Unit 7.
Quality Management

Contents

- Software Quality Assurance (SQA)
- Software Reviews
  - Formal Technical Reviews
  - Defect Amplification Model
- Statistical Software Quality Assurance

Software Quality

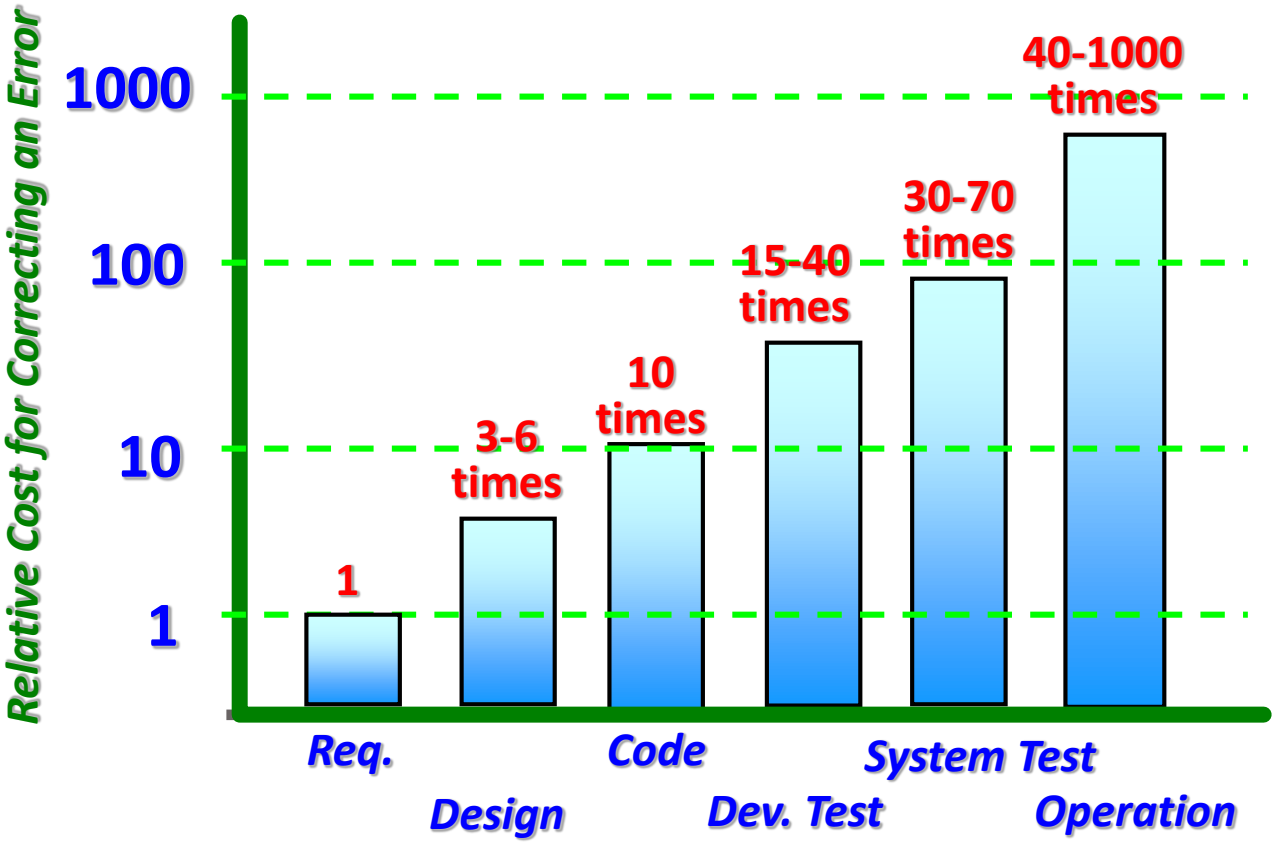
non-function 요구사항
〉앞에서 노트동기화의 경우
명시적 요구사항이지만
그 노트동기화가 1초이내에 일어나야한다
= =묵시적 요구사항
즉, 묵시적요구사항이란
효율성, 신뢰성등을 의미한다

- **Definition**
  - Degree of conformance to explicitly stated functional and performance requirements, explicitly documented development standards, and implicit characteristics that are expected of all professionally developed software.

common sense를 이용해서
〉srs에 없더라도 기본적으로 필요한
기능을 추가해야한다(묵시적 기능)

- **Three Emphases**
  - Requirements are the foundation from which quality is measured.
    - Lack of conformance to requirements is lack of quality.
  - Developers should follow the specified standards for development.
  - Software must conform to its implicit requirements as well.

Cost to Correct an Error



Software Quality Assurance (SQA)

S/W 품질보장

- **Two Groups of SQA Tasks**
  - **Software Engineers**
    - Software Engineers who do technical work address quality by applying solid technical methods and measures, conducting formal technical reviews and testing.
  - **SQA Group** 코드를 작성하는데에 관심이 있는것이 아니라, 품질을 보증하기 위해서 확인하는 작업이 주된경우!
    - SQA group responsible for QA planning, oversight, record keeping, analysis and reporting.
    - It assists developers in achieving a high quality end product.

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. 산출물을 review한다
- 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.

Software Reviews

- **Concept** 에러나, 문제점, 결함(error, problem, defect)같은것을 걸러주는 필터이다.
  - **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

Cost Impact of Software Defects

- **Terms**
  - Defect (Fault)
    - Quality problem that is discovered after release
  - Error
    - Quality problem that is discovered by engineers before release

defect 는 s/w 출시이후이기 때문에 error 보다 고치는 코스트가 훨씬 더 비싸다

- **Objective of Review**
  - To find errors during the process that become defects after release of the software.
  - Benefit: Early Discovery of Errors

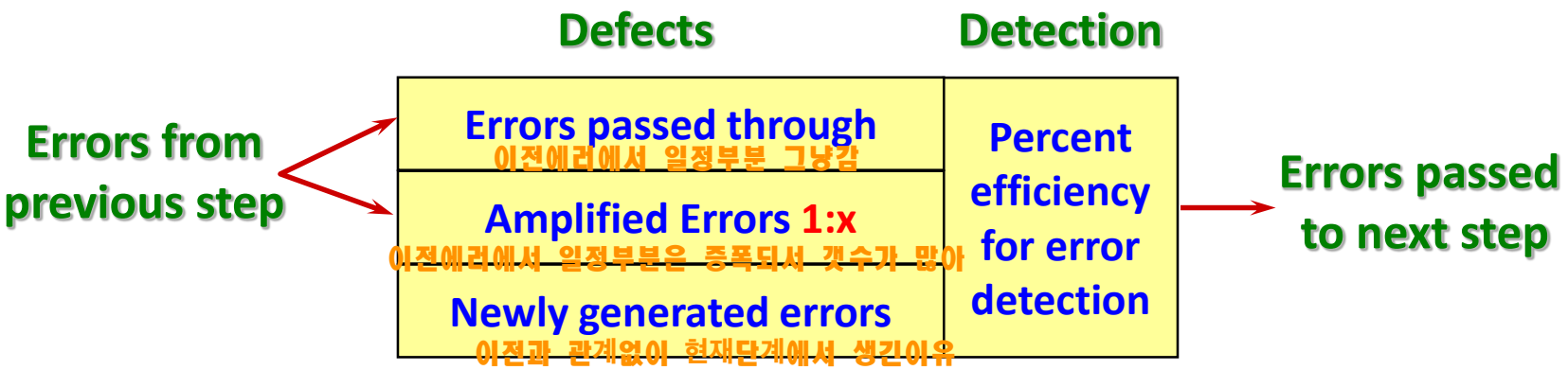
보통 에러는 디자인타임에 발생한다
하지만 계속 리뷰하게 된다면 75%정도 제거할 수 있다
〉〉리뷰를 하라

- **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.

오류증폭모델

Defect Amplification Model

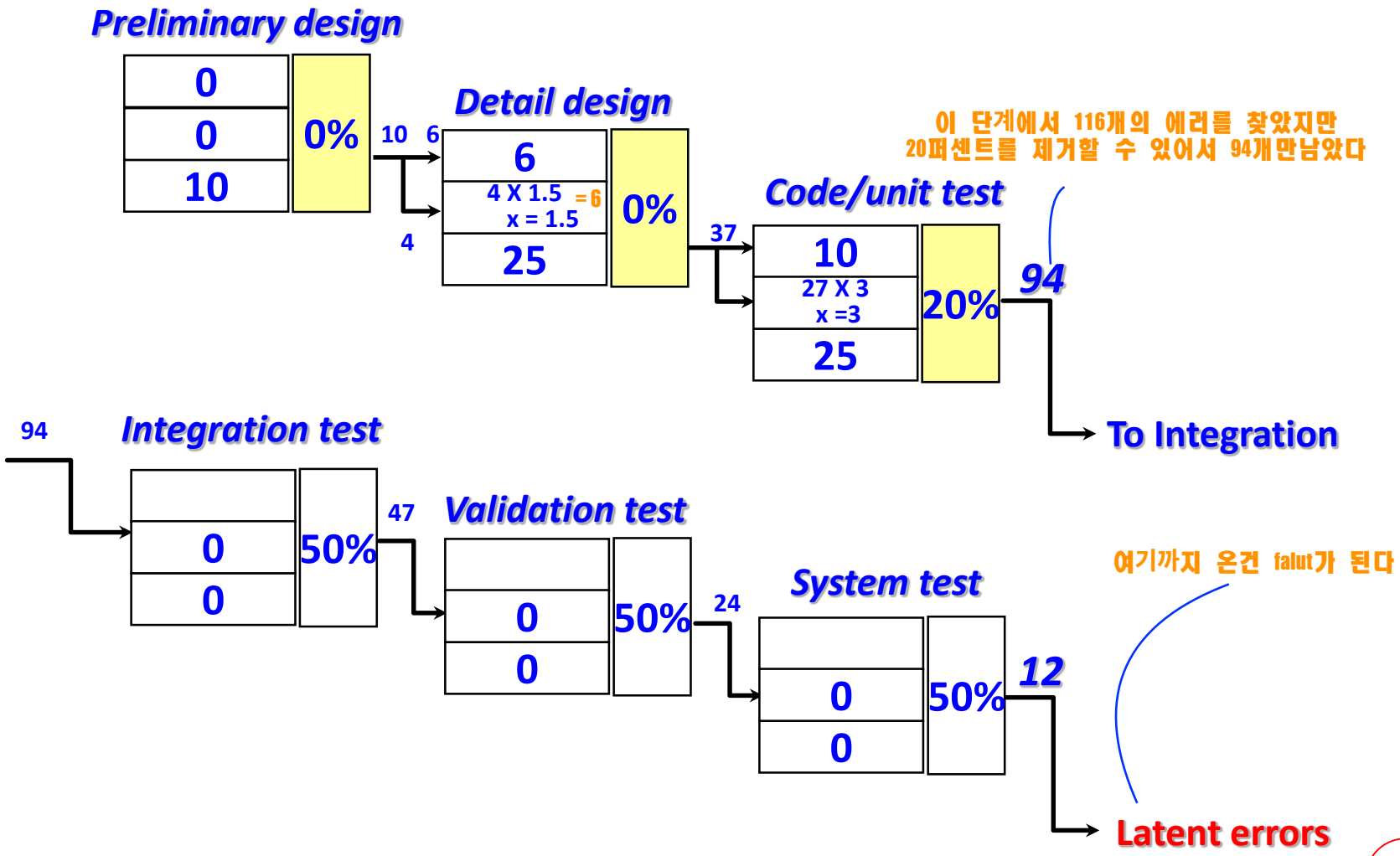
- Errors may be inadvertently generated.
- Illustrates the generation and detection of errors during each of developing software.
  - Analysis, Preliminary Design, Detailed Design, Coding
- Review may fail to uncover newly generated errors and errors from previous steps, resulting in some errors that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



A Development Step

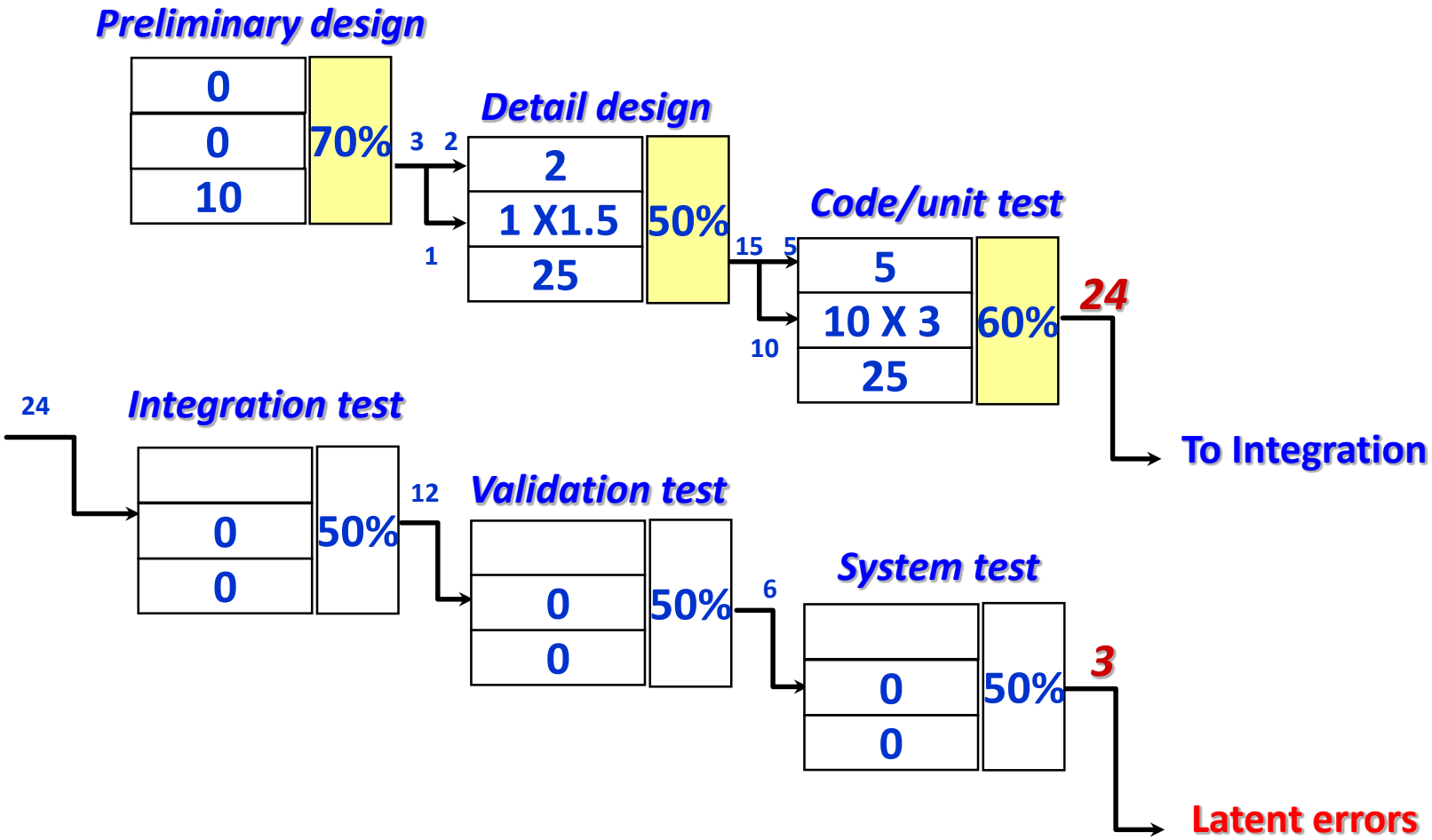
Defect Amplification Model

- **No Reviews applied**



Defect Amplification Model

- **Reviews applied**



정형검토 Formal Technical Review (FTR)

- **FTR is a SQA activity performed by software engineers.**
- **Objectives**
  - To cover 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.**

사전준비 2시간정도 해야한다.

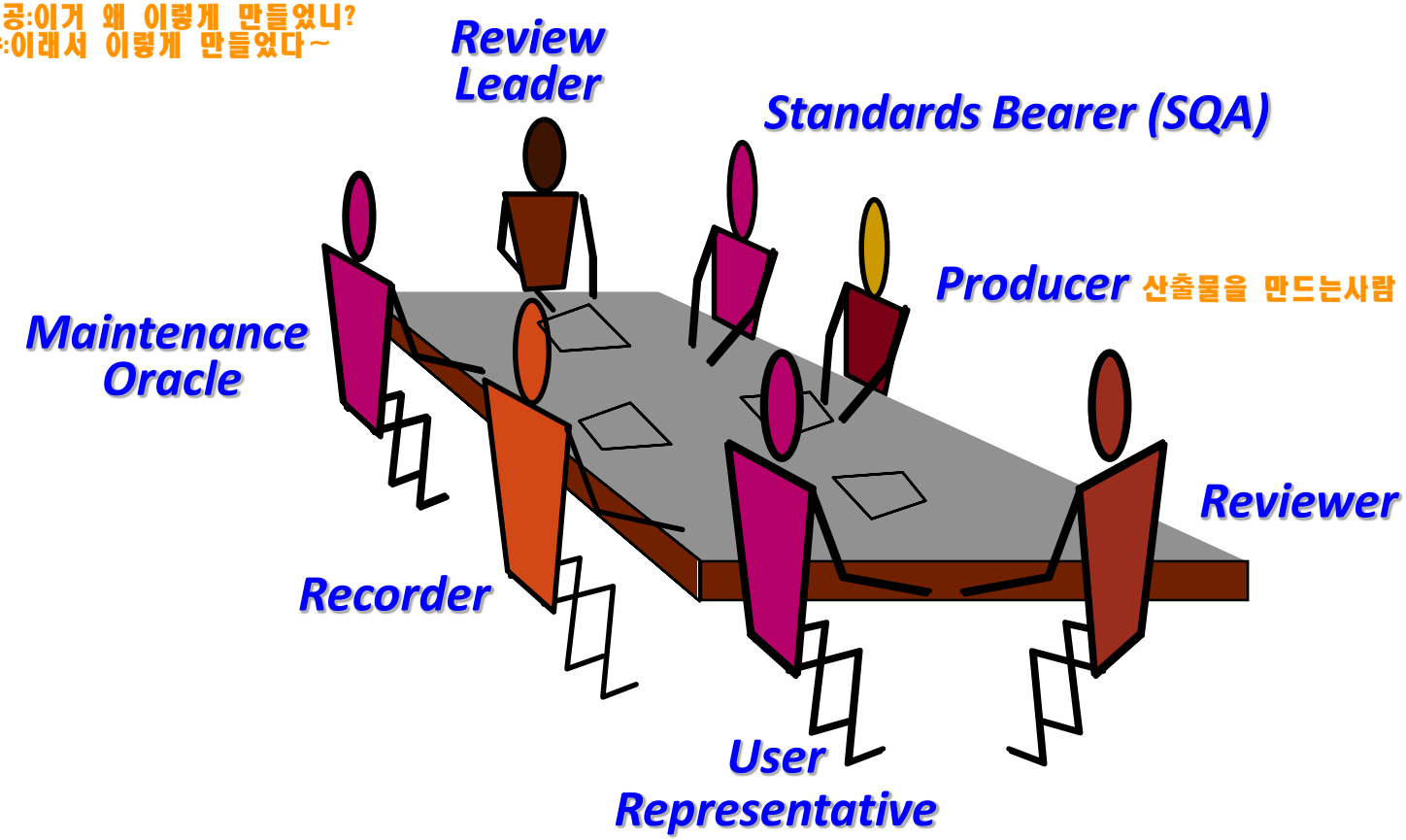
제약,해야하는것

리뷰매니저가〉〉리뷰어들한테 리뷰해야할 자료를 미리 보내주고 확인하라고 하는것

- **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

producer는 수비수
나머지는 공격수
〉〉 공:이거 왜 이렇게 만들었니?
수:이래서 이렇게 만들었다~



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. 가끔 PL == RL인 경우도 있다
- 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: 순서대로 승인,거절, 조건부 승인
  - Accept the work product without further modification.
  - Reject the work product due to severe errors. 산출물이 큰 문제를 가지기 때문에 〉〉 거부하고 다시해와라 라고 하는것
  - Accept the work product provisionally.
    - Minor errors must be corrected, but no additional review is required.

아주 간단한 문제해결

Guidelines for Review Meeting

- Review the product, not the producer.
  - 리뷰하는 사람이 producer와 안좋은 관계가 있을 경우 〉〉 더 안좋게 평가할 가능성이 있음.
    그러니 산출물을 평가해라, 사람말고
- Set an agenda and maintain it.
  - 일정을 정하고 그걸 유지해라
- Limit debate and rebuttal.
  - 논쟁과 반박을 제한해라
- Enunciate problem areas, but don't attempt to solve every problem noted.
- Take written notes.
- Limit the number of participants and insist upon advance preparation.
- Allocate resources and time schedule for FTRs.
- Conduct meaningful training for all reviewers.
- Review your early reviews.

Statistical Quality Assurance

- **Industry prefers more quantitative approach about quality.**

- **General Steps**

1. Information about software defects is collected and categorized. 문제들 모으고
2. Trace each defect to its underlying cause. 원인을 찾고
3. Using the *Pareto* principle, isolate the 20% (the vital few). 파레토법칙〉〉 20%의 사람이 80%결과를 만든다
   - 80% of defects are traced to 20% of all possible causes.
   여기서는 주요한(essential)20%부분의 에러원인을 제거하면 80%의 문제를 해결할 수 있다
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 PL 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.**

분석해보면서 〉〉 에러중 가장 많은 원인에 대해 확인하고
퍼센트가 높은것부터 제거

	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)**

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

  - 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.

PM에 의해 사용될 수 있다

각각의 단계(phase)에 얼마나 많은 에러가 존재하는가
예시
전체에러 100개
serious 30
moderate 40
tiny 30
이렇게 된 경우
각각의 단계에 가중치를 줘서 계산
Ws = 10
Wm = 3
Wt = 1
10 *(30/100) + 3*(40/100) + 1*(30/100)
= =4.5
개수의 개념이 아니라 비교했을때 값이 더 크다면
에러가 더 많거나, 더 심각한 에러가 존재하거나 둘다이다

- **Error Index (EI)**

전체 프로젝트 기간중
PI가 증가했는지 감소했는지 보여주는
지표이다

  - An overall indication of improvement in software quality.
  - $EI = \Sigma(i \times PI_i) / PS$

    $= (PI_1 + 2*PI_2 + 3*PI_3 + i*PI_i) / PS$

    - PS = Size of the product

EI가 높다는 것은 제대로 리뷰가 되지않았다는 것
EI가 낮다는 것은 리뷰가 제대로 되고 에러가 잘 잡혔다는 것