CISC/CMPE 327 Software Quality Assurance

Queen's University, 2019-fall

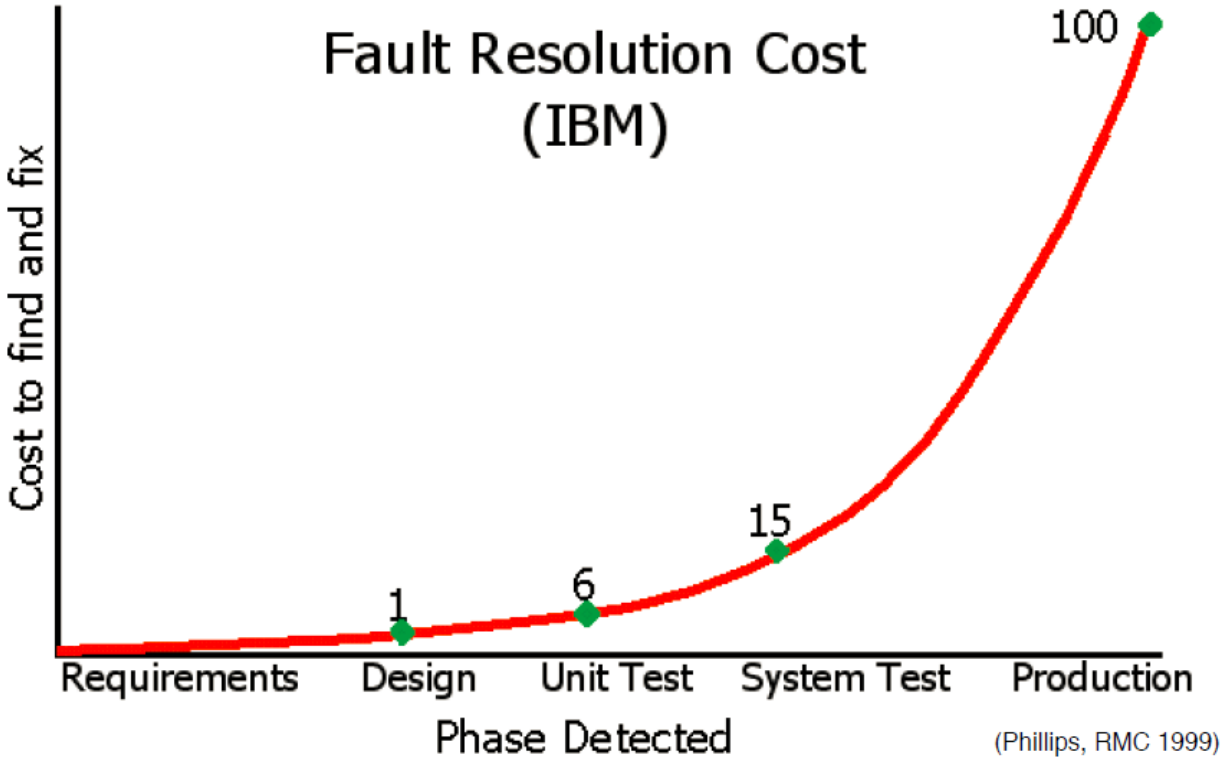
Lecture #24-26 Inspection

Inspections

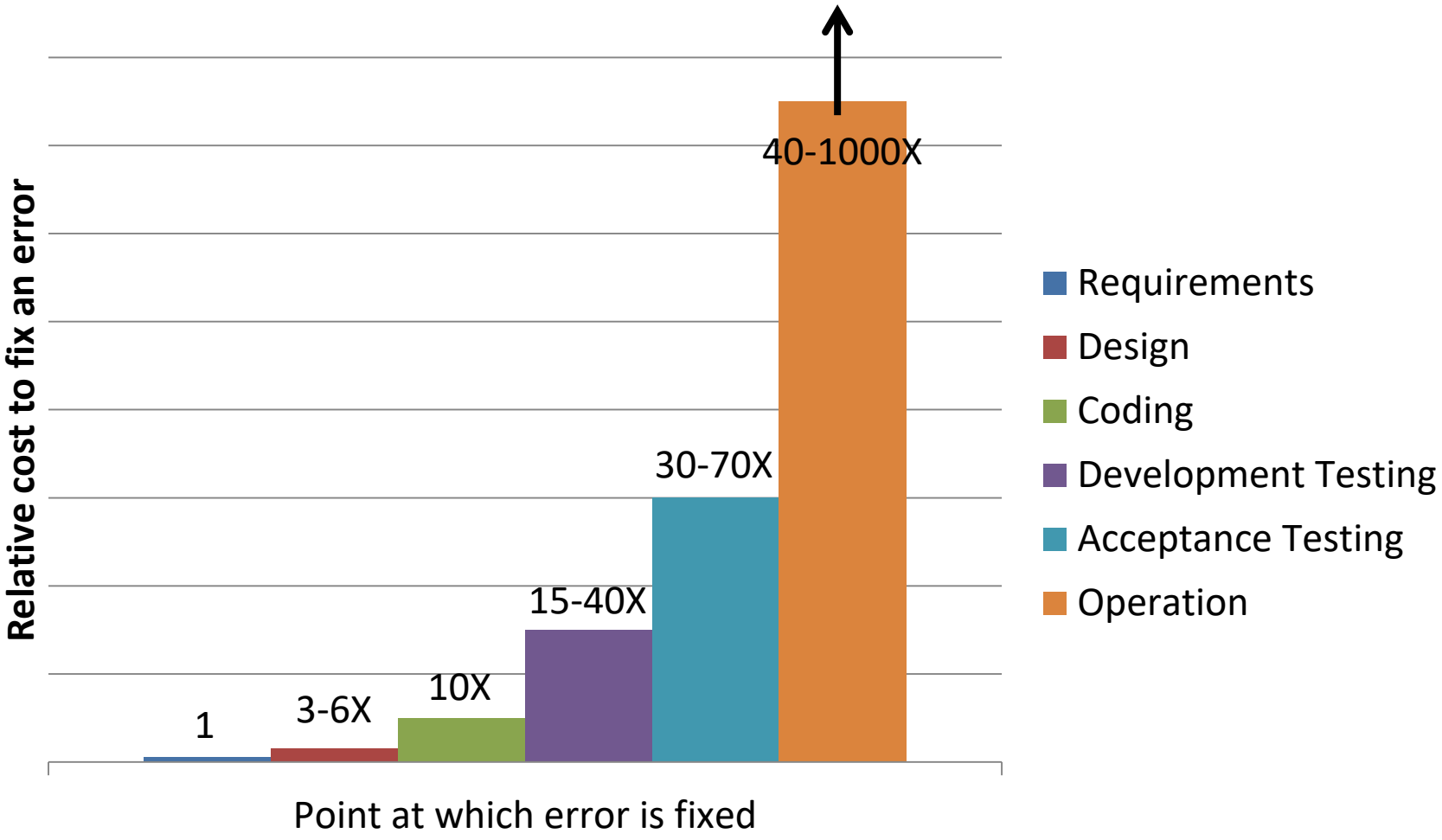
- Today we begin our look at inspection as a quality assurance technique
  - Statistical Process Control
  - What is inspection?
  - Informal vs. formal inspection
  - Inspection in the software process
  - Inspection roles
  - Effectiveness of inspections vs. testing

First Law of Software Development

- Earlier is Cheaper
  - The later in the development cycle a fault is detected, the more expensive it is to fix
    - Methods that find faults earlier deliver more bang for the buck



Cost of Fixing Errors



Reviews, Walkthroughs, and Inspections

- Terminology
  - Unfortunately, there is no good agreement on precise definitions for these terms, but...
- Reviews
  - …are the management practice of meetings to informally consider state of the project at certain stages, to gain confidence in project direction
    - e.g., preliminary design review, critical design review
  - Used to provide confidence that the design is sound
  - Often attended by management and customers

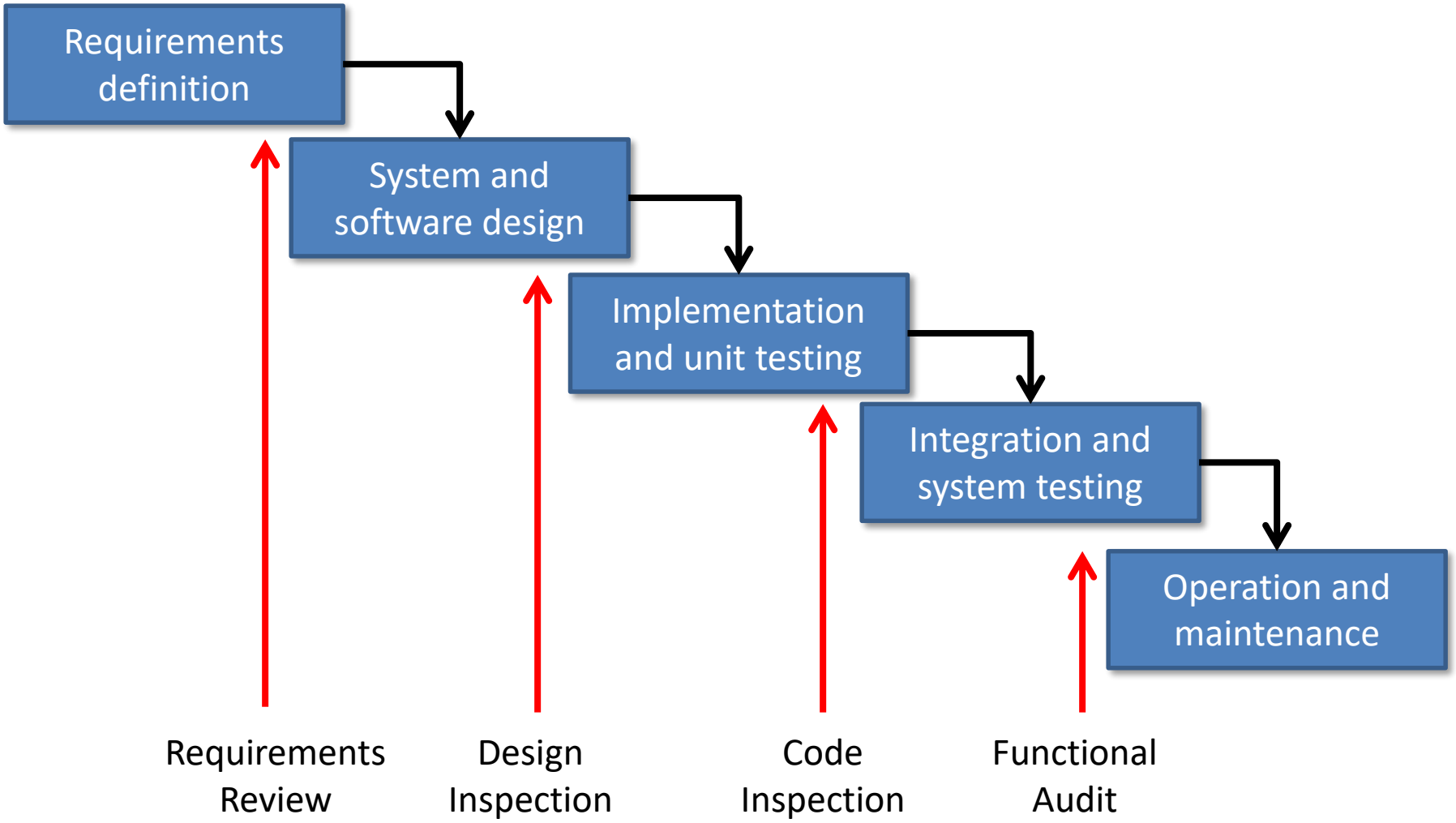
Reviews, Walkthroughs, and Inspections

- Walkthrough
  - …refers to an informal technical review, normally carried out by developers
  - Used by development teams to improve product quality by involving whole team in quality assurance at each stage
  - Focus is on critical analysis of artifacts, in an attempt to find or predict defects

Reviews, Walkthroughs, and Inspections

- Inspection
  - …refers to a completely formal process of review, also known as formal technical reviews
  - A formal system used to identify and remove defects, and improve the overall quality of the development process
  - Involves: Formal written reports, defect data collection and analysis, required standards and measures
    - Emphasis on documenting process progress and defects
  - First introduced by Fagan (IBM) about 1976, now required by some customers (e.g., U.S. military)

Inspections in the Software Process



Inspection

- IEEE Definition of Inspection
  - "... a formal evaluation technique in which software requirements, design, or code are examined in detail by a person or group other than the author to detect faults, violations of development standards, and other problems..."
- IEEE Objective of Inspection
  - "... to detect and identify software element defects. This is a rigorous, formal peer examination..."

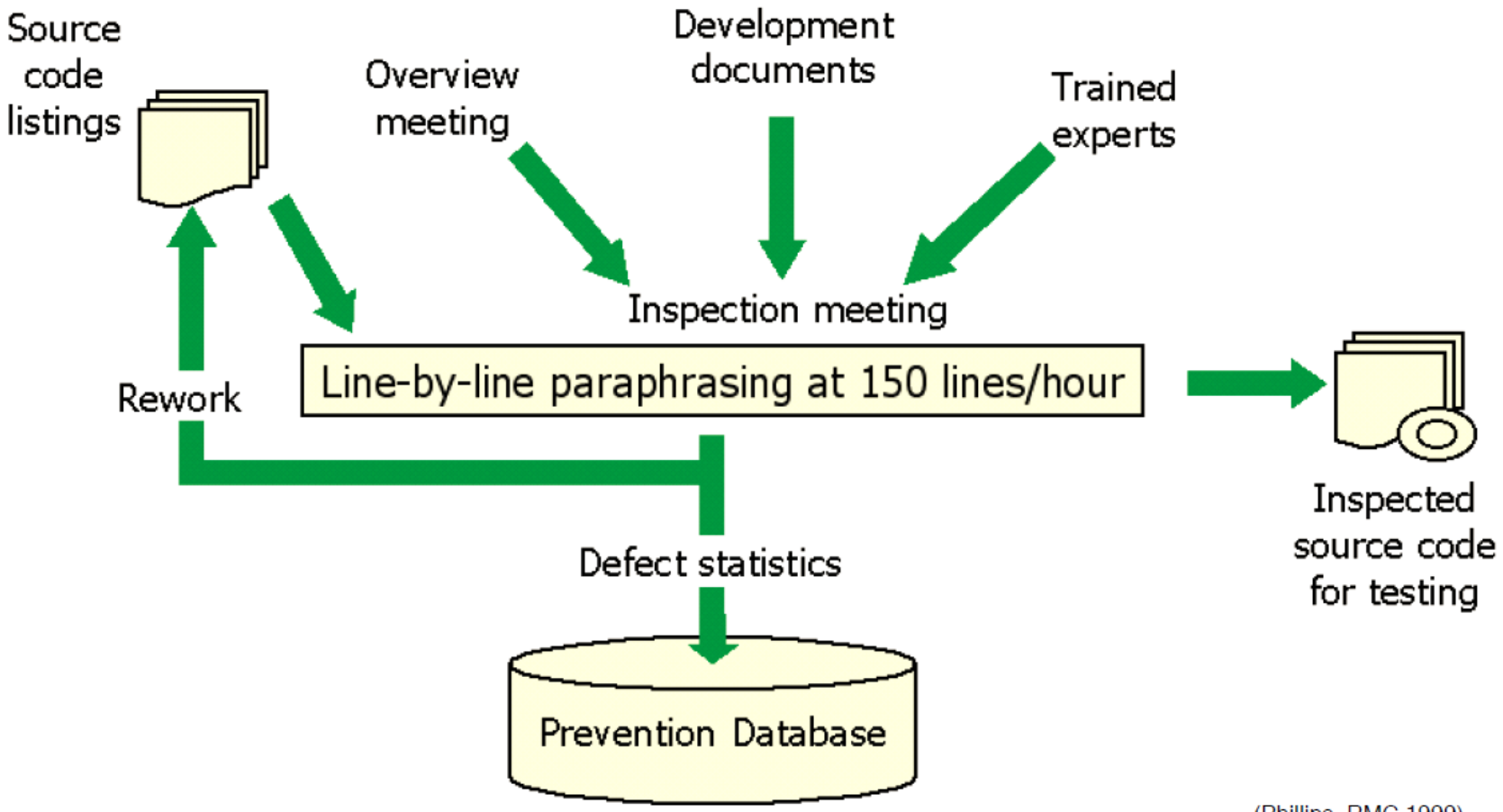
Inspection

- Verifies that the software elements satisfy its specifications
- Verifies that the software elements conform to applicable standards
- Identifies deviations from standards and specifications
- Collects software engineering data (for example, defect and effort data)
- Does not examine alternatives or stylistic issues

Inspection

- But Inspection (capital i) is a formal process!
  - One study found that 84% of surveyed organizations performed reviews or inspections, but 0% performed inspections entirely correctly
  - Even a walkthrough or a poorly done Inspection can be effective at improving software quality
  - Inspection is not only about defect correction, but also importantly about defect prevention

Fagan Inspections (e.g., for Code)



(Phillips, RMC 1999)

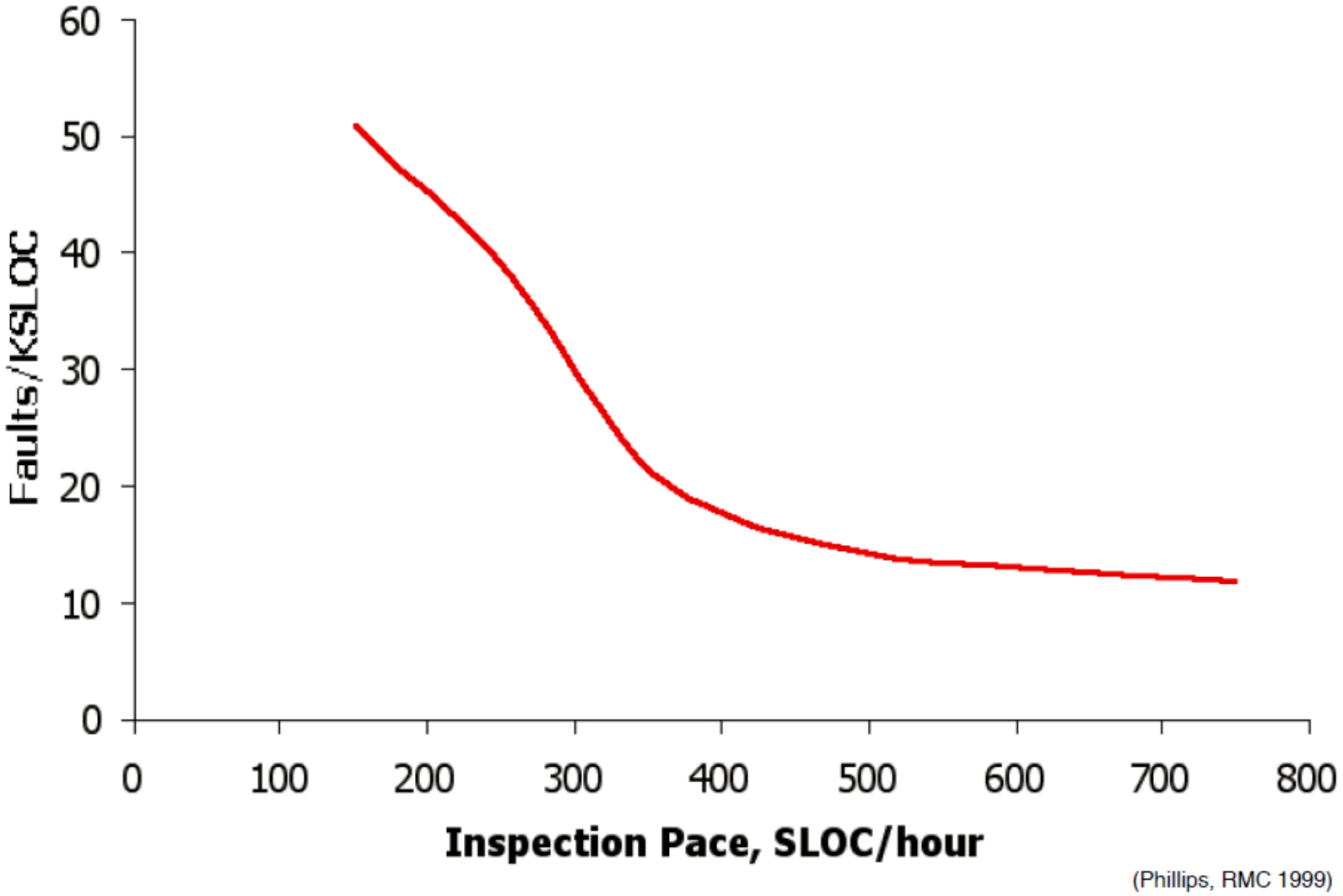
Inspection Roles (Fagan, Code Inspection)

- Moderator
  - Chairs the meeting, records faults found
  - Helps others stick to paraphrasing code, at the right pace
  - Keeps proceedings objective, professional, friendly
- Inspectors (2 or 3)
  - Knowledgeable peers who paraphrase the code, line by line
  - Must have attended overview meeting, reviewed requirements and design documents, must understand context of code
- Author
  - Silent observer who assists or clarifies only when asked

Choosing Inspectors (Fagan)

- Good Choices
  - Review specialists (e.g., QA analysts)
  - Technical people from the same team as author
  - Technical people with special expertise in subject matter of code
  - People with a special interest in the product
  - People from other parts of the org. or outside it
- Bad Choices (exclude!)
  - Managers, supervisors, or appraisers of the author
  - Anyone with a personality clash with the author or other reviewers
  - All management
  - Anyone with a conflict of interest

Inspection Efficiency

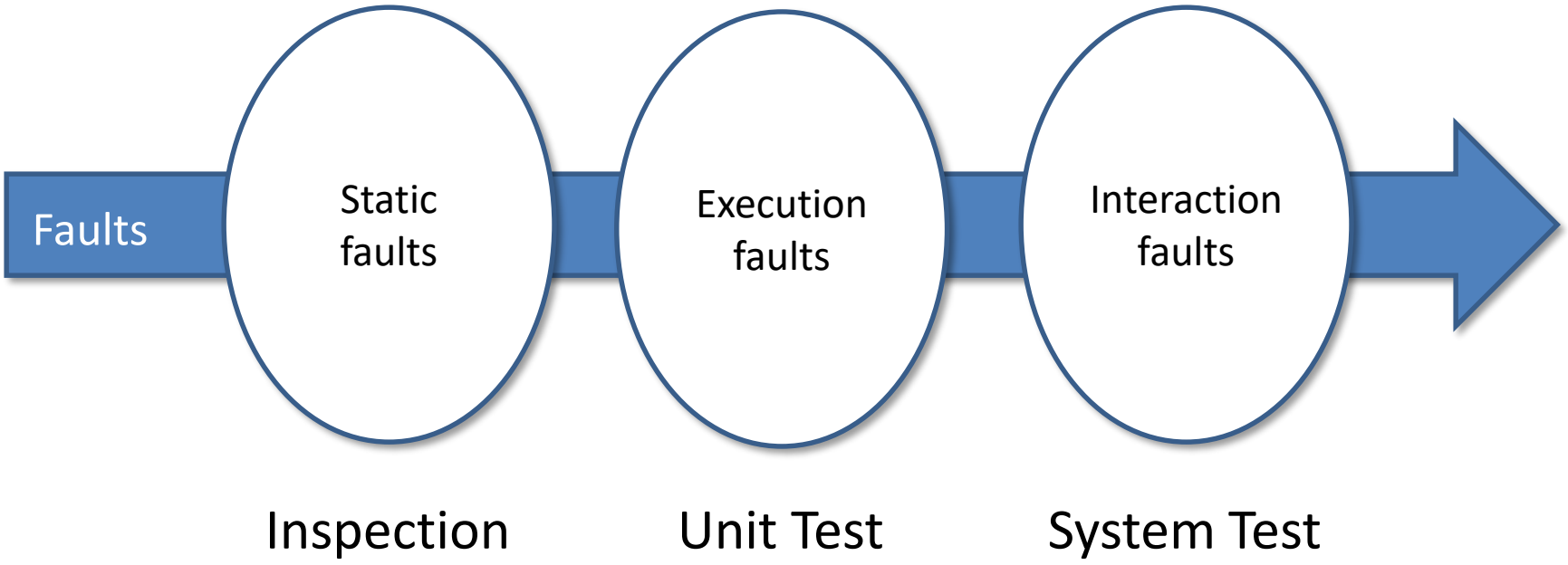


(Phillips, RMC 1999)

Side Benefits of Inspection

- Cultural
  - Team members gain a broader perspective on the software system as they review each other's work
  - Promotes a shared "quality culture", joint responsibility
- Organizational
  - Coding standards and practices are learned and enforced
  - Consistency improves
- Educational
  - Quality improves over time, as authors become more aware of the kinds of faults they are prone to make

Inspection in Context

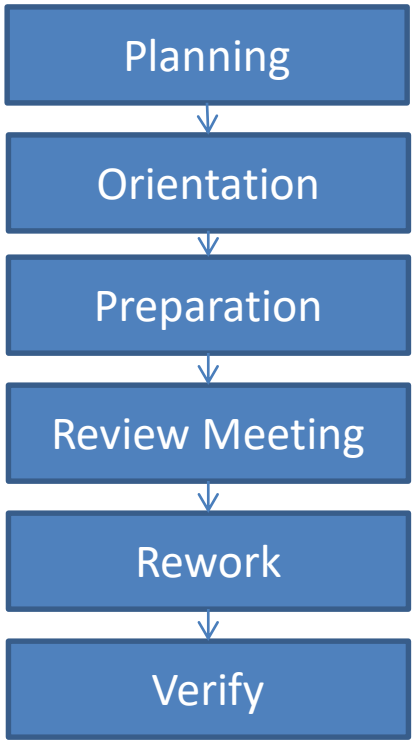


Inspection At Any Stage

- Inspections may be used at any stage of software development
  - Requirements, design, coding, testing, acceptance
- Ideally, inspections can be applied at every stage, to catch problems as early as they appear
- No matter what stage inspection is applied to, the inspection process is roughly the same

A Generic Inspection Process

- The basic process of formal inspection is always the same, no matter the artifact being inspected



Choose team, materials, schedule for inspection

Introduce artifact, process, goals to learn

Individually check artifact, note issues

Meet to discuss and consolidate issues

Correct defects noted

Verify artifact and process quality

Planning

- Objectives
  - Gather review package: artifact being inspected, references for it, checklists of inspection criteria, data sheets to record
  - Form inspection team
  - Set schedule

Planning

- Procedure
  - Moderator assembles team and review package
  - Moderator customizes checklist to artifact
  - Moderator plans schedule
  - Moderator checks artifact is ready for review
  - Moderator helps Author prepare overview of artifact

Orientation Meeting

- Objectives
  - Author provides overview of artifact
  - Inspectors obtain review package
  - Preparation goals set
  - Inspectors commit to participating

Orientation Meeting

- Procedure
  - Moderator distributes review package
  - Author presents overview
  - Moderator outlines preparation procedure

Preparation

- Objectives
  - Find the maximum number of non-minor defects in the artifact

Preparation

- Procedure (for Inspectors only)
  - Allocate scheduled time
  - Do detailed individual inspection of the artifact
  - Use checklists as a guide to focus on potential issues
  - Use references for calibration of what is expected or needed
  - Note critical, severe, and moderate level defects on reviewer report form
  - Note minor defects and questions for author clarification on artifact document

Example Defect Classification

- Critical
  - Defects that will cause the system to hang, crash, or produce incorrect results or behaviour, with no known workarounds
- Severe
  - Defects that will cause incorrect results or behaviour, but have known workarounds
- Moderate
  - Defects that affect limited areas of functionality that can either be worked around or ignored
- Minor
  - Defects that can be overlooked without loss of functionality

Example Checklists and References

- Checklists
  - Checklists often include questions concerning completeness, style, adherence to company standards, etc.
  - Code inspection checklists often include detailed questions about use of language features (e.g., no gotos), naming of variables, methods and classes, depth of nesting, etc.

Example Checklists and References

- References
  - May include:
  - Company standards documents
  - High quality examples of artifacts similar to the one being inspected
  - Chapters of reference textbooks on quality practice for artifacts
  - Online resources on quality practice for artifacts

Review Meeting

- Objectives
  - Make consolidated, comprehensive list of non-minor defects to be addressed
  - Help provide group synergy
  - Help provide shared knowledge of artifacts

Review Meeting

- Procedure
  - Moderator requests defects sequentially, in order of importance
  - Inspectors point out defects found, compare notes
  - Moderator (or note taker) writes down consolidated list of defects found and summarizes results of meeting in review summary defect report

Rework

- Objectives
  - Assess each defect listed in the review defect report, determine if really a defect, and repair as necessary
  - Written report on handling of each non-minor defect
  - Resolve minor issues as necessary and appropriate

Rework

- Procedure (for Author)
  - Author gets review defect summary report as well as marked-up copies of inspected artifact with details
  - Author assesses each defect, categorizes root cause and notes actions taken in an author action report
  - When finished, Author provides author action report and reworked artifact to Moderator for verification

Verify

- Objectives
  - Assess reworked artifact quality
  - Assess inspection process
  - Pass or fail the artifact

Verify

- Procedure (for Moderator)
  - Obtain reworked artifact and author action report
  - Review reworked artifact and action report for remaining problems
  - Provide recommendation for artifact (pass / fail)
  - With inspectors, sign off on artifact
  - Compute summary statistics for inspection and archive review documents in quality database
  - Generate process improvement proposals (if any)

Code Inspection Practices

- When the artifact is the actual code, we can use:
  - Checklists
  - Paraphrasing

Code Checklists

- Code checklists give a concrete list of properties of the code to check for
- Checklists may be general properties for any program, or specific properties for the specific program or kind of program
- Both desired properties (ones we want the code to have) and undesired properties (ones the code should not have) may appear in the list

Code Checklists

- Checklist items can range from simple surface properties such as code format to deep semantic properties such as termination
- The idea is that the inspector should look through the code to check for the presence or absence of each individual property, and check it off the list
- Checklists are only part of the inspection - the correctness of the code must also be checked

An Example: Generic Java Code Inspection Checklist

- 1. Variable and Constant Declaration Defects
  - 1.1 Are descriptive variable and constant names used in accord with naming conventions?
  - 1.2 Are there variables with confusingly similar names?
  - 1.3 Is every variable properly initialized?
  - 1.4 Can any non-local variables be made local?
  - 1.5 Are there literal constants that should be named constants?
  - 1.6 Are there variables that should be constants?
- 2. Method Definition Defects
  - 2.1 Are descriptive method names used in accord with naming conventions?
  - 2.2 Is every parameter value checked before being used?
  - 2.3 Does every method return a correct value at every return point?

An Example: Generic Java Code Inspection Checklist

...

– 4. Computation Defects
  - 4.1 Is underflow or overflow possible in any computation?
  - 4.2 Does any expression depend on order of evaluation of operators? Are parentheses used to avoid ambiguity?

...

– 6. Control Flow Defects
  - 6.1 Will all loops terminate in all cases?

...

Code Checklists

- Google Python Style Rules (partial)
  - Semicolons: Do not terminate your lines with semicolons and do not use semicolons to put two commands on the same line.
  - Line length: Maximum line length is 80 characters.
  - Parentheses: Use parentheses sparingly.
  - Indentation: Indent your code blocks with 4 spaces.
  - Blank Lines: Two blank lines between top-level definitions, one blank line between method definitions.
  - Strings: Use the % operator for formatting strings, even when the parameters are all strings. Use your best judgement to decide between + and % though.

Code Paraphrasing

- Reading the Code in English
  - Code paraphrasing is the original method of review described by Fagan for use in code inspections
  - Consists of reading the lines of code for their meaning in the problem domain, not in the programming language
  - The object is to ensure that the code really does implement what we want to have done

Code Paraphrasing

- Reading the Code in English
  - Paraphrasing should avoid mentioning variables or control flow, rather it should be phrased in terms of the abstract meaning of the concepts and processes being implemented by them
  - Discussion is seeded by scenarios, potential situations that may have to be handled
  - Paraphrasing is often coupled with checklists - the checklist addresses low level properties of the code itself, while the paraphrasing addresses its high-level meaning