Quality Concepts

Different views of Quality

Transcendental View

Immediately recognize, but cannot explicitly define

User View

Quality in terms of an end user's specific goals

Manufacturer's View

Quality in terms of the original specification of the product

Product View

Functions and Features of product

Value-Based View

How much a customer is willing to pay for a product

Quality of Design

User Satisfaction = compliant product + good quality + delivery within budget and schedule

Software Quality

An effective software process applied in a manner that creates a **useful product that provides measurable value** for those **who produced it** and those **who use it**.

Effective software process
Useful product
Adding value for both the producer and user

David Garvin's Quality Dimensions

Performance quality

Feature quality

Reliability

Conformance

Durability

Serviceability

Aesthetics

Perception

McCall's Quality Factors

Correctness

Reliability

Efficiency

Integrity

Usability

Maintainability

Flexibility

Testability

Portability

Reusability

Interoperability

ISO 9126 Quality Factors

Functionality

Reliability

Usability

Efficiency

Maintainability

Portability

Achieving Software Quality

Activities that help a software team achieve high software quality

- Software Engineering Methods
- Project Management Techniques
- Quality Control
- Quality Assurance

Review Techniques

- Software reviews are a filter for the software process
- Serve to uncover errors and defects that can then be removed
- Purify software engineering work products

- Informal meeting
- Formal presentation

Review Metrics

Preparation effort – E_P (in Person-hours)

Assessment effort – E_A (in Person-hours)

Rework effort – E_R (in Person-hours)

Work product size – WPS (LoC, No.of Use Cases, Documents)

Minor errors found – Err_{Minor} (Requiring less effort)

Major error found – Err_{Major} (Requiring more effort)

Analyzing Metrics

Total review effort

$$E_{REVIEW} = E_P + E_A + E_R$$

Total number of errors discovered

$$Err_{TOT} = Err_{MINOR} + Err_{MAJOR}$$

Error Density

$$\text{Error Density} = Err_{TOT} / WPS$$

Analyzing Metrics

The requirements model contains 18 UML diagrams as part of 32 overall pages of descriptive materials. The review uncovers 18 minor errors and 4 major errors.

Total number of errors discovered

$$Err_{TOT} = Err_{MINOR} + Err_{MAJOR}$$

Error Density

$$\text{Error Density} = Err_{TOT} / WPS$$

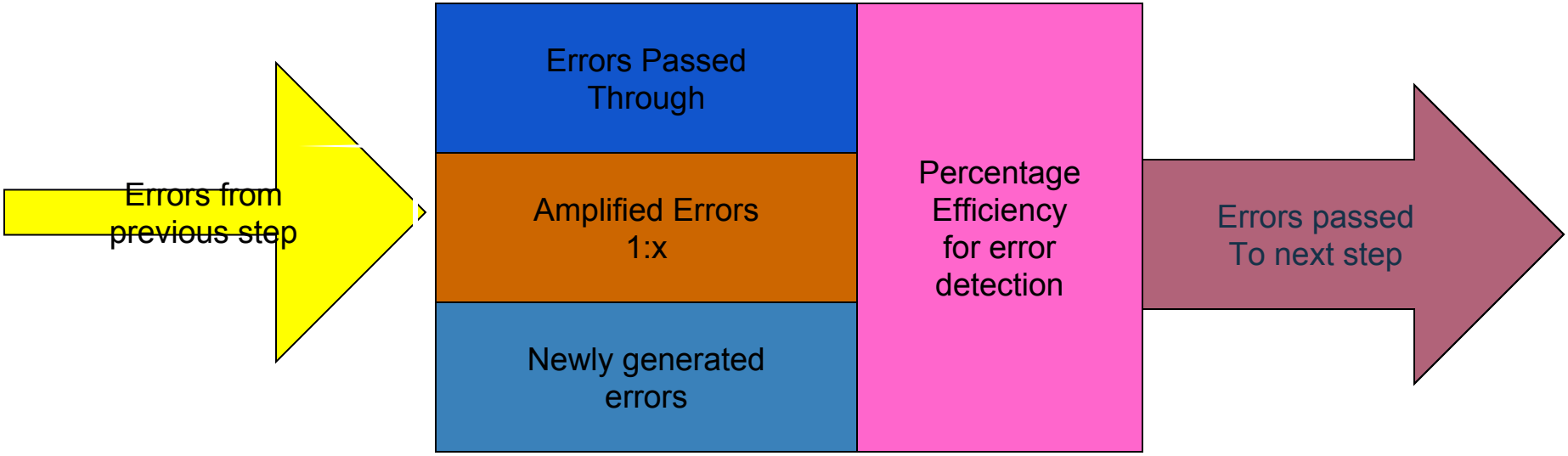
Error density is 1.2 errors per UML diagram or 0.68 errors per requirements model page.

Analyzing Metrics

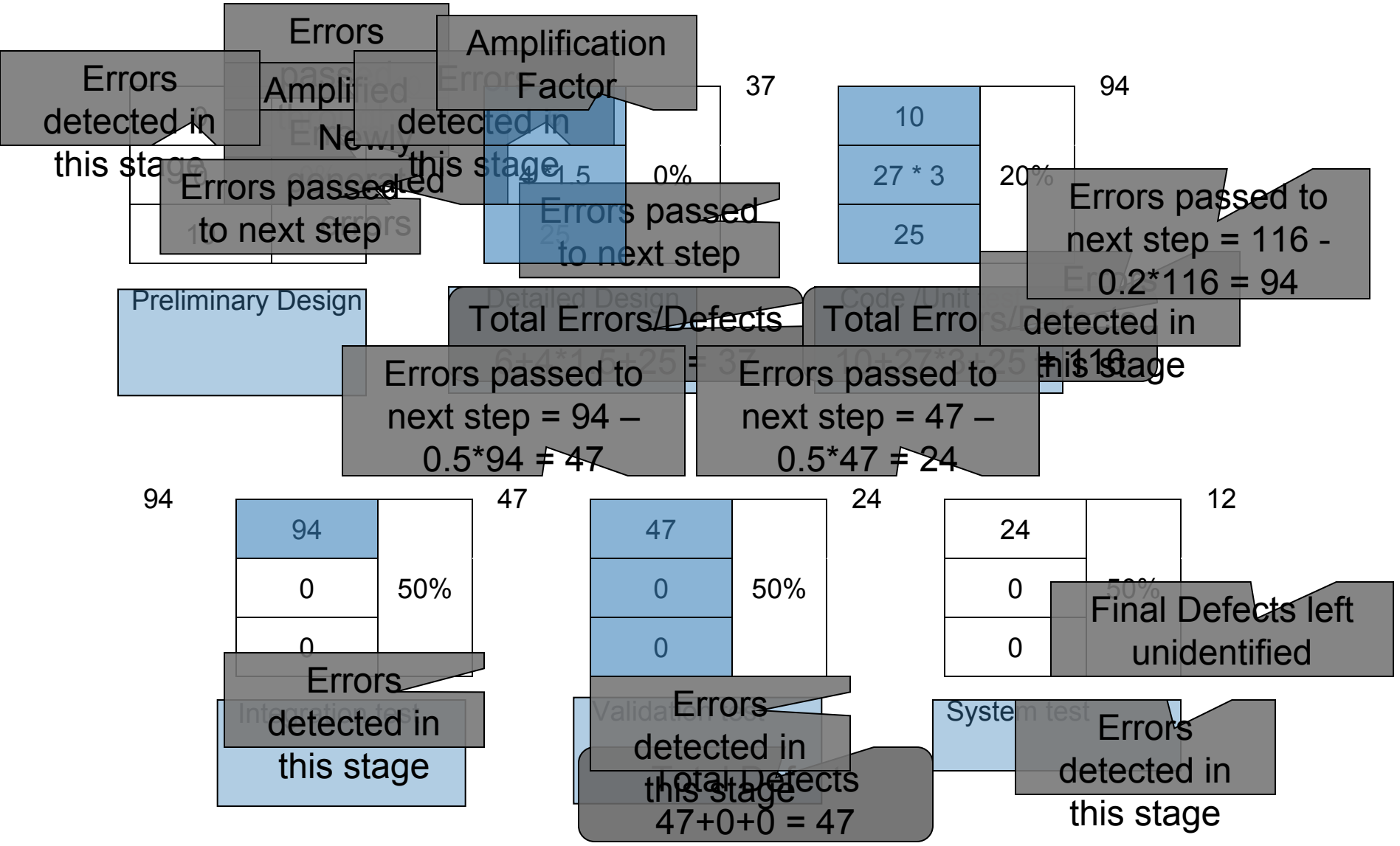
Average values for error density enable you to estimate the number of errors to be found in a new un reviewed document.

For example, if the average error density for a requirements model is 0.6 errors per page, and a new requirement model is 32 pages long, a rough estimate suggests that your software team will find about 19 or 20 errors during the review of the document. If you find only 6 errors, you've done an extremely good job in developing the requirements model or your review approach was not thorough enough.

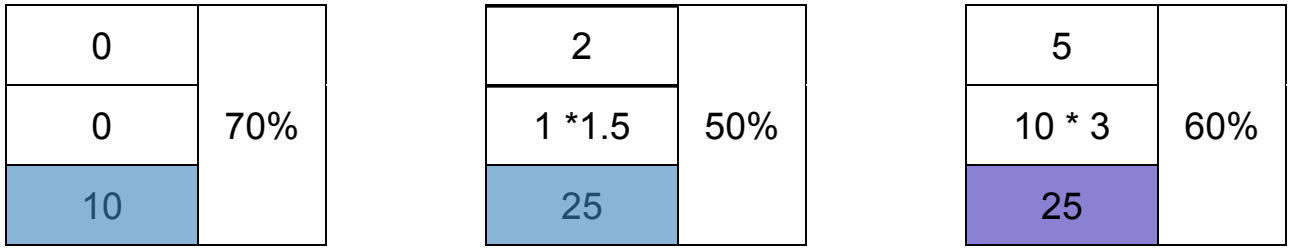
Defect Amplification Model

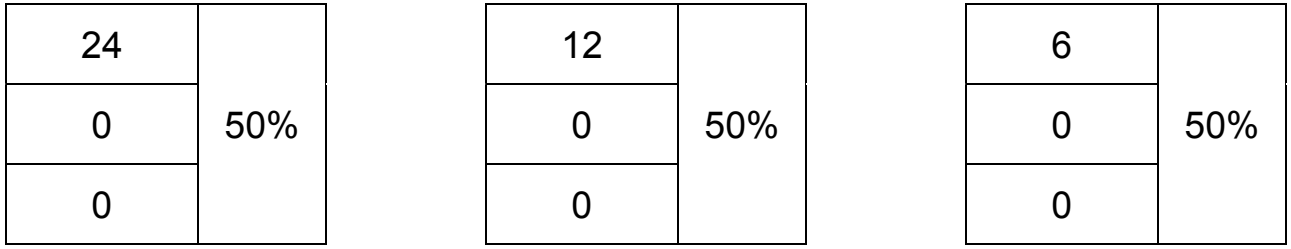


Development Step


Errors
Amplification Factor
Errors detected in this stage
Errors passed to next step
37
94
10
27 * 3
25
20%
Errors passed to next step = 116 - 0.2*116 = 94
Preliminary Design
Total Errors/Defects
Errors passed to next step = 94 – 0.5*94 = 47
Errors passed to next step = 47 – 0.5*47 = 24
94
47
24
12
94
0
0
50%
47
0
0
50%
24
0
0
50%
Final Defects left unidentified
Errors detected in this stage
Total Defects 47+0+0 = 47
Errors detected in this stage

After Introducing Inspections at the Design and Code Level





Informal Reviews

A simple desk check or a casual meeting conducted with a colleague.

No advance planning and preparation, no agenda, no follow-up on the errors.

Formal Technical Reviews

Is a software quality control activity.

Properly planned and controlled.

Objectives are:

✓To uncover errors

✓To verify the software under review meets its requirements

✓To ensure that the software has been represented according to predefined standards

✓To make projects more manageable

Formal Technical Review

The Review Meeting

- Between three and five people
- Should be less than two hours
- Producer, Review Leader, Reviewer, Recorder

Review Reporting and Record Keeping

- What was reviewed?
- Who Reviewed it?
- What were the findings and conclusions?

Review Guidelines

- ✓Review the product, Not the producer.
- ✓Set an agenda and maintain it.
- ✓Limit debate
- ✓Enunciate problem areas
- ✓Take written notes
- ✓Limit the number of participants
- ✓Develop a check list for each product that is likely to be reviewed.
- ✓Allocate resource and schedule time for FTR
- ✓Review your early reviews

Sample Based Review

Software Quality Assurance (SQA)

Objective is to produce on-time, high quality software

What is it?

✓Explicitly define what is meant when you say software quality.

✓Create a set of activities that will help ensure that every software engineering work product exhibits high quality

✓Perform quality control and assurance activities on every software project

✓Use metrics to develop strategies for improving software process

Elements of SQA

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

SQA Tasks

Prepares an SQA plan for a project.

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

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.

SQA Tasks

Reviews software engineering activities to verify compliance with the defined software process.

identifies, documents, and tracks deviations from the process and verifies that corrections have been made.

Audits designated software work products to verify compliance with those defined as part of the software process.

reviews selected work products; identifies, documents, and tracks deviations; verifies that corrections have been made

periodically reports the results of its work to the project manager.

Ensures that deviations in software work and work products are documented and handled according to a documented procedure.

Records any noncompliance and reports to senior management.

Noncompliance items are tracked until they are resolved.

SQA Goals

Requirements quality.
The correctness, completeness, and consistency of the requirements model will have a strong influence on the quality of all work products that follow.

Design quality.
Every element of the design model should be assessed by the software team to ensure that it exhibits high quality and that the design itself conforms to requirements.

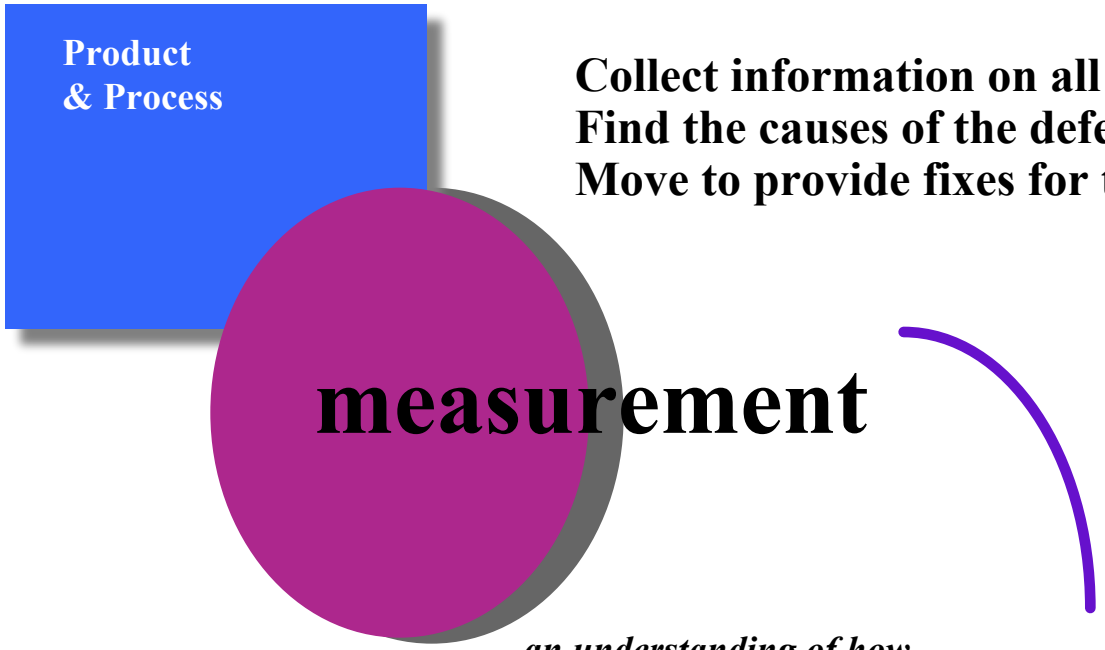
Code quality.
Source code and related work products (e.g., other descriptive information) must conform to local coding standards and exhibit characteristics that will facilitate maintainability.

Quality control effectiveness.
A software team should apply limited resources in a way that has the highest likelihood of achieving a high quality result.

Requirement quality	Ambiguity	Number of ambiguous modifiers (e.g., many, large, human-friendly)
	Completeness	Number of TBA, TBD
	Understandability	Number of sections/subsections
	Volatility	Number of changes per requirement
		Time (by activity) when change is requested
	Traceability	Number of requirements not traceable to design/code
	Model clarity	Number of UML models
		Number of descriptive pages per model
		Number of UML errors
Design quality	Architectural integrity	Existence of architectural model
	Component completeness	Number of components that trace to architectural model
		Complexity of procedural design
	Interface complexity	Average number of pick to get to a typical function or content
		Layout appropriateness
	Patterns	Number of patterns used

Code quality	Complexity	Cyclomatic complexity
	Maintainability	Design factors (Chapter 8)
	Understandability	Percent internal comments
		Variable naming conventions
	Reusability	Percent reused components
	Documentation	Readability index
QC effectiveness	Resource allocation	Staff hour percentage per activity
	Completion rate	Actual vs. budgeted completion time
	Review effectiveness	See review metrics (Chapter 14)
	Testing effectiveness	Number of errors found and criticality
		Effort required to correct an error
		Origin of error

Formal Approaches to SQA



Collect information on all defects
Find the causes of the defects
Move to provide fixes for the process

... an understanding of how

to improve quality ...

Statistical SQA

✓Information about software errors and defects is collected and categorized.

✓An attempt is made to trace each error and defect to its underlying cause (e.g., non-conformance to specifications, design error, violation of standards, poor communication with the customer).

✓Using the Pareto principle (80 percent of the defects can be traced to 20 percent of all possible causes), isolate the 20 percent (the *vital few*).

✓Once the vital few causes have been identified, move to correct the problems that have caused the errors and defects.

Six Sigma for Software Engineering

Define customer requirements and deliverables and project goals via well-defined methods of customer communication

Measure the existing process and its output to determine current quality performance (collect defect metrics)

Analyze defect metrics and determine the vital few causes.

Improve the process by eliminating the root causes of defects.

Control the process to ensure that future work does not reintroduce the causes of defects.

Software Reliability

The probability of failure-free operation of a computer program in a specified environment for a specified time.

A simple measure of reliability is *mean-time-between-failure* (MTBF), where

MTBF = MTTF + MTTR

The acronyms MTTF and MTTR are *mean-time-to-failure* and *mean-time-to-repair*, respectively.

Software Availability

Software availability is the probability that a program is operating according to requirements at a given point in time and is defined as

Availability = [MTTF/(MTTF + MTTR)] x 100%

Software Safety

Software safety is a software quality assurance activity that focuses on the identification and assessment of potential hazards that may affect software negatively and cause an entire system to fail.

If hazards can be identified early in the software process, software design features can be specified that will either eliminate or control potential hazards.