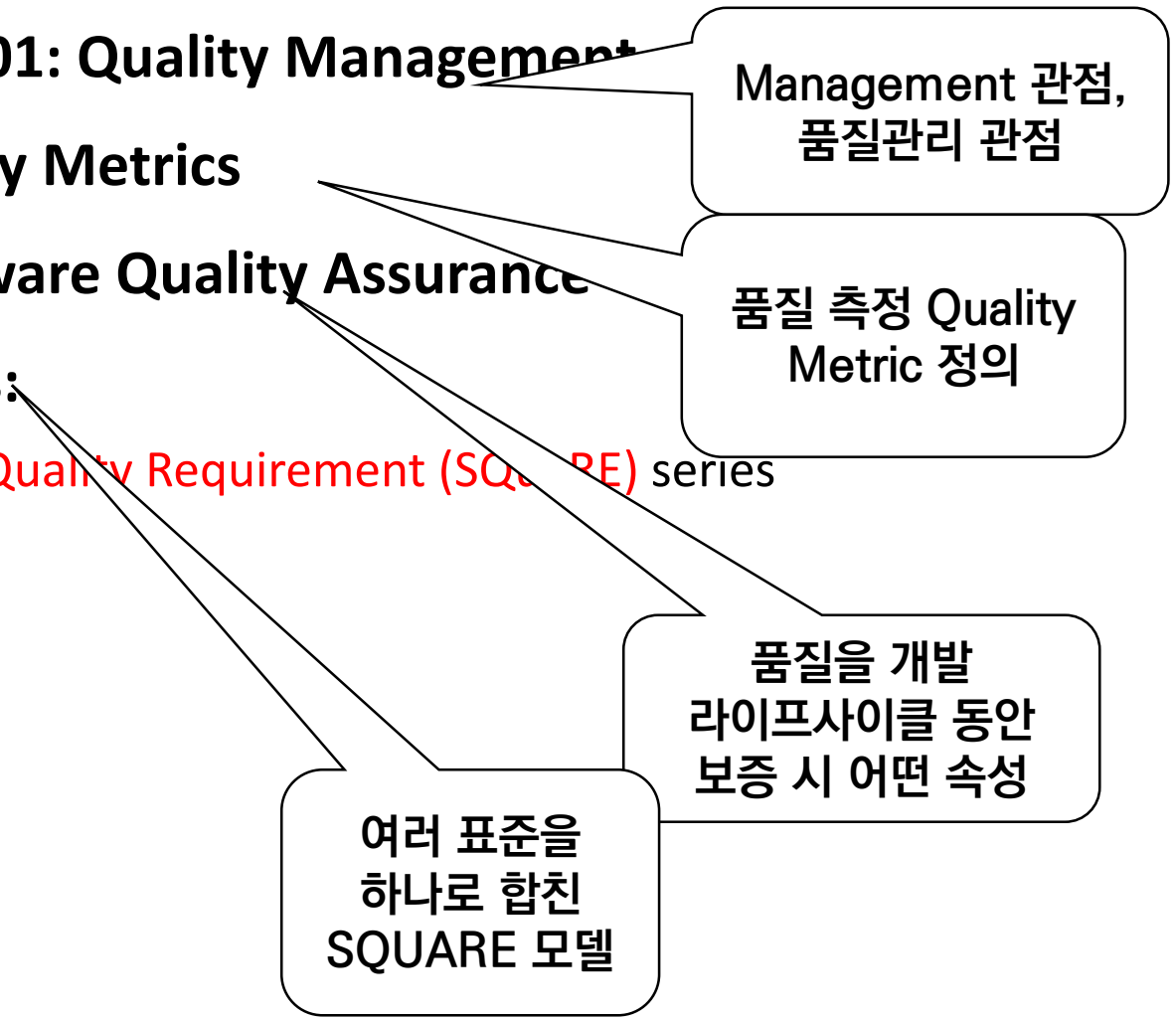
- 비기능적 요구사항이 아닌 품질 속성이라는 용어를 사용하는 이유는 무엇입니까?
  - 비기능적이라는 용어는 오해의 소지가 있는 용어
    - 요구 사항을 두 개의 깔끔한 절반으로 명확하게 나눌 수 있음을 암시.
    - 품질 속성은 필연적으로 시스템의 기능과 연결
    - Try to describe an “–ility” without describing functionality.

Quality of Attributes-III

- **Quality Attributes (품질속성)**
  - Performance
  - Reusability
  - Reliability
  - Stability
  - Security
  - Extendibility
  - Portability
  - Usability
  - Scalability
  - Modifiability
  - Data integrity & more

Quality Standards

- **ISO 9001 or ISO 14001: Quality Management**
- **ISO/IEC 9126: Quality Metrics**
- **IEEE 730-2014: Software Quality Assurance**
- **ISO/IEC 25000 series:**
  - System and Software Quality Requirement (SQuaRE) series



Quality Standards

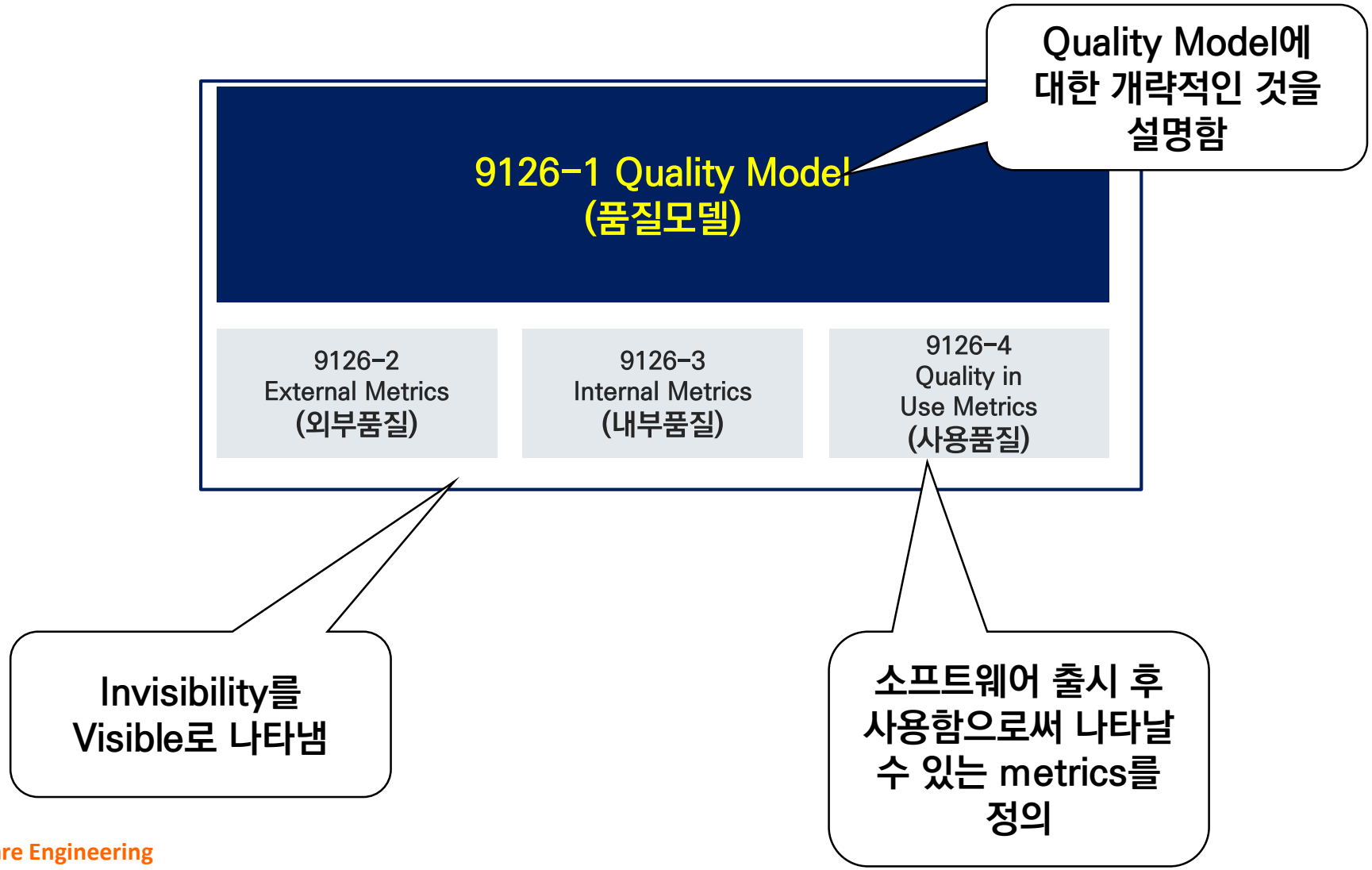
- 소프트웨어 품질 표준 모델

구분	Standards
제품 품질	ISO9126: Information Technology-Software Quality Characteristics and Metrics)은 소프트웨어 품질 특성과 척도에 관한 지침으로 고객 관점에서 소프트웨어에 관한 품질 특성과 품질 부특성을 정의 ISO12119: 소프트웨어 제품의 품질을 요구사항을 규정 ISO14598 : 소프트웨어 제품의 품질을 측정하거나 평가하는데 필요한 방법과 절차 ISO25000 : 소프트웨어 제품 품질 평가를 위한 프레임워크
SW 프로세스 품질	ISO12207 :기본 생명주기, 지원 생명주기, 조직 생명주기 SPICE (ISO 15504) :Process 평가를 위한 프레임워크 CMMI: 소프트웨어와 시스템 공학 능력/성숙도 평가 모델
품질 경영	ISO9000 : 국제표준화 기구 기술위원회에서 제정한 품질경영시스템의 국제규격 6 시그마: 100만개의 업무요소에서 3.4개의 결함을 목표로 하는 모토롤라의 경영 혁신 기법

ISO 9126 Quality Model

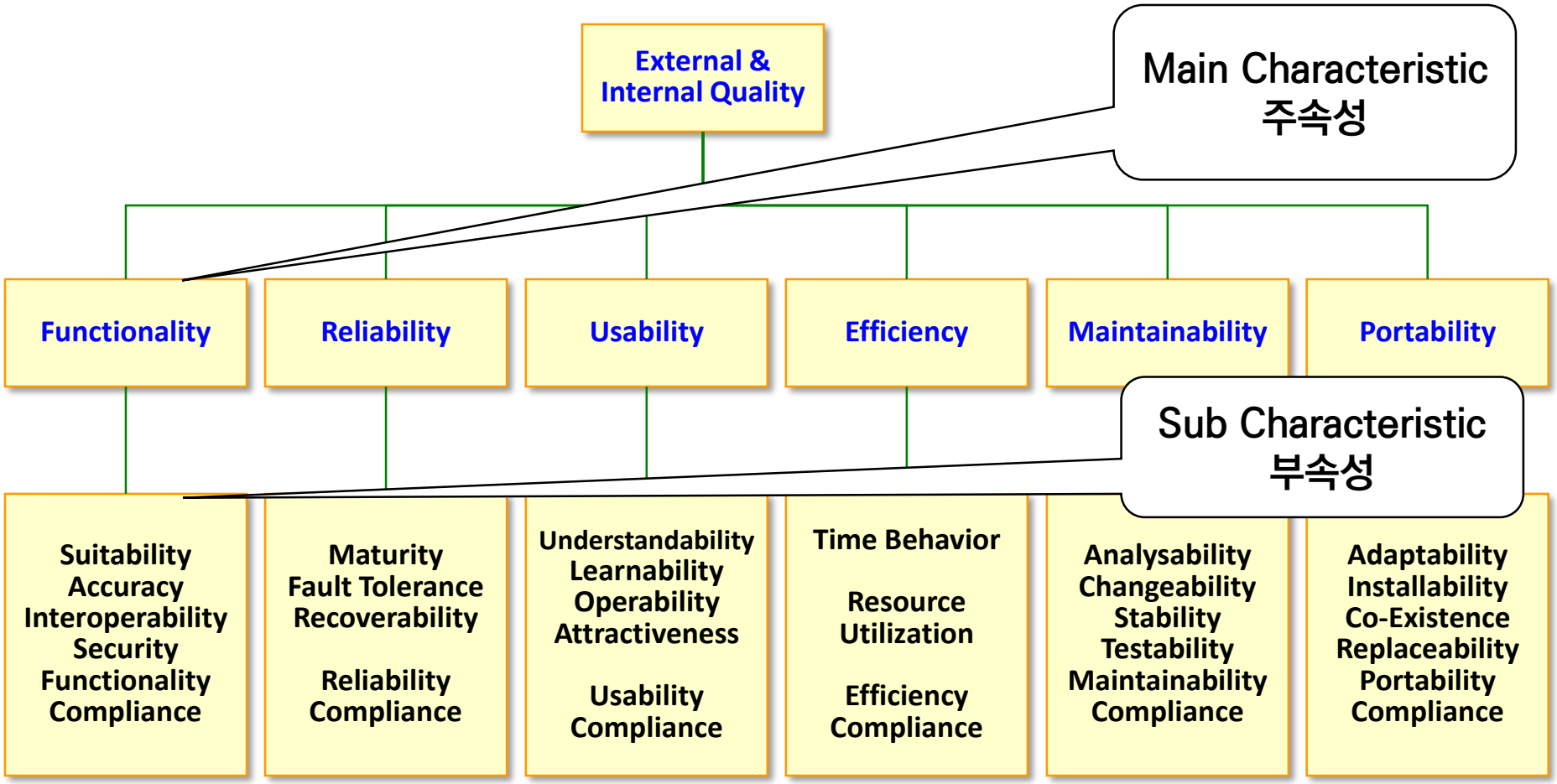
Organization of ISO 9126

- **Information Technology Software Quality Characteristics and Metrics**



Quality Model of ISO 9126

- Quality Model for External and Internal Quality



Six Main Characteristics

- 1. Functionality
  - 주어진 기능이 얼마큼 잘 구현돼 있는지를 측정하는 Metrics
- 2. Reliability
  - 소프트웨어가 주어진 시간, 주어진 환경에서 얼마나 문제없이 잘 돌아가는지, Working 하는지를 나타내는 측정 단위
- 3. Usability
  - Product을 일반 사용자가 얼마나 쉽게 배우고 또는 쉽게 이해하고 잘 사용할 수 있는지를 측정

Six Main Characteristics

- 4. Efficiency
  - 리소스를 잘 사용하는지에 대한 효율성에 대한 속성
- 5. Maintainability
  - Product 자체를 얼마나 쉽게 이해해서 잘 고칠 수 있는 속성
- 6. Portability
  - 하나의 플랫폼에서 개발된 Product이 쉽게 다른 플랫폼으로 이식될 수 있는지에 대한 속성

Sub-characteristics of Functionality

- **Suitability**
  - Capability of the software product to provide an appropriate set of functions for specified tasks and user objectives
- **Accuracy**
  - Capability of the software product to provide the right or agreed results or effects with the needed degree of precision
- **Interoperability**
  - Capability of the software product to interact with one or more specified systems

Sub-characteristics of Functionality

- **Security**
  - Capability of the software product to protect information and data so that unauthorised persons or systems cannot read or modify them and authorised persons or systems are not denied access to them
- **Functionality Compliance**
  - Capability of the software product to adhere to standards, conventions or regulations in laws and similar prescriptions relating to functionality

Sub-characteristics of Reliability

- **Maturity**
  - Capability of software product to avoid failure as a result of faults in the software
- **Fault Tolerance**
  - Capability of software product to maintain a specified level of performance in cases of software faults or infringement of its specified interface

Sub-characteristics of Reliability

- **Recoverability**
  - Capability of software product to re-establish a specified level of level of performance and recover the data directly affected in the case of a failure
- **Reliability Compliance**
  - Capability of software product to adhere to standards, conventions or regulations relating to reliability

Sub-characteristics of Usability

- **Understandability**
  - Capability of the software product to enable the user to understand whether the software is suitable, and how it can be used for particular tasks and conditions of use
- **Learnability**
  - Capability of the software product to enable the user to learn its application
- **Operability**
  - Capability of the software product to enable the user to operate and control it

Sub-characteristics of Usability

- **Attractiveness**
  - Capability of the software product to be attractive to the user
- **Usability Compliance**
  - Capability of the software product to adhere to standards, conventions, style guides or regulations relating to usability

Sub-characteristics of Efficiency

- **Time Behavior**
  - Capability of the software product to provide appropriate response and processing times and throughput rates when performing its function, under stated conditions
- **Resource Utilization**
  - Capability of the software product to use appropriate amounts and types of resources when the software performs its function under stated conditions
- **Efficiency Compliance**
  - Capability of the software product to adhere to standards or conventions relating to efficiency

Sub-characteristics of Maintainability

- **Analyzability**
  - Capability of the software product to be diagnosed for deficiencies or causes of failures in the software, or for the parts to be modified to be identified
- **Changeability**
  - Capability of the software product to enable a specified modification to be implemented
- **Stability**
  - Capability of the software product to avoid unexpected effects from modifications of the software

Sub-characteristics of Maintainability

- **Testability**
  - Capability of the software product to **enable modified software to be validated**
- **Maintainability Compliance**
  - Capability of the software product to adhere to **standards or conventions** relating to maintainability

Sub-characteristics of Portability

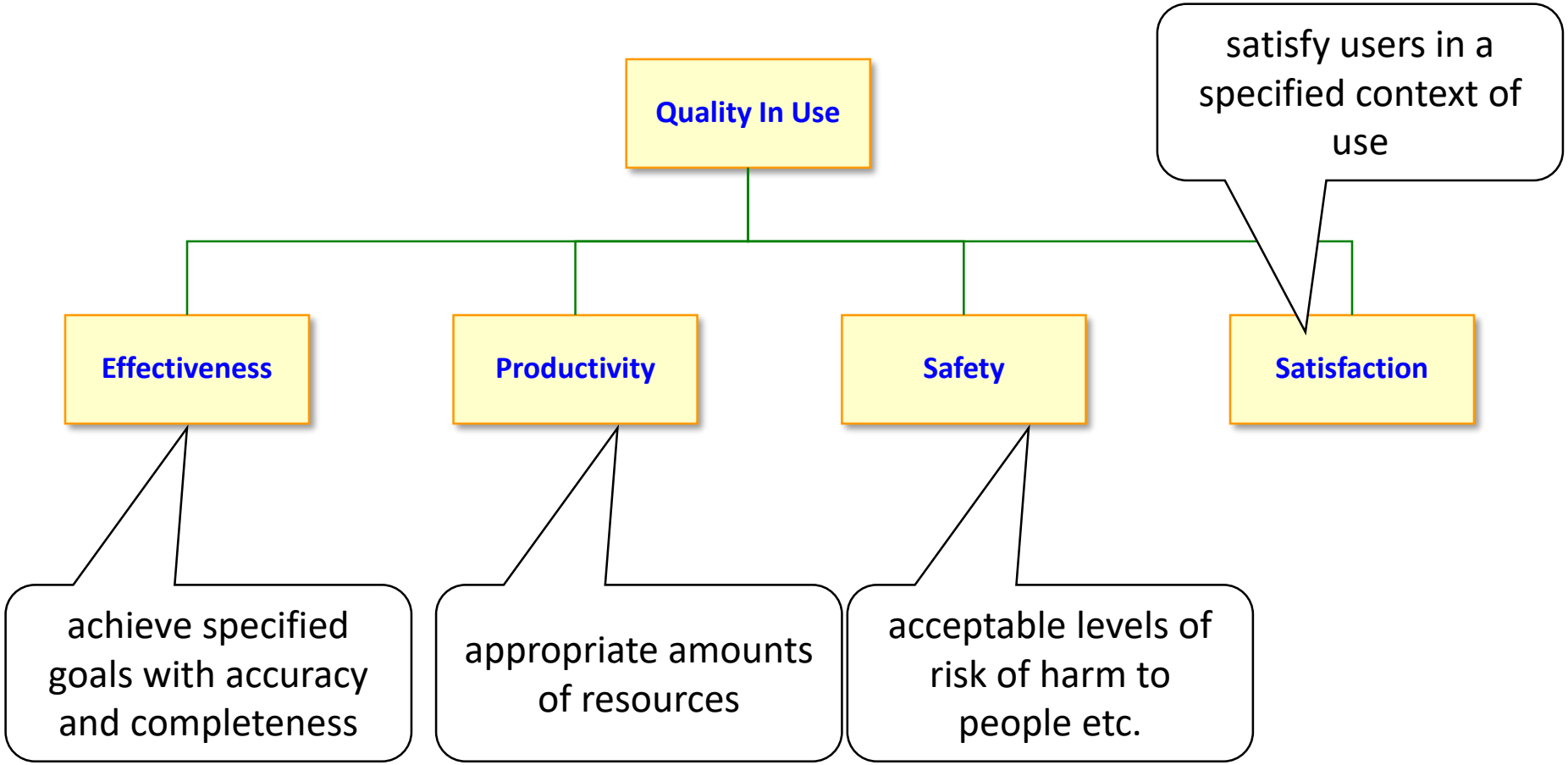
- **Adaptability**
  - Capability of the software product to be adapted for different specified environments without applying actions or means other than those provided for this purpose for the software considered
- **Installability**
  - Capability of the software product to be installed in a specified environment
- **Co-existence**
  - Capability of the software product to co-exist with other independent software in a common environment sharing common resources

Sub-characteristics of Portability

- **Replaceability**
  - Capability of the software product to be used in place of another specified software product for the same purpose in the same environment
- **Portable Compliance**
  - Capability of the software product to adhere to standards or conventions relating to portability

Quality In Use

- **Quality Model for Quality in Use**
  - User's view of the quality of an environment containing software



Four Characteristics

- **1. Effectiveness**
  - Capability of the software product to enable users to achieve specified goals with accuracy and completeness in a specified context of use
- **2. Productivity**
  - Capability of the software product to enable users to expend appropriate amounts of resources in relation to the effectiveness achieved in a specified context of use

Four Characteristics

- **3. Safety**
  - Capability of the software product to achieve acceptable levels of risk of harm to people, business, software, property or the environment in a specified context of use
- **4. Satisfaction**
  - capability of the software product to satisfy users in a specified context of use

Summary Quality Model Framework

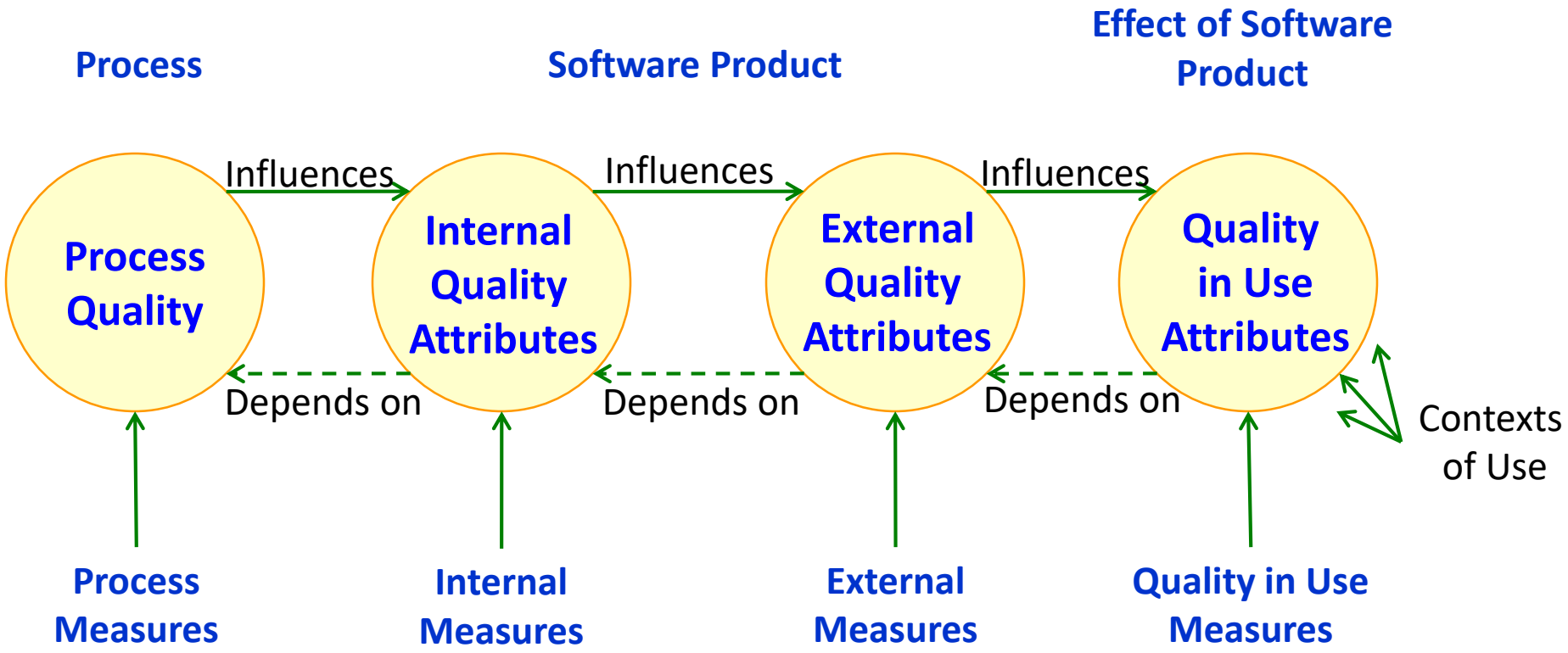
- **Process Quality**
  - Quality of **Life-cycle Process**
  - Process quality contributes to improving product quality.
- **Product Quality**
  - Can be evaluated by measuring **internal attributes or measuring external attributes.**
  - Internal quality
    - is evaluated by the static measure of intermediate products.
    - View at Technical Level
  - External quality
    - is evaluated by measuring the behavior of the code when executed.
    - View of User/Management
  - Product quality contributes to improving quality in use.

Summary Quality Model Framework

- **Quality In Use**
  - **User's view of the quality** of an environment containing software, and is measured from the results of using the software in the environment.
    - Rather than properties of the software itself.
  - User's environment may be different from development environment.

Summary Quality Model Framework

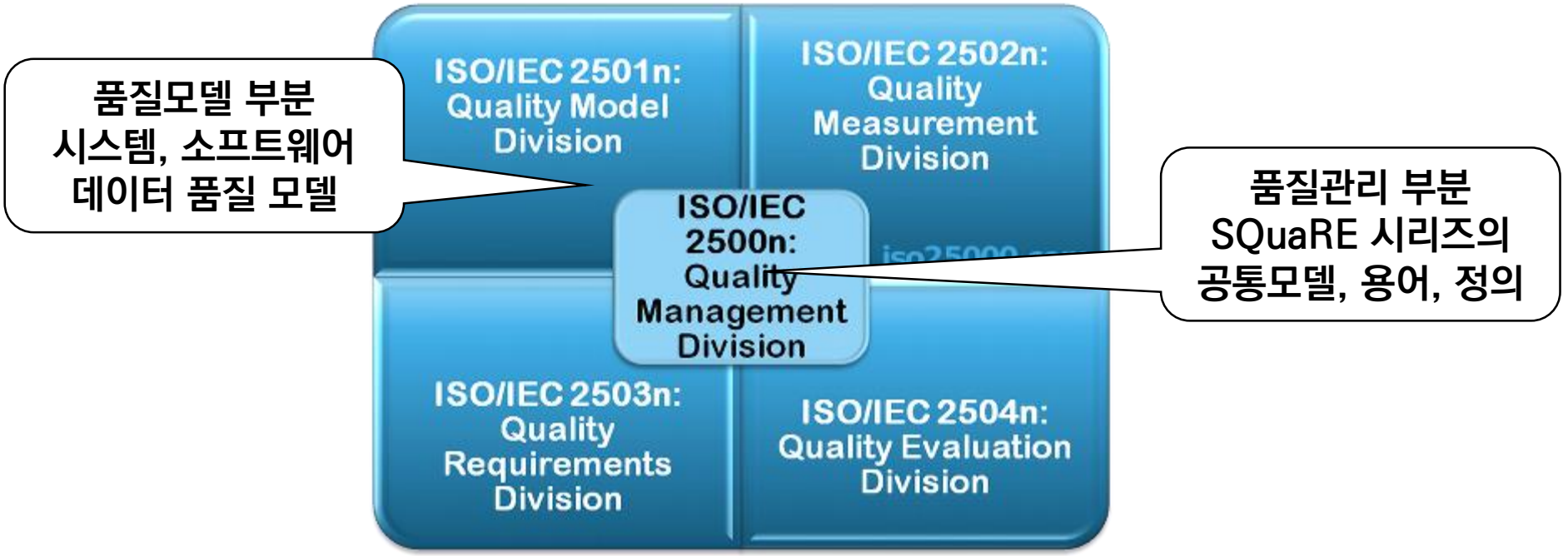
- **Quality in the Lifecycle**



ISO 25000 Series

Organization of ISO 25000

- SQuaRE(System and Software Quality Requirements and Evaluation, 시스템 및 소프트웨어 품질 요구사항 및 평가)
- 소프트웨어 제품 평가를 위한 품질모델을 정의하는 ISO/IEC 9126과 소프트웨어 제품 평가 프로세스를 정의하는 ISO/IEC 14598에서 유래
- 총 5개 부문으로 구성



https://iso25000.com/index.php/en/iso-25000-standards

Quality in Use- ISO/IEC 25010

- 사용품질지표 (Quality in Use)
  - 효과성(Effectiveness):
    - 사용자가 목표를 달성하는 정확성과 완전성
    - 시스템을 이용하여 원하는 작업을 얼마나 정확하고 완벽하게 수행할 수 있는지에 대한 정도.
  - 효율성(Efficiency):
    - 사용자가 원하는 정확성으로 목표를 완전히 달성하기 위해 투입하는 자원의 양
    - 시스템을 사용하는데 드는 시간, 노력, 인지적 부담 등을 최소화하여 사용자가 효율적으로 작업을 완료할 수 있어야 함.
  - 만족도(Satisfaction):
    - 사용성, 신뢰성, 즐거움, 편안함 등 사용자가 시스템을 사용하면서 느끼는 만족감.
    - 시스템이 사용하기 쉽고, 신뢰할 수 있으며, 사용자에게 즐거움과 편안함을 제공해야 사용자가 지속적으로 사용할 의지가 생김.
  - 위험 방지(Freedom from Risk):
    - 시스템 사용으로 인해 발생할 수 있는 경제적, 건강, 안전, 환경적 위험을 완화하는 정도.
    - 시스템 설계 시 안전에 대한 신중한 고려를 통해 사용자의 안전을 최대한 보장해야 함.
  - 상황 적응성(Context Coverage):
    - 시스템이 다양한 상황에 얼마나 완벽하게 적용되고 유연하게 사용될 수 있는지를 나타냄.
    - 예를 들어, 예상치 못한 상황에서도 시스템이 안정적으로 작동해야 하고, 사용자의 작업 환경이나 요구사항에 따라 기능을 조정할 수 있어야 함.
  - 이러한 5가지 특징을 고려하여 시스템을 개발하고 평가함으로써 사용자가 만족스럽게 사용할 수 있는 우수한 사용성을 갖춘 소프트웨어를 만들 수 있습니다.

Product Quality - ISO/IEC 25010

- 기능적 적합성(Functional Suitability):
  - 제품이 사용자의 요구사항을 완벽하게, 정확하게, 그리고 적절하게 충족하는 정도
  - 즉, 제품이 해당 기능을 제대로 수행할 수 있어야 함.
- 성능 효율성(Performance Efficiency):
  - 제품의 속도, 자원 사용량, 처리 능력
  - 제품은 빠른 속도로 작동하고, 최소한의 자원을 사용하며, 높은 처리 능력을 가져야 사용자가 효과적으로 업무를 수행할 수 있음.
- 호환성(Compatibility):
  - 다른 제품과 함께 사용할 수 있는지, 서로 정보를 주고받을 수 있는지를 나타냄
  - 제품은 다른 시스템과 호환되어야 사용 환경의 유연성을 높이고 사용자의 편의를 도울 수 있음.
- 사용성(Usability):
  - 사용자가 제품을 얼마나 쉽게 배우고, 사용하고, 오류 없이 작동할 수 있는지를 나타냄
  - 직관적이고 명확한 디자인, 친절한 사용자 인터페이스, 오류 방지 메커니즘 등을 통해 사용성을 높여야 함.
- 신뢰성(Reliability):
  - 제품이 일관되게 정확한 결과를 제공하고, 예상대로 작동하는 정도
  - 사용자가 언제든지 제품을 신뢰할 수 있어야 함

Product Quality - ISO/IEC 25010

- 보안성(Security):
  - 제품이 데이터의 기밀성(Confidentiality), 무결성(Integrity), 진본성(Authenticity)을 보장하는 정도
  - 중요 데이터를 안전하게 보관하고 부정적인 접근을 막아야 함.
- 유지보수성(Maintainability):
  - 제품을 수정, 개선, 테스트하기 쉽고 효율적으로 수행할 수 있는 정도
  - 제품 개발 이후에도 쉽게 유지보수가 가능해야 함.
- 이식성(Portability):
  - 제품을 다른 환경에서도 쉽게 사용할 수 있는 정도
  - 하드웨어, 소프트웨어 플랫폼 등의 변화에 따라 제품을 쉽게 이식하여 사용할 수 있어야 함

Software Review

Software Reviews

- Reviews
  - 소프트웨어 개발 과정에서 핵심적인 품질 관리 활동 중 하나
- What are they?
  - 기술 전문가 회의(A meeting conducted by technical people)
    - 리뷰는 소프트웨어 개발, 설계, 문서 등 기술적인 업무 산출물을 검토하기 위해 기술 전문가들이 모여 진행하는 회의
  - 업무 산출물 기술 평가(A technical assessment of a work product):
    - 개발 과정에서 생성된 업무 산출물 (예: 코드, 설계 문서, 테스트 케이스) 을 기술적인 관점에서 평가하는 활동.
    - 이를 통해 결함, 개선 가능성, 규칙 위반 사항 등을 식별하고 수정.
  - 소프트웨어 품질 보증 메커니즘(A software quality assurance mechanism):
    - 리뷰는 소프트웨어 개발 과정에서 품질을 지속적으로 관리하고 개선하는 데 도움이 되는 메커니즘
    - 결함을 조기에 발견하고 수정함으로써 전체적인 개발 비용과 시간을 절감
  - 훈련의 장(A training ground):
    - 리뷰는 개발자들이 서로의 코드 및 설계 방식을 검토하고 배우는 좋은 기회.
    - 경험이 많은 개발자는 리뷰 과정을 통해 후배 개발자에게 지식과 경험을 전수.
- What they are not!
  - 프로젝트 요약 또는 진행 상황 평가(A project summary or progress assessment):
    - 리뷰는 프로젝트의 전체적인 진행 상황을 평가하는 것이 아님. 특정 업무 산출물에 대한 집중적인 기술 검토에 초점.
  - 정보 전달만을 위한 회의(A meeting intended solely to impart information)
    - 리뷰는 단순히 정보를 전달하는 것이 목적이 아님. 검토자들은 업무 산출물을 주의 깊게 검토하고 의견을 공유하며, 개선을 위한 토론.
  - 정치적 또는 개인적인 비난의 장소(A mechanism for political or personal reprisal):
    - 리뷰는 개발자를 비난하거나 공격하는 장소가 되어서는 안됨. 리뷰는 건강하고 객관적인 토론을 통해 업무 산출물의 품질을 향상시키는 데 목적

Software Reviews

- **Concept**
  - Reviews are a '**Filter**' for software engineering process.
  - Applied at various points during software development and serve to **uncover errors**.
  - To **purify the work products** that occur at analysis, design and coding.
- **Types of review**
  - Informal Review
  - Formal Technical Review (FTR)
  - Walkthrough

Difference between Inspection and Walkthrough

- **Walkthrough**
  - a method of conducting **informal group/individual review** but with purpose
  - author **describes and explain work product** in a informal meeting to his peers or supervisor to get feedback
  - **validity** of the proposed solution for work product is checked
  - **cheaper to make changes** when design is on the paper rather than at time of conversion.
- **Inspection**
  - **formal**, rigorous, in depth group review designed to identify problems as close to their point of origin as possible
  - improve **reliability, availability, and maintainability** of software product
  - Inspections can be **combined with** structured, systematic **testing** to provide a powerful tool for creating defect-free programs
  - consists of **three to eight members** who plays roles of moderator, author, reader, recorder and inspector.

Cost Impact of Software Defects

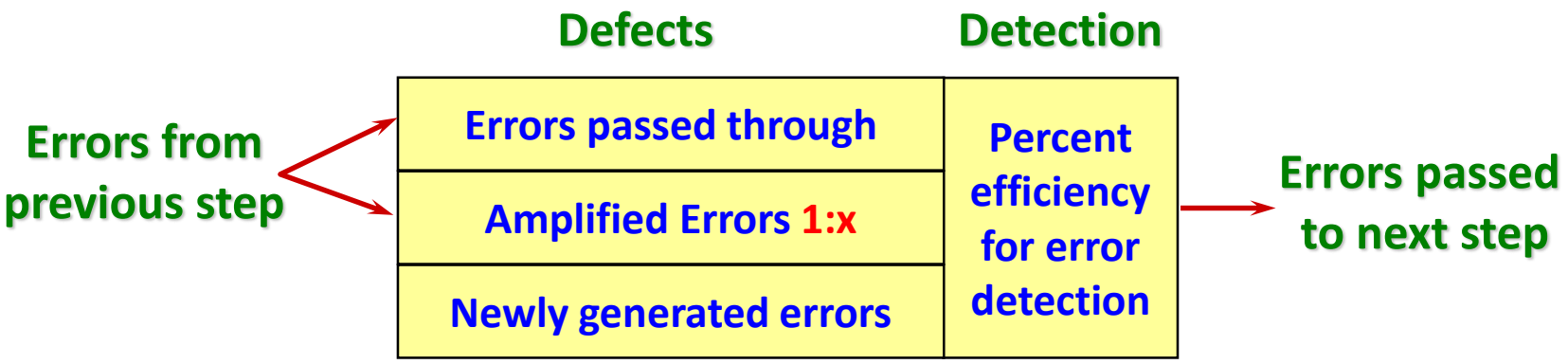
- **Terms**
  - Error
    - Quality problem that is discovered by engineers before release
  - Defect (Fault)
    - Quality problem that is discovered after release
- **Objective of Review**
  - To find errors during the process that become defects after release of the software.
  - Benefit: Early Discovery of Errors
- **Industry Studies**
  - Design activities introduce **50-65% of all errors**.
  - Formal review techniques have been shown to **be up to 75% effective** in uncovering design errors.

Defect Amplification and Removal

- 결함 증폭 및 제거 (Defect Amplification and Removal)
- 결함 증폭 (Defect Amplification):
  - 초기 단계에서 발견되지 않은 결함이 개발 과정을 거쳐 더욱 심각해지는 현상을 의미.
  - 예를 들어, 요구사항 모델링 단계에서 발견되지 않은 결함은 설계 단계에서 여러 오류로 증폭되고, 코딩 단계에서는 훨씬 더 많은 오류로 발전할 수 있음.
  - 이와 같은 결함 증폭은 개발 과정 전체에 걸쳐 발생할 수 있으며, 궁극적으로 소프트웨어 품질 저하, 개발 지연, 비용 증가 등을 초래.
- 결함 전파 (Defect Propagation):
  - 발견되지 않은 결함이 미래의 개발 활동이나 제품 동작에 미치는 영향.
  - 미 발견 결함은 개발 과정의 다음 단계로 전파되어 예상치 못한 문제를 야기.
  - 이는 테스트 단계에서도 발견되지 않을 수 있으며, 결함이 최종 사용자에게 노출되어 심각한 문제를 일으킬 수 있음.
- 기술 부채 (Technical Debt):
  - 초기 단계에서 결함을 발견하고 수정하지 않거나, 소프트웨어 변경 사항에 대한 문서를 업데이트하지 않음으로써 발생하는 비용(pressman)
  - 현 시점에서 더 오래 소요될 수 있는 더 나은 접근방식을 사용하는 대신 쉬운(제한된) 솔루션을 채택함으로써 발생되는 추가적인 재 작업의 비용 (위키)
  - 기술 부채는 곧바로 문제가 발생하지 않을 수도 있지만, 장기적으로는 개발 속도 저하, 유지보수 비용 증가, 코드 품질 저하 등을 초래하며, 결과적으로 프로젝트 전체 비용을 증가.

Defect Amplification Model

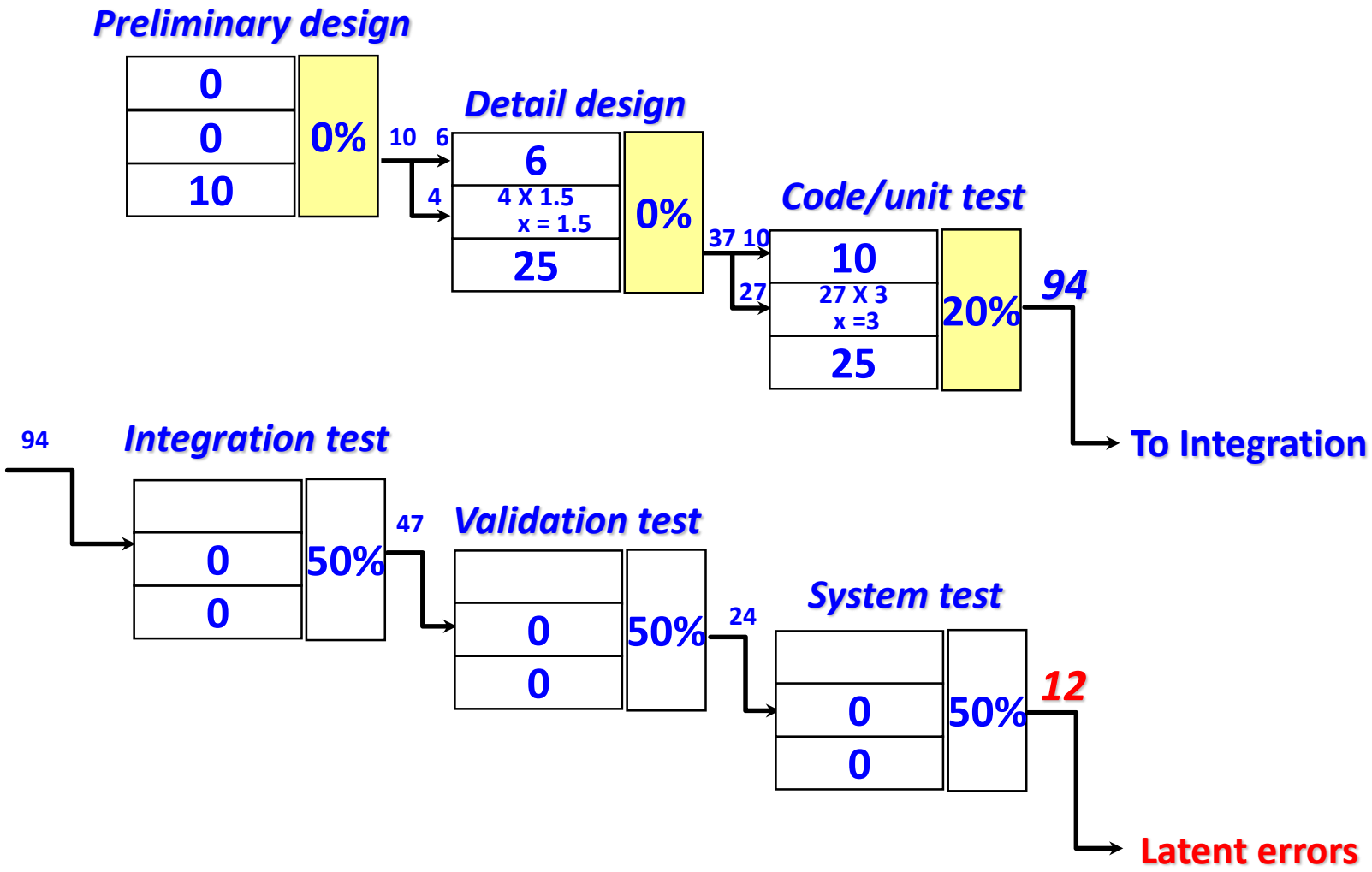
- **Errors may be inadvertently generated.**
- **Illustrates the generation and detection of errors during each of developing software.**
  - Analysis, Preliminary Design, Detailed Design, Coding
- **Review may fail to uncover newly generated errors and errors from previous steps, resulting in some errors that are passed through.**



A Development Step

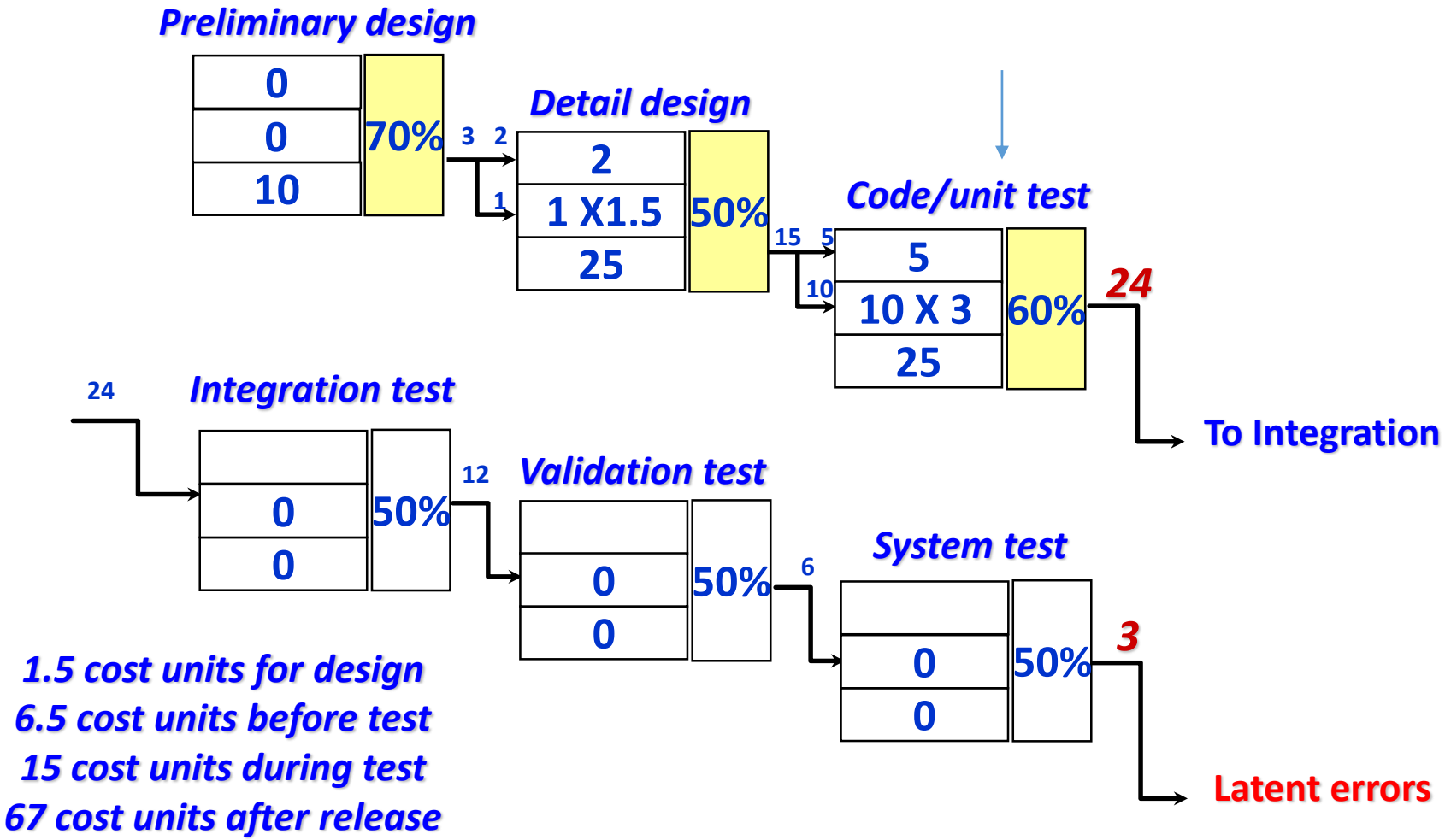
Defect Amplification Model

- No Reviews applied



Defect Amplification Model

- Reviews applied



Review Metrics

- ***Preparation effort, E_p*** — the effort (in person-hours) required to review a work product prior to the actual review meeting.
- ***Assessment effort, E_a*** — the effort (in person-hours) that is expending during the actual review.
- ***Rework effort, E_r*** — the effort (in person-hours) that is dedicated to the correction of those errors uncovered during the review.
- ***Work product size, WPS*** — a measure of the size of the work product that has been reviewed (for example: the number of UML models, or the number of document pages, or the number of lines of code).
- ***Minor errors found, Err_{minor}*** — the number of errors found that can be categorized as minor (requiring less than some pre-specified effort to correct).
- ***Major errors found, Err_{major}*** — the number of errors found that can be categorized as major (requiring more than some pre-specified effort to correct).

Review Metrics

- ***Total errors found, Err_{tot}. Represents the sum of the errors found:***

$$Err_{tot} = Err_{minor} + Err_{major}$$

- ***Error density. Represents the errors found per unit of work product reviewed:***

$$\text{Error density} = Err_{tot} \div \text{WPS}$$

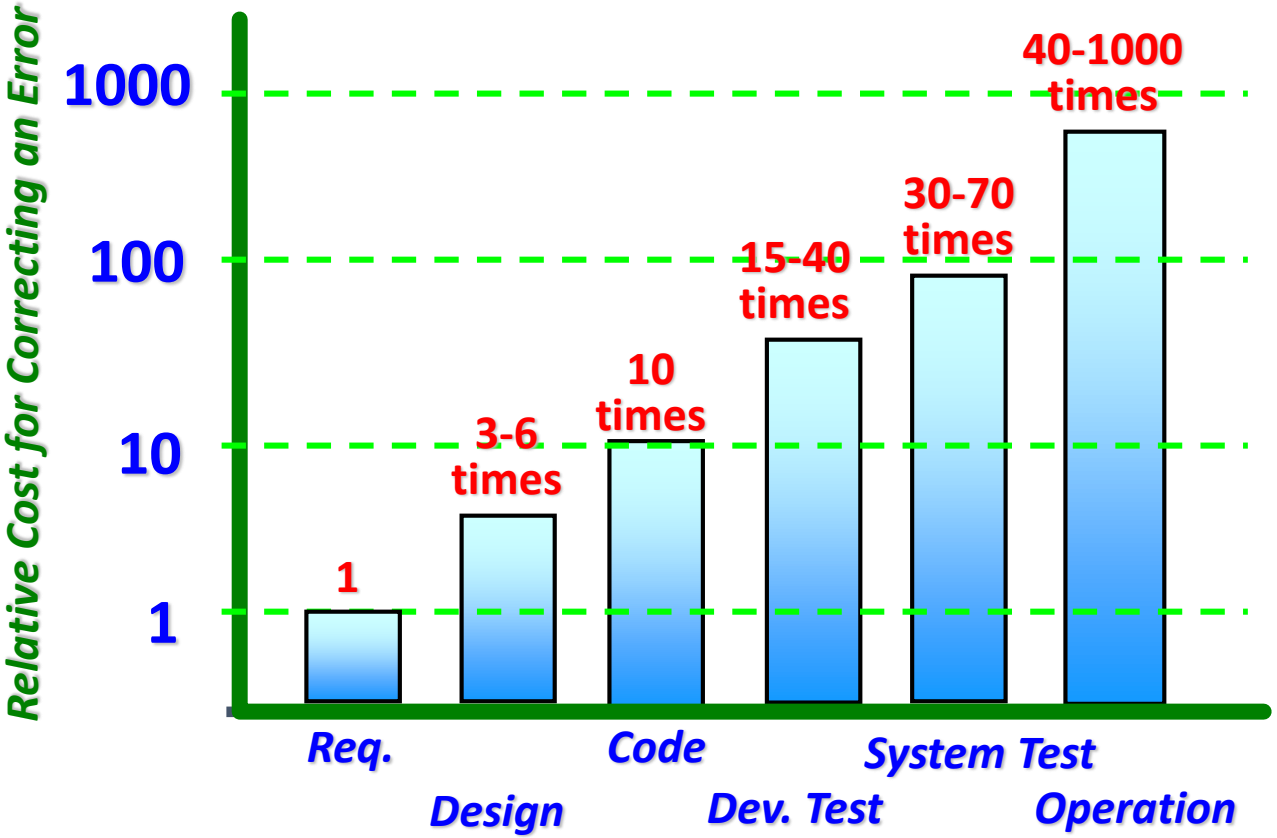
Metrics Example

- **Estimation Defect Density**
  - Assume Requirement model contains 18 UML Diagrams as part of 32 overall pages, the review uncovers 18 minor errors and 4 major errors
    - Err_{tot} = 22, Error density = 1.2 errors/UML (or 0.68 errors/page)
    - The average defect density for a requirements model is 0.68 errors per page
  - A new requirement model is 40 pages long.
    - A rough estimate suggests that your software team will find about 27 errors during the review of the document.
      - 총 결함 수 = 결함 밀도 * 페이지 수 총 결함 수 = 0.68 (errors/page) * 40 pages ≈ 27.2 errors (round up to 27 for estimation)
    - If you find only 9 errors, you've done an extremely good job in developing the requirements model or your review approach was not thorough enough.

Metrics Example

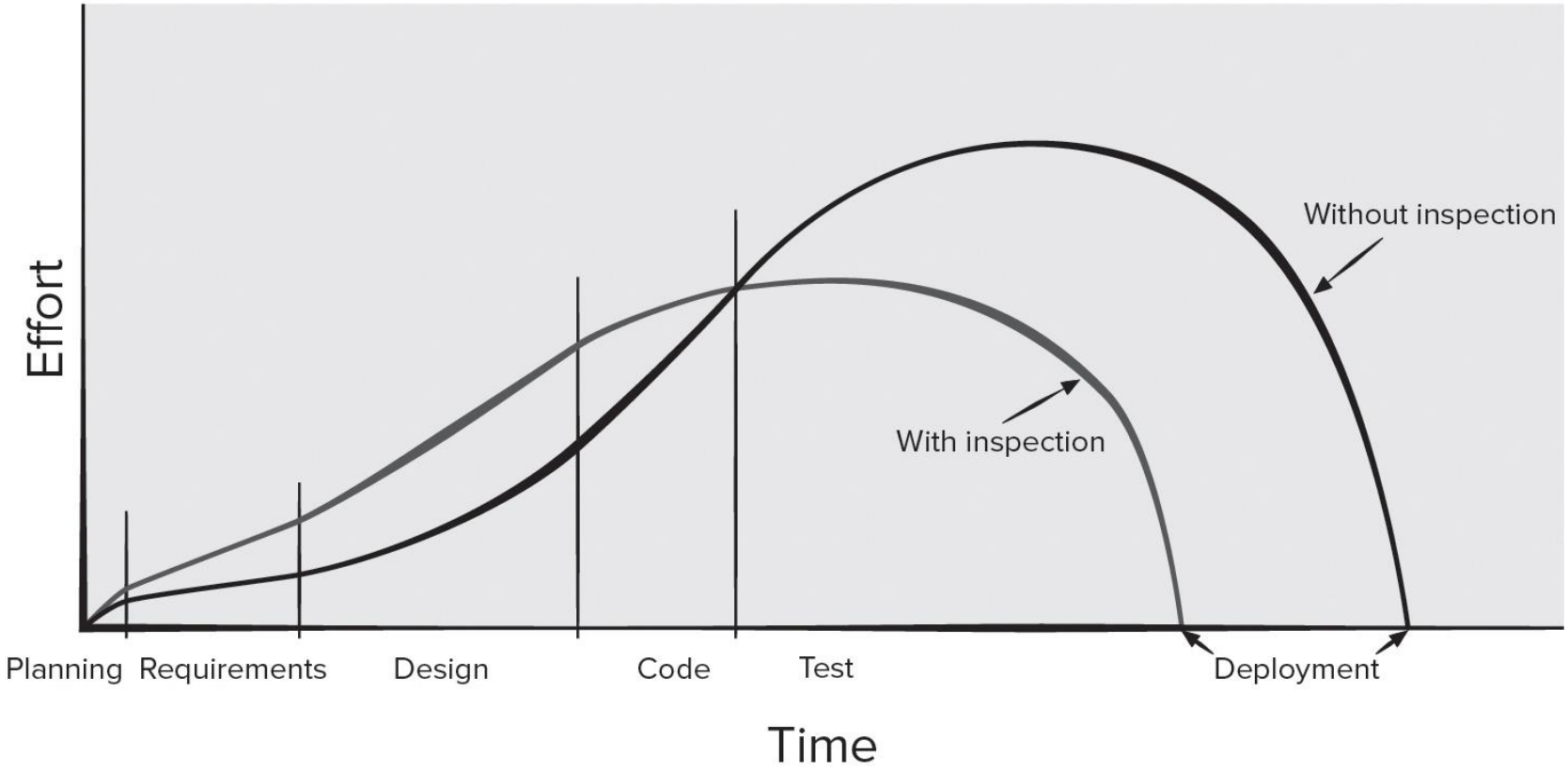
- **Cost effectiveness**
  - 이전 페이지 총 결함 수 = 0.68 (errors/page) * 40 = 27
  - Assume 사소한 요구사항 모델 오류(minor requirement model error)를 수정하는 데 필요한 effort은 약 4 person/hour.
  - Assume 심각한 요구사항 모델 오류(major requirement model error)를 수정하는 데 필요한 effort은 약 18 person/hour.
  - 사소한 오류는 심각한 오류보다 약 6배 더 빈번하게 발생.
    - Minor error = 전체 에러 중 약 85%(약 23개, 23.14), Major error = 약 15%(약 3개, 3,85)
  - Review 시에 에러 fix하는 effort는 about 6 person/hours
    - Correct cost = (minor error fix cost + major error fix cost)/total error count = about 6 person/hours
  - Requirements related errors uncovered during testing require an average of 45 person-hours to find and correct. Using the averages noted, we get:
  - Effort saved per error = $E_{testing} - E_{reviews}$
    $= 45 - 6 = 39$ person-hours/error
  - 22 errors were found during the review of the requirements model
  - a saving of about 858 person-hours of testing effort would be achieved. And that's just for requirements-related errors.

Cost to Correct an Error



Effort Expended With and Without Reviews

- **Effort-time graph**



Source: Fagan, Michael E., "Advances in Software Inspections," IEEE Transactions on Software Engineering, vol. SE-12, no. 7, July 1986, 744–751.

Effort Expended With and Without Reviews

- A graph plots expended effort with and without inspection.
- The graph is plots time on the x-axis and effort on the y-axis.
- The time axis has intervals for planning, requirements, design, code, test and deployment.
- The curve for with inspection shows and increase in effort expenses for planning, requirements, design and code. The curve begins to fall from test to deployment displaying a decrease in effort.
- Without inspection curve shows less effort is expensed for the planning, requirements, design and code phases, the curve is below the with inspection curve. Between test and deployment, the without inspection curve displays greater expensed effort when compared to the with inspection curve. The curve is above the with inspection curve.

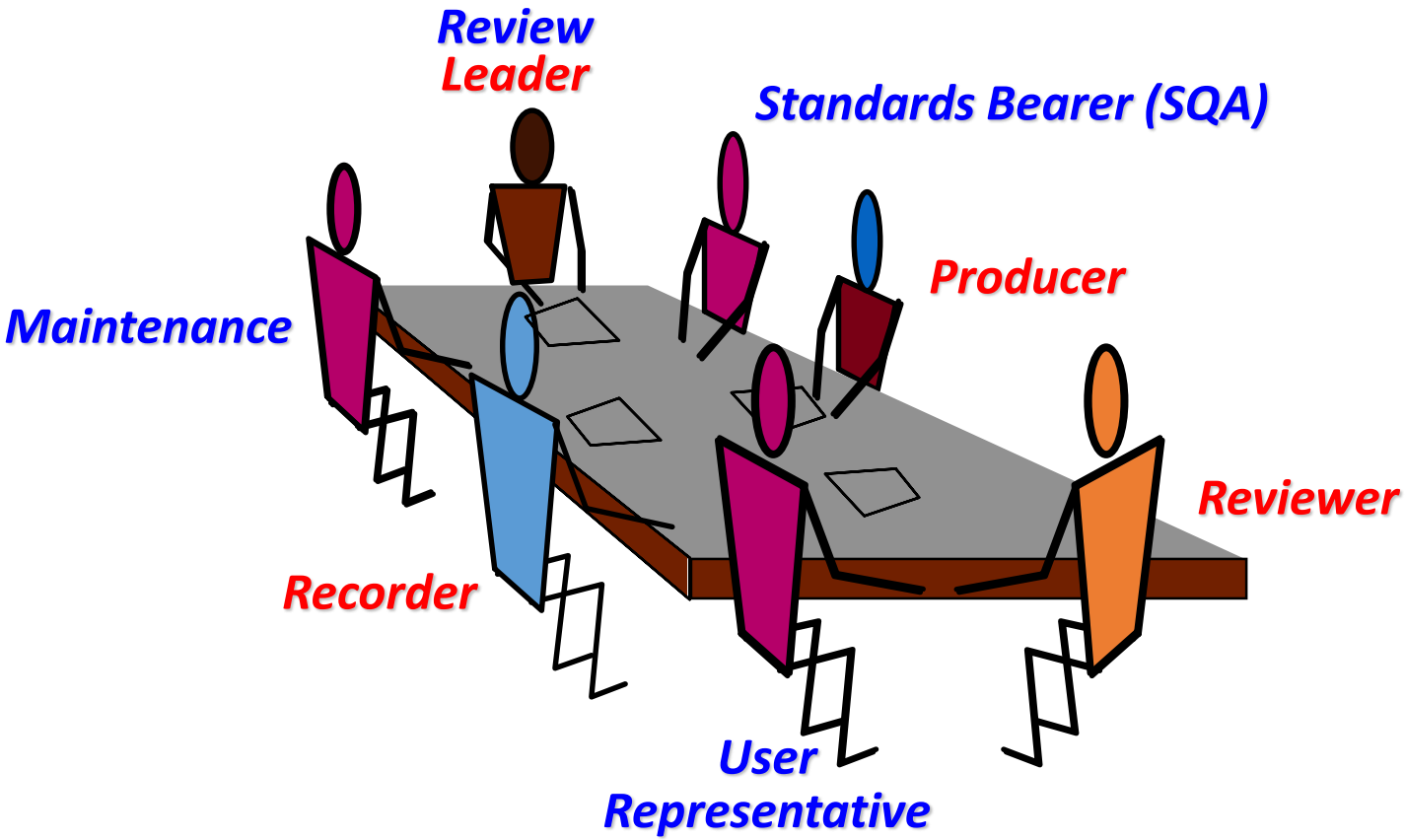
Formal Technical Review (FTR)

- **FTR is a SQA activity performed by software engineers.**
- **Objectives**
  - To uncover errors in function, logic or implementation
  - To verify that the software under review meets its requirements
  - To ensure that the software has been represented according to predefined standards
  - To achieve software that is developed in a uniform manner
  - To make projects more manageable

FTR Meeting

- **Each FTR is conducted as a meeting.**
- **Constraints for Review Meetings**
  - Between three and five people should be involved.
  - Advance preparation should occur. (No more 2 hours)
  - Duration of the meeting should be less than 2 hours.
- **Hence, an FTR focuses on a specific part of the overall software.**
  - A higher likelihood of uncovering errors
- **Focus of FTR is on a work product.**

Players of FTR Meeting



Procedure for FTR Meeting

- **The producer informs the project leader that work product is complete and that a review is required.**
- **The project leader contacts an review leader.**
- **The review leader does a brief review and distributes materials to 2-3 reviewers.**
- **The Reviewers review the work and make note.**
  - Concurrently, the review leader makes agenda for review meeting.

Procedure for FTR Meeting

- **Begins with agenda and brief introduction.**
- **Producer proceeds to "walk through" the work product.**
- **Reviewers may raise issues based on their advance preparation.**
- **When valid problems or errors are discovered, the recorder notes each.**
- **At the end of the meeting, all attendees must decide whether to:**
  - Accept the product without any modification.
  - Reject the project due to serious error (Once corrected, another app need to be reviewed)
  - Accept the product provisional (*minor errors* are encountered and should be corrected, but no additional review will be required)

Guidelines for Review Meeting

- **Review the product, not the producer.**
- **Take written notes(record purpose)**
- **Limit the number of participants and insist upon advance preparation**
- **Develop a checklist for each product that is likely to be reviewed.**
- **Allocate resources and time schedule for FTRs in order to maintain time schedule.**
- **Conduct meaningful training for all reviewers in order to make reviews effective.**
- **Reviews earlier reviews which serve as the base for the current review being conducted.**
- **Set an agenda and maintain it.**
- **Separate the problem areas, but do not attempt to solve every problem notes.**
- **Limit debate and rebuttal.**

Postmortem Evaluations

- 사후 평가(PME)는 소프트웨어 개발 프로젝트에서 사용된 공정 및 실무에 대해 무엇이 잘되었고, 무엇이 잘못되었는지를 파악하기 위한 메커니즘
- 참여자:
  - PME에는 소프트웨어 개발 팀 구성원과 이해관계자들이 함께 참여
- 평가 항목:
  - PME에서는 전체 소프트웨어 프로젝트를 검토
- 주요 평가 항목
  - 성공 사례 (Excellences): 프로젝트 진행 과정에서 달성한 성과와 긍정적인 경험
  - 문제점 (Challenges): 프로젝트 진행 과정에서 발생한 문제점과 부정적인 경험
- 목적:
  - 성공 사례와 문제점으로부터 얻은 경험을 추출하여 앞으로의 프로젝트 개선에 활용.
  - 개발 프로세스 및 실무 개선 방안을 제안

Software Quality Metrics

Software Metrics

- **Software metrics can be classified into three categories**
- **Product metrics**
  - Describes the characteristics of the product such as size, complexity, design features, performance, and quality level
- **Process metrics**
  - These characteristics can be used to improve the development and maintenance activities of the software.
- **Project metrics**
  - These metrics describe the project characteristics and execution.
  - Examples include the number of software developers, the staffing pattern over the life cycle of the software, cost, schedule, and productivity

Software Quality Metrics

- Software Quality Metrics
  - 일반적인 소프트웨어 측정 지표의 하위 집합으로, 제품, 프로세스, 프로젝트의 품질 특성에 초점을 둔 지표
  - 입니다. 이 지표들은 일반적인 소프트웨어 측정 지표 분류 중 프로젝트 측정 지표보다는 제품 측정 지표 및 프로세스 측정 지표와 더욱 밀접하게 관련이 있음.
- Classified into three categories
  - 제품 품질 측정 지표 (Product Quality Metrics)
    - 제품 품질 측정 지표는 개발된 소프트웨어 자체의 품질을 평가하는 지표.
    - 예시:
      - 결함 수 (Number of Defects), 결함 밀도 (Defect Density), 테스트 커버리지 (Test Coverage), 기능적 적합성 (Functional Suitability), 사용성 (Usability),성능 (Performance)
  - 개발 과정 내 품질 측정 지표 (In-process Quality Metrics)
    - 개발 과정 내 품질 측정 지표는 소프트웨어 개발 과정 중 품질을 개선하는 데 사용되는 지표
    - 개발 진행 중 발생하는 문제를 식별하고 개선하는 데 도움을 줌.
    - 예시:
      - 코드 리뷰 빈도 (Code Review Frequency),정적 코드 분석 결과 (Static Code Analysis Results),단위 테스트 통과율 (Unit Test Pass Rate),결함 수정 시간 (Defect Fix Time)
  - 유지보수 품질 측정 지표 (Maintenance Quality Metrics)
    - 유지보수 품질 측정 지표는 소프트웨어 유지보수성을 평가하는 지표
    - 개발된 소프트웨어를 얼마나 쉽게 수정, 개선 및 유지보수할 수 있는지를 평가하는 데 사용.
    - 예시:
      - 코드 복잡성 (Code Complexity),모듈성 (Modularity),문서화 수준 (Documentation Level),유지보수 비용 (Maintenance Cost)
  - 이 세 가지 측정 지표를 활용하여 소프트웨어 개발 전 과정을 통하여 품질을 지속적으로 관리하고 개선할 수 있음

- **Example of Product Quality Metrics**
  - **결함 밀도 (Defect Density)**
    - Defect Density is the number of defects confirmed in software(module) during a specific period of operation or development divided by the size of the software(module)
    - Defect Density = Defect count/size of the release
    - Example)
    - you have 3 modules integrated into your software product. Each module has the following number of bugs discovered
      - Module 1 = 10 bugs, 1000 LOC (Line of Code)
      - Module 2 = 20 bugs, 1500 LOC
      - Module 3 = 10 bugs , 500 LOC
      - **Total bugs** = 10+20+10 =40, **Total Line of Code** = 1000+1500+500 = 3000
      - **Defect Density** = 40/3000 = 0.013333 defects/loc = 13.333 defects/KLoc

Software Quality Metrics

- **Example of Product Quality Metrics (cont.)**
  - 결함 밀도는 낮을수록 일반적으로 품질이 더 좋다고 언급.
  - 하지만 결함 밀도의 "좋은" 기준값은 절대적인 것이 아니며, 다음과 같은 요소에 따라 달라질 수 있음
    - 프로젝트 복잡성: 더 복잡한 프로젝트는 일반적으로 결함 밀도가 높을 수 있음
    - 개발 방법론: 엄격한 개발 방법론을 사용하는 프로젝트는 결함 밀도가 낮을 수 있음
    - 테스트 수준: 광범위한 테스트를 수행하는 프로젝트는 결함 밀도가 낮을 수 있음
    - 참고) 연구 결과에 따르면 일반적으로 천 코드 라인당 1개 이하의 결함 밀도는 좋은 프로젝트 품질의 지표로 간주. 이는 절대적인 기준 값이 아니며, 프로젝트의 특수성을 고려하여 적절한 해석이 필요
  - **Advantages of defect density**
    - 테스트 효과 측정
    - 구성요소/소프트웨어 모듈 결함 구별
    - 수정 또는 개선 영역 식별
    - 고위험 구성요소 식별
    - 나머지 결함 추정
    - 테스트의 충분성 등

- **Example of Product Quality Metrics (cont.)**
  - **Mean Time to Failure(MTTF)**
    - It is the time between failures. This metric is mostly used with safety critical systems such as the airline traffic control systems, avionics, and weapons.

- Mean Time to Repair (평균 수리시간): 평균적으로 걸리는 수리시간
- Mean Time to Failure (평균 고장시간): (평균) 첫사용부터 고장시간까지
- Mean Time Between Failure (평균 고장 시간 간격)
- MTBF = MTTR+MTTF

- **Example of Product Quality Metrics (cont.)**
  - **Software Availability**
    - The probability that a program is operating according to requirements at a given point in time and is defined as .
    - **Availability =** $\frac{MTTF}{(MTTF+MTTR)} \times 100\%$

Software Quality Assurance

Software Quality Management

- Phil Crosby 품질 관리 인식에 대한 지적
  - "품질 관리 문제는 사람들이 그것에 대해 모르는 것이 아니라, 자신들이 알고 있다고 생각하는 것."
- 품질 관리에 대한 다음과 같은 일반적인 오해를 비판.
  - 모두가 품질 관리를 지지 (하지만 특정 조건 하에서만):
    - 많은 사람들은 품질 관리의 중요성을 인정하지만 실제 행동으로 옮기는 데에는 소극적인 태도.
    - 예를 들어, 품질 관리를 위한 노력이 프로젝트 진행 속도를 저하시킬까봐 우려하는 경우.
  - 모두가 품질 관리를 이해한다고 생각 (하지만 설명하기 어려워 한다):
    - 대부분의 사람들은 품질 관리의 개념 자체는 이해한다고 생각하지만 구체적인 방법이나 실행 방식에 대한 이해는 부족.
    - 이러한 부족한 이해는 실제 품질 개선 활동으로 이어지지 않을 수 있음.
  - 품질 관리 실행은 단순히 본능적인 경향을 따르는 것 (어떻게든 잘 될 것이다):
    - 일부 사람들은 품질 관리가 특별한 노력 없이도 자연스럽게 이루어질 것이라고 생각.
    - 하지만 품질 관리를 위해서는 명확한 목표 설정, 프로세스 개선, 지속적인 노력 필요.
  - 대부분의 사람들은 문제는 다른 사람들이 일으킨다고 생각한다 (만약 다른 사람들이 제대로 하기만 했으면):
    - 문제가 발생했을 때 책임을 스스로 지기보다는 다른 사람이나 상황의 탓을 하는 경향.
    - 품질 관리 개선을 위해서는 문제의 근본 원인을 파악하고, 시스템적인 개선 노력이 필요.
  - 품질은 단순히 지지하거나 이해하는 것만으로는 이루어지지 않으며, 구체적인 방법론과 지속적인 노력을 통해 실천해야 함

Software Quality Assurance (SQA)

- **SQA**
  - The software quality assurance is displayed in a circle at the center with the components displayed around with arrows pointing to it. The components are define process, change control, reviews and testing, methods and tools, audits and compliance, and measurement and reporting

Software Quality Assurance (SQA)

- **SQA**
  - simply **a way to assure quality** in the software.
  - a process which **works parallel to development** of software.
  - **focuses on improving the process of development** of software so that problems can be **prevented** before they become a major issue.
  - a kind of Umbrella activity that is applied throughout the software process.
- **SQA has**
  - A quality management approach ：품질관리 접근방식
  - Formal technical reviews ：공식 기술검토
  - Multi testing strategy ：다중 테스트 전략
  - Effective software engineering technology：효과적인 소프트웨어 엔지니어링 기술
  - Measurement and reporting mechanism：측정 및 보고 메커니즘

Elements of SQA

- **Standards**
- **Review/Audits**
- **Testing**
- **Error/defect collection and analysis**
- **Change management**
- **Education**
- **Vendor management**
- **Security management**
- **Safety**
- **Risk management**

Roles of SQA Group

- **Prepare a SQA plan for a project.**
- **Participates in the development of process description.**
- **Reviews SE activities to verify compliance with the defined software process.**
- **Audits designated software work products to verify compliance with those defined as part of the process.**
- **Ensures that deviations in work and work products are documented and handled.**
- **Records any non-compliance and reports to senior management.**

Major SQA Activities

- **SQA Management Plan**
  - Make a plan for how you will carry out the sqa through out the project. Think about which set of software engineering activities are the best for project. check level of sqa team skills.
- **Set The Check Points**
  - SQA team should set checkpoints. Evaluate the performance of the project on the basis of collected data on different check points.
- **Multi testing Strategy**
  - Do not depend on a single testing approach. When you have a lot of testing approaches available use them.
- **Measure Change Impact**
  - The changes for making the correction of an error sometimes re introduces more errors keep the measure of impact of change on project. Reset the new change to change check the compatibility of this fix with whole project

Statistical Quality Assurance

Statistical Quality Assurance

- **Industry prefers more quantitative approach about quality.**
- **General Steps**
  1. Information about software defects is collected and categorized.
  2. Trace each defect to its underlying cause.
  3. Using the *Pareto* principle, isolate the 20% (the vital few).
     - 80% of defects are traced to 20% of all possible causes.
  4. Once the 'vital few' causes have been identified, correct the problems that have caused the defects.

Statistical Quality Assurance

- **Typical Causes**
  - Incomplete or Erroneous Specification **(IES)**
  - Misinterpretation of Customer Communication **(MCC)**
  - Intentional Deviation from Specifications (IDS)
  - Violation of Programming Standards (VPS)
  - Error in Data Representation **(EDR)**
  - Inconsistent Module Interface (IMI)
  - Error in Design Logic (EDL)
  - Incomplete or Erroneous Testing (IET)
  - Inaccurate or Incomplete Documentation (IID)
  - Error in Programming Language Translation of Design (PLT)
  - Ambiguous or Inconsistent Human-Computer Interface (HCI)
  - Miscellaneous (MIS)

Statistical Quality Assurance

- **Statistical QA Table**
  - **IES, MCC and EDR** are the vital causes that account **for 53% of all errors**.
    - Begin corrective action on the vital few causes.

	Total		Serious		Moderate		Minor	
ERROR	No.	%	No.	%	No.	%	No.	%
IES	205	22%	34	27%	68	18%	103	24%
MCC	156	17%	12	9%	68	18%	76	17%
IDS	48	5%	1	1%	24	6%	23	5%
VPS	25	3%	0	0%	15	4%	10	2%
EDR	130	14%	26	20%	68	18%	36	8%
IMI	58	6%	9	7%	18	5%	31	7%
EDL	45	5%	14	11%	12	3%	19	4%
IET	95	10%	12	9%	35	9%	48	11%
IID	36	4%	2	2%	20	5%	14	3%
PLT	60	6%	15	12%	19	5%	26	6%
HCI	28	3%	3	2%	17	4%	8	2%
MIS	56	6%	0	0%	15	4%	41	9%
Total	942	100%	128	100%	379	100%	435	100%

Statistical Quality Assurance

- **Phase Index (PI)**

$$PI_i = w_s (S_i / E_i) + w_m (M_i / E_i) + w_t (T_i / E_i)$$

  - w_s, w_m, w_t are weighting factors for *Serious*, *Moderate* and *Trivial* Errors.
    - Typically 10, 3, and 1.
  - E_i = Total number of errors uncovered.
  - S_i = The number of *serious* errors.
  - M_i = The number of *moderate* errors.
  - T_i = The number of *minor* errors.

- **Error Index (EI)**
  - An overall indication of improvement in software quality.
  - $EI = \sum(i \times PI_i) / PS$
    $= (1*PI_1 + 2*PI_2 + 3*PI_3 + i*PI_i) / PS$
    - PS = Size of the product
    - I : Index no

END