SOFTWARE QUALITY ASSURANCE

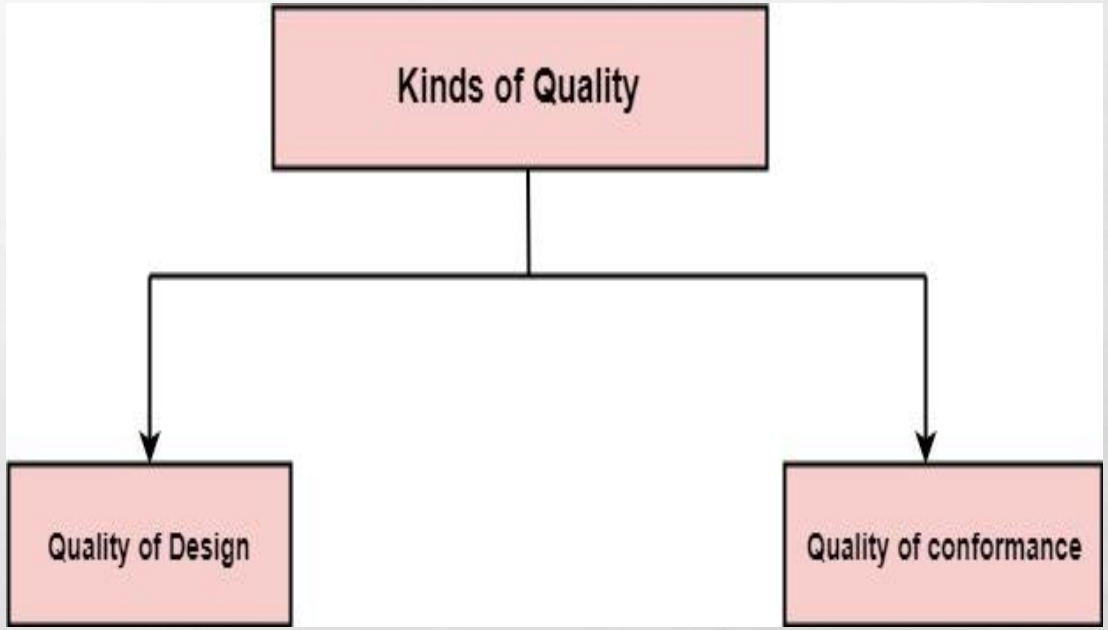
WHAT IS QUALITY?

QUALITY DEFINES TO ANY MEASURABLE CHARACTERISTICS SUCH AS CORRECTNESS, MAINTAINABILITY, PORTABILITY, TESTABILITY, USABILITY, RELIABILITY, EFFICIENCY, INTEGRITY, REUSABILITY, AND INTEROPERABILITY.

TWO KINDS OF QUALITY:

- **QUALITY OF DESIGN:** QUALITY OF DESIGN REFERS TO THE CHARACTERISTICS THAT DESIGNERS SPECIFY FOR AN ITEM. THE GRADE OF MATERIALS, TOLERANCES, AND PERFORMANCE SPECIFICATIONS THAT ALL CONTRIBUTE TO THE QUALITY OF DESIGN.

- **QUALITY OF CONFORMANCE:** QUALITY OF CONFORMANCE IS THE DEGREE TO WHICH THE DESIGN SPECIFICATIONS ARE FOLLOWED DURING MANUFACTURING. GREATER THE DEGREE OF CONFORMANCE, THE HIGHER IS THE LEVEL OF QUALITY OF CONFORMANCE.



- **QUALITY CONTROL:** QUALITY CONTROL INVOLVES A SERIES OF INSPECTIONS, REVIEWS, AND TESTS USED THROUGHOUT THE SOFTWARE PROCESS TO ENSURE EACH WORK PRODUCT MEETS THE REQUIREMENTS PLACE UPON IT. QUALITY CONTROL INCLUDES A FEEDBACK LOOP TO THE PROCESS THAT CREATED THE WORK PRODUCT.

- **QUALITY ASSURANCE:** QUALITY ASSURANCE IS THE PREVENTIVE SET OF ACTIVITIES THAT PROVIDE GREATER CONFIDENCE THAT THE PROJECT WILL BE COMPLETED SUCCESSFULLY.

IMPORTANCE OF QUALITY

- **INCREASING CRITICALITY OF SOFTWARE**
- **THE INTANGIBILITY OF SOFTWARE**
- **ACCUMULATING ERRORS DURING SOFTWARE DEVELOPMENT**

QUALITY MANAGEMENT SYSTEM (QMS)

- A QUALITY MANAGEMENT SYSTEM (QMS) IS A **FORMALIZED SYSTEM** THAT DOCUMENTS PROCESSES, PROCEDURES, AND RESPONSIBILITIES FOR ACHIEVING QUALITY POLICIES AND OBJECTIVES.

- A QMS HELPS **COORDINATE** AND DIRECT AN ORGANIZATION'S **ACTIVITIES** TO **MEET CUSTOMER AND REGULATORY REQUIREMENTS** AND IMPROVE ITS EFFECTIVENESS AND EFFICIENCY ON A CONTINUOUS BASIS.

QUALITY MANAGEMENT SYSTEM (QMS)

QUALITY MANAGEMENT SYSTEMS SERVE MANY PURPOSES, INCLUDING:

- IMPROVING PROCESSES
- REDUCING WASTE
- LOWERING COSTS
- FACILITATING AND IDENTIFYING TRAINING OPPORTUNITIES
- ENGAGING STAFF
- SETTING ORGANIZATION-WIDE DIRECTION

BENEFITS OF QUALITY MANAGEMENT SYSTEMS

- **MEETING THE CUSTOMER'S REQUIREMENTS,** WHICH HELPS TO INSTIL CONFIDENCE IN THE ORGANIZATION, IN TURN LEADING TO MORE CUSTOMERS, MORE SALES, AND MORE REPEAT BUSINESS

- **MEETING THE ORGANIZATION'S REQUIREMENTS,** WHICH ENSURES COMPLIANCE WITH REGULATIONS AND PROVISION OF PRODUCTS AND SERVICES IN THE MOST COST- AND RESOURCE-EFFICIENT MANNER, CREATING ROOM FOR EXPANSION, GROWTH, AND PROFIT

SOFTWARE QUALITY ASSURANCE (SQA)

- **DEFINITION:**
  - CONFORMANCE TO EXPLICITLY STATED FUNCTIONAL AND PERFORMANCE REQUIREMENTS, EXPLICITLY DOCUMENTED DEVELOPMENT STANDARDS, AND IMPLICIT CHARACTERISTICS THAT ARE EXPECTED OF ALL PROFESSIONALLY DEVELOPED SOFTWARE.

- DEFINITION SERVES TO EMPHASIZE THREE IMPORTANT POINTS:
  - SOFTWARE REQUIREMENTS ARE THE FOUNDATION FROM WHICH QUALITY IS MEASURED. **LACK OF CONFORMANCE TO REQUIREMENTS IS LACK OF QUALITY**.
  - SPECIFIED STANDARDS DEFINE A SET OF DEVELOPMENT CRITERIA, **IF THE CRITERIA ARE NOT FOLLOWED, LACK OF QUALITY** WILL ALMOST SURELY RESULT.
  - A SET OF **IMPLICIT REQUIREMENTS OFTEN GOES UNMENTIONED**. IF SOFTWARE CONFORMS TO ITS EXPLICIT REQUIREMENTS BUT FAILS TO MEET IMPLICIT REQUIREMENTS, SOFTWARE QUALITY IS SUSPECT

SQA GROUP ACTIVITIES

- SQA GROUP MADE UP OF **SOFTWARE ENGINEERS, PROJECT MANAGERS, CUSTOMERS, SALESPEOPLE, AND THE INDIVIDUALS MEMBERS.**
- SOFTWARE QUALITY ASSURANCE IS COMPOSED OF TWO DIFFERENT CONSTITUENCIES.
  - **SOFTWARE ENGINEERS** WHO DO TECHNICAL WORK
  - **SQA GROUP** THAT HAS RESPONSIBILITY FOR QUALITY ASSURANCE PLANNING, OVERSIGHT, RECORD KEEPING, ANALYSIS, AND REPORTING.
- SOFTWARE ENGINEERS ADDRESS **QUALITY ACTIVITIES** BY APPLYING TECHNICAL METHODS AND MEASURES, CONDUCTING FORMAL TECHNICAL REVIEWS, AND PERFORMING WELL-PLANNED SOFTWARE TESTING.
- SQA GROUP IS TO **ASSIST THE SOFTWARE TEAM** IN ACHIEVING A HIGH QUALITY END PRODUCT.

ROLE OF AN SQA GROUP

1. PREPARES AN SQA PLAN FOR A PROJECT.

- THE PLAN IS DEVELOPED DURING PROJECT PLANNING AND IS REVIEWED BY ALL STAKEHOLDERS.
- THE PLAN IDENTIFIES
  - EVALUATIONS TO BE PERFORMED
  - AUDITS AND REVIEWS TO BE PERFORMED
  - STANDARDS THAT ARE APPLICABLE TO THE PROJECT
  - PROCEDURES FOR ERROR REPORTING AND TRACKING
  - DOCUMENTS TO BE PRODUCED BY THE SQA GROUP
  - AMOUNT OF FEEDBACK PROVIDED TO THE SOFTWARE PROJECT TEAM

2. **PARTICIPATES IN THE DEVELOPMENT OF THE PROJECT'S SOFTWARE PROCESS DESCRIPTION.**
   - THE SQA GROUP REVIEWS THE PROCESS DESCRIPTION FOR COMPLIANCE WITH ORGANIZATIONAL POLICY, INTERNAL SOFTWARE STANDARDS, EXTERNALLY IMPOSED STANDARDS (E.G., ISO-9001), AND OTHER PARTS OF THE SOFTWARE PROJECT PLAN.

3. **REVIEWS SOFTWARE ENGINEERING ACTIVITIES TO VERIFY COMPLIANCE WITH THE DEFINED SOFTWARE PROCESS.**
   - THE SQA GROUP IDENTIFIES, DOCUMENTS, AND TRACKS DEVIATIONS FROM THE PROCESS AND VERIFIES THAT CORRECTIONS HAVE BEEN MADE.

4. **AUDITS DESIGNATED SOFTWARE WORK PRODUCTS TO VERIFY COMPLIANCE WITH THOSE DEFINED AS PART OF THE SOFTWARE PROCESS.**
   - THE SQA GROUP REVIEWS SELECTED WORK PRODUCTS; IDENTIFIES, DOCUMENTS, AND TRACKS DEVIATIONS; VERIFIES THAT CORRECTIONS HAVE BEEN MADE; AND PERIODICALLY REPORTS THE RESULTS OF ITS WORK TO THE PROJECT MANAGER.

5. ENSURES THAT DEVIATIONS IN SOFTWARE WORK AND WORK PRODUCTS ARE DOCUMENTED AND HANDLED ACCORDING TO A DOCUMENTED PROCEDURE.

- DEVIATIONS MAY BE ENCOUNTERED IN THE PROJECT PLAN, PROCESS DESCRIPTION, APPLICABLE STANDARDS, OR TECHNICAL WORK PRODUCTS.

6. **RECORDS ANY NONCOMPLIANCE AND REPORTS TO SENIOR MANAGEMENT.**

- NONCOMPLIANCE ITEMS ARE TRACKED UNTIL THEY ARE RESOLVED.

SOFTWARE REVIEW

- SOFTWARE REVIEWS ARE A "**FILTER**" FOR THE SOFTWARE ENGINEERING PROCESS.
- THAT IS, REVIEWS ARE APPLIED AT VARIOUS POINTS DURING SOFTWARE DEVELOPMENT **AND SERVE TO UNCOVER ERRORS AND DEFECTS** THAT CAN THEN BE REMOVED.
- TYPES OF REVIEW
  - **INFORMAL REVIEW**
    - MEETING AROUND THE COFFEE MACHINE AND DISCUSSING TECHNICAL PROBLEMS.
  - **FORMAL REVIEW**
    - FORMAL PRESENTATION OF SOFTWARE DESIGN TO AN AUDIENCE OF CUSTOMERS, MANAGEMENT, AND TECHNICAL STAFF
- A FORMAL TECHNICAL REVIEW IS THE MOST EFFECTIVE FILTER FROM A QUALITY ASSURANCE STANDPOINT.

COST IMPACT OF SOFTWARE DEFECTS

- THE PRIMARY OBJECTIVE OF FORMAL TECHNICAL REVIEWS IS TO FIND ERRORS DURING THE PROCESS SO THAT THEY DO NOT BECOME DEFECTS AFTER RELEASE OF THE SOFTWARE.
- DIRECT BENEFIT IS EARLY DISCOVERY OF ERROR, DO NOT PROPAGATE TO THE NEXT STEP.
- **DESIGN** ACTIVITIES INTRODUCE BETWEEN **50 AND 65 PERCENT OF ALL ERRORS** (AND ULTIMATELY, ALL DEFECTS) DURING THE SOFTWARE PROCESS.
- HOWEVER, FTR HAVE BEEN SHOWN TO BE UP TO 75 PERCENT EFFECTIVE IN UNCOVERING DESIGN FLAWS.
- FTP SUBSTANTIALLY REDUCES THE COST OF SUBSEQUENT STEPS IN THE DEVELOPMENT AND SUPPORT PHASES.

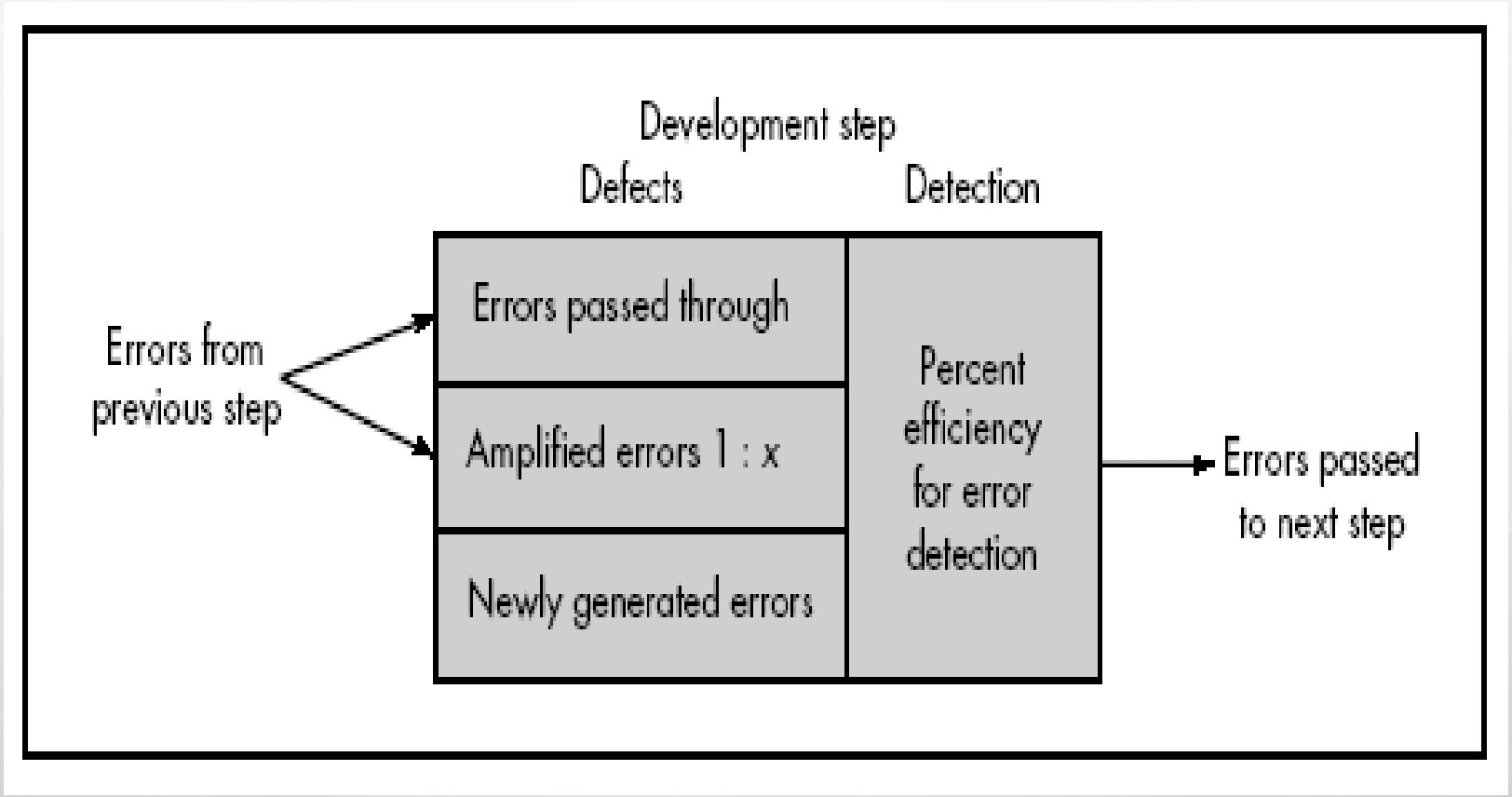
CONTD.

- ASSUME THAT AN ERROR UNCOVERED DURING **DESIGN WILL COST 1.0 MONETARY UNIT** TO CORRECT.
- RELATIVE TO THIS COST, THE SAME ERROR UNCOVERED JUST **BEFORE TESTING COMMENCES WILL COST 6.5 UNITS**.
- DURING **TESTING IS 15 UNITS.**
- AFTER **RELEASE, BETWEEN 60 AND 100 UNITS.**

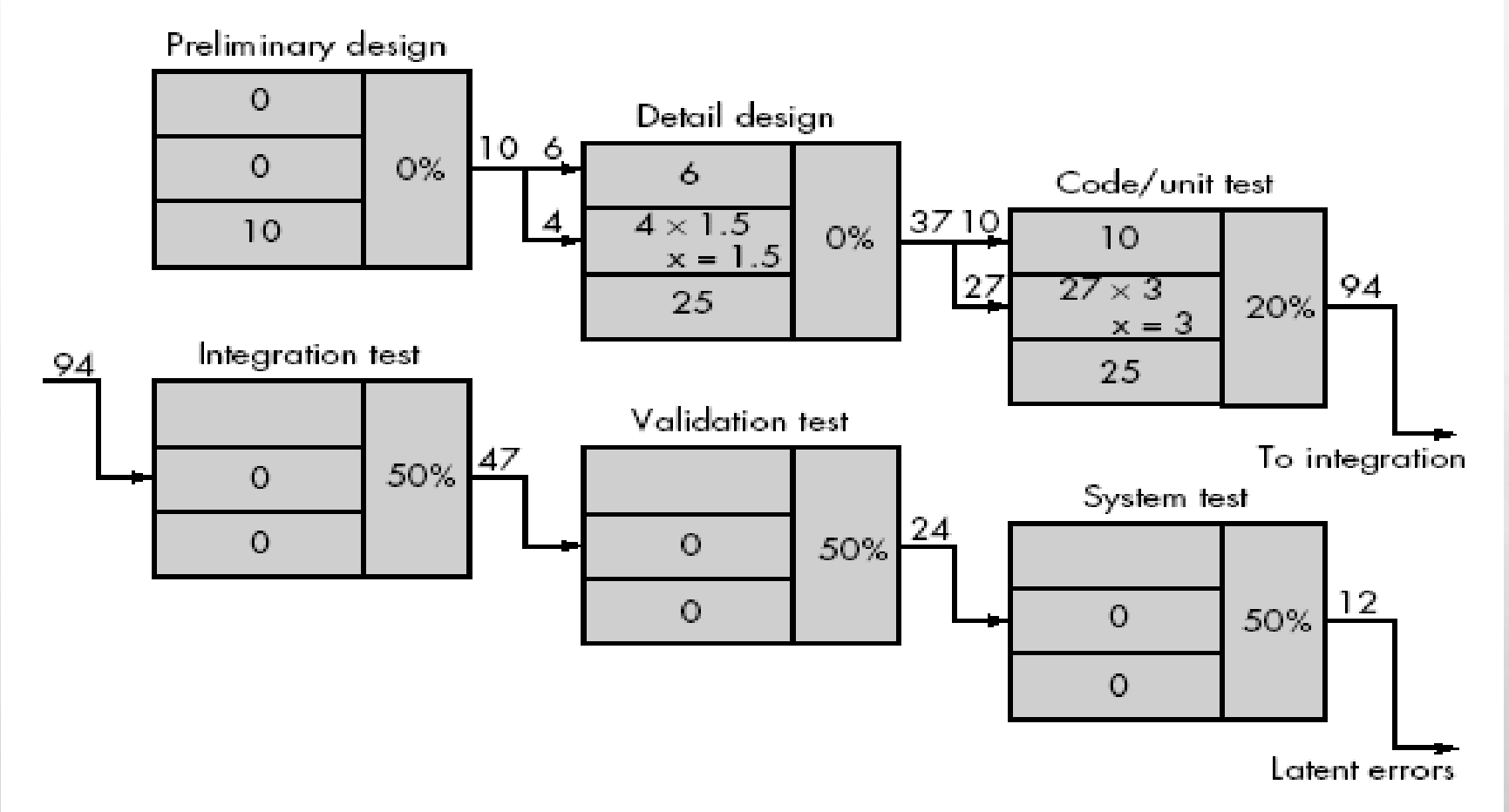
DEFECT AMPLIFICATION AND REMOVAL

- A DEFECT AMPLIFICATION MODEL CAN BE USED TO ILLUSTRATE THE GENERATION AND DETECTION OF ERRORS DURING THE PRELIMINARY DESIGN, DETAIL DESIGN, AND CODING STEPS OF THE SOFTWARE ENGINEERING PROCESS.
- A BOX REPRESENTS A SOFTWARE DEVELOPMENT STEP. DURING THE STEP, ERRORS MAY BE BY MISTAKE GENERATED.
- REVIEW MAY FAIL TO UNCOVER NEWLY GENERATED ERRORS AND ERRORS FROM PREVIOUS STEPS, RESULTING IN SOME NUMBER OF ERRORS THAT ARE PASSED THROUGH.
- IN SOME CASES, ERRORS PASSED THROUGH FROM PREVIOUS STEPS ARE AMPLIFIED (AMPLIFICATION FACTOR, *X*) BY CURRENT WORK.
- THE BOX SUBDIVISIONS REPRESENT EACH OF THESE CHARACTERISTICS AND THE PERCENT OF EFFICIENCY FOR DETECTING ERRORS.

DEFECT AMPLIFICATION MODEL

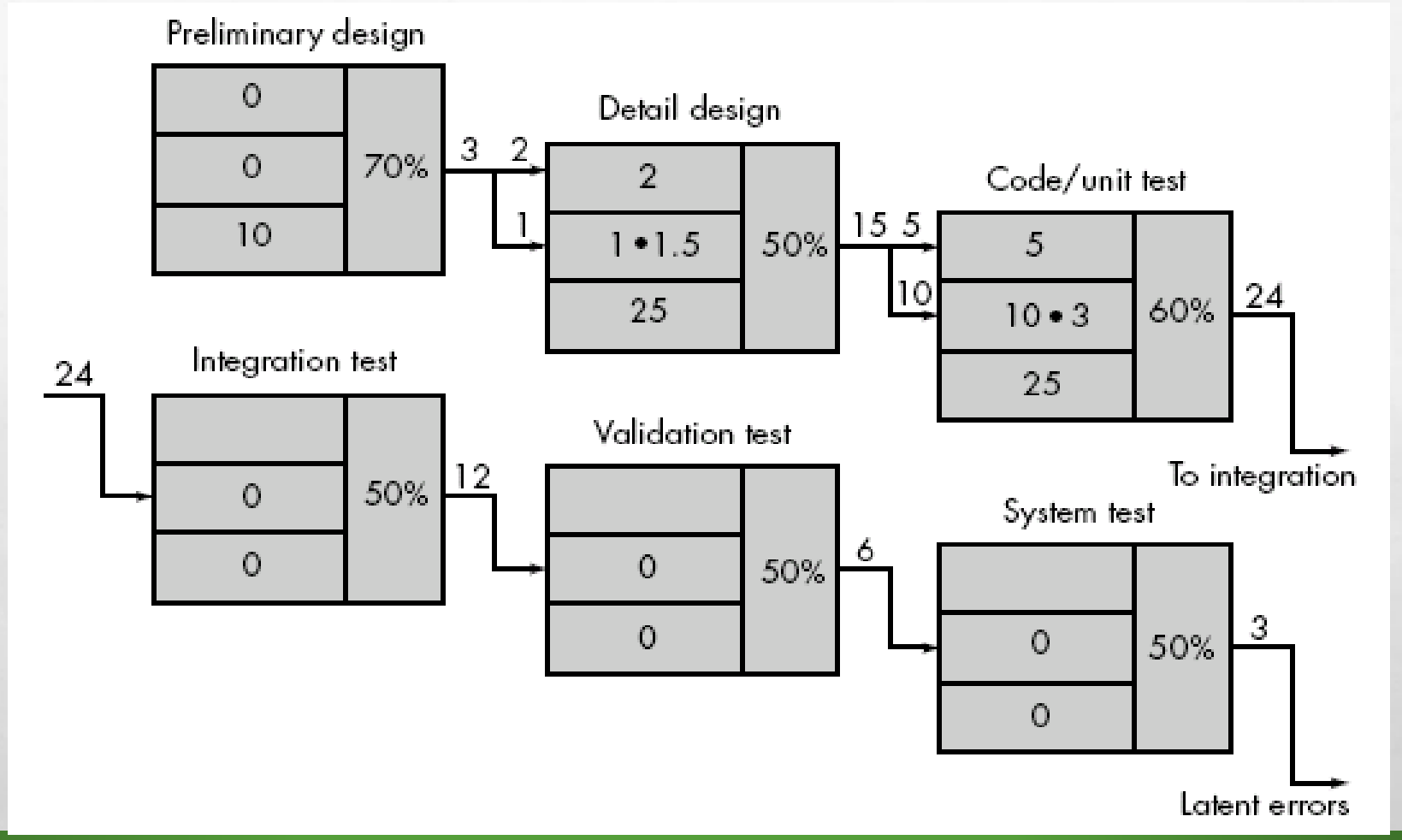


DEFECT AMPLIFICATION – NO REVIEW



- BECAUSE OF NO REVIEW CONDUCTED, TEN PRELIMINARY DESIGN DEFECTS ARE AMPLIFIED TO 94 ERRORS BEFORE TESTING COMMENCES.
- EACH TEST STEP IS ASSUMED TO CORRECT 50 PERCENT OF ALL INCOMING ERRORS WITHOUT INTRODUCING ANY NEW ERRORS.
- TWELVE LATENT ERRORS ARE RELEASED TO THE FIELD OR TO THE CUSTOMERS.

DEFECT AMPLIFICATION –REVIEW CONDUCTED



- CONSIDERS THE SAME CONDITIONS EXCEPT THAT DESIGN AND CODE REVIEWS ARE CONDUCTED AS PART OF EACH DEVELOPMENT STEP.
- IN THIS CASE, TEN INITIAL PRELIMINARY DESIGN ERRORS ARE AMPLIFIED TO 24 ERRORS BEFORE TESTING COMMENCES.
- ONLY THREE LATENT ERRORS EXIST.
- NOW WE CAN ESTIMATE OVERALL COST (WITH AND WITHOUT REVIEW FOR OUR HYPOTHETICAL EXAMPLE)
- USING THESE DATA, THE TOTAL COST FOR DEVELOPMENT AND MAINTENANCE WHEN REVIEWS ARE CONDUCTED.
- TO CONDUCT REVIEWS, A SOFTWARE ENGINEER MUST EXPEND TIME AND EFFORT AND THE DEVELOPMENT ORGANIZATION MUST SPEND MONEY.

COLLECTING SOFTWARE ENGINEERING DATA

- THE CHALLENGE OF COLLECTING SOFTWARE ENGINEERING DATA IS TO MAKE SURE THAT THE COLLECTED DATA CAN PROVIDE **USEFUL INFORMATION FOR PROJECT, PROCESS, AND QUALITY MANAGEMENT**.
- THE DATA MUST BE BASED ON **WELL-DEFINED METRICS AND MODELS,** WHICH ARE USED TO DRIVE IMPROVEMENTS.
- THE AMOUNT OF DATA TO BE COLLECTED AND THE **NUMBER OF METRICS TO BE USED NEED NOT BE VAST.**
- IT IS MORE IMPORTANT THAT THE INFORMATION EXTRACTED FROM THE DATA BE FOCUSED, ACCURATE, AND USEFUL THAN THAT IT BE PLENTIFUL.
- WITHOUT BEING METRICS DRIVEN, **OVER-COLLECTION** OF DATA COULD BE WASTEFUL.

- BASILI AND WEISS PROPOSE A DATA COLLECTION METHODOLOGY THAT COULD BE APPLICABLE ANYWHERE.
- THE SCHEMA CONSISTS OF SIX STEPS WITH CONSIDERABLE FEEDBACK AND ITERATION OCCURRING AT SEVERAL PLACES:
    - ESTABLISH THE GOAL OF THE DATA COLLECTION.
    - DEVELOP A LIST OF QUESTIONS OF INTEREST.
    - ESTABLISH DATA CATEGORIES.
    - DESIGN AND TEST DATA COLLECTION FORMS.
    - COLLECT AND VALIDATE DATA.
    - ANALYZE DATA.