Code Review

Arnav Gupta

April 22, 2024

Contents

1 Inspection and Reviews

Verification and Validation (V&V) can be static (analysis) or dynamic (executing code).

Software inspections & reviews: use system knowledge, its domain, and technologies used to discover problems

Advantages over dynamic V&V are:

- cascading errors can obfuscate test results (later errors may be new or come from prior error)
- incomplete versions can be inspected (tests require executable system, inspections don't)
- good inspections are more than bug hunts (uncover inefficiencies and style issues)

Review process: a process or meeting during which a work product, or set of work products, is presented to project personell, managers, users, customers, or other interested parties for comment or approval

Purposes:

- identify defects and new risks
- informally exchange knowledge
- collect data to learn from mistakes

Human examination of a work product is static, white-box. Applies to requirements, design, source code, test cases, etc. Types are inspection (structured and formal), walkthrough (less formal), buddy check (informal), and web-based reviews. Essential activity for quality.

Review reports are to rate the severity of a defect. Possibly determine statistics about findings and invested resources, quality/efficiency metrics. For formal inspection/reviews, reports may be required, like an inspection/FDR meeting notice report to launch review process.

Checklists are the most important tool for reviews. Generic checklists are for reviewing requirements, design, generic code, specific language code, and generic documents. Organizations develop specific checklists for particular objectives, past experience, etc. They should be maintained, improved, developed, and updated.

1.1 Inspections

Inspections are a strict, very formal form of review, to obtain defects, collect data, and communicate development documents. Done in a team of 3 to 6 people. Roles are:

- **moderator**
  - crucial role
  - ensures inspection procedures are followed
  - verifies work product's readiness for inspection
  - verifies entry criteria are met
  - assembles an effective inspection team
  - keeps the inspection meeting on track

  - verifies that exit criteria are met
  - comes from outside the project team (but still a peer)
- **recorder** (scribe)
  - documents all defects that arise from the inspection meeting in an inspection defect list
  - not just a procedural task, requires technical knowledge
- **reviewer**
  - analyzes and detects defects in the work product
  - all participants play that role
  - depending on reviewed document, consider a representative each from requirements (or user), test, design, and implementation teams
- **reader** (presenter)
  - leads inspection team through inspection meeting by reading aloud small logical units, paraphrasing where appropriate
- **producer** (author)
  - work product author
  - responsible for correcting any found defects

Inspection process is:

- **planning** (done by moderator)
  - identify work product
  - determine whether product inspected meets entry criteria
  - select team
  - assign roles
  - prepare and distribute the inspection forms and materials
  - set inspection schedule
  - determine whether to hold an overview
- **overview** (done by moderator and reviewers)

  - optional phase where team members who are unfamiliar with the work product to be inspected receive orientation

- **preparation** (done by reviewers)
  - key stage
  - members of inspection team inspect work individually looking for defects in the work product
  - most defects found during this step, not during inspection meeting
- **inspection meeting** (done by moderator, reviewers, recorder, presenter, producer)
  - inspection team members meet to find, categorize, and record possible defects in the work product
  - no resolution of defects attempted, but action items assigned
  - beware of team size and long, detailed presentations
- **third hour**
  - optional additional time apart from inspection meeting that can be used for discussion, possible solutions, or closure of open issues raised during the inspection meeting
  - if actual review task needs >2 hours because of complexity, schedule another review session (no more than 2 a day)
- **rework** (done by producer)
  - work product is revised by the author to correct defects found during inspection
- **follow-up** (done by moderator and producer)
  - meeting to determine if defects found during inspection meeting have been corrected and to ensure that no additional defects have been introduced
  - final inspection data is collected and summarized, and the inspection is officially closed

Coverage of 5 to 15% of document pages represents a significant contribution to quality, which depends on choice of review target.

Recommended for inclusion:

- sections with complicated logic
- critical sections (defects severely damage essential system capability)
- sections dealing with new environments

Recommended for omission:

- straightforward sections
- sections similar to already reviewed ones
- sections not expected to affect functionality if faulty
- reused sections

Inspections help with learning from each other, foster team spirit, help testing team identify problem areas and test cases to focus on, and give management an idea of whether the project is on track.

Inspection Guidelines:

1. review product, not producer
2. set agenda and maintain it
3. limit debate and rebuttal (90 to 120 min for meeting)
4. identify problem areas, but problem solving should be after review meeting
5. take written notes
6. limit number of participants and insist upon preparation
7. develop checklist for each product that is likely to be reviewed
8. allocate resources and schedule time for reviews
9. conduct meaningful training for all reviewers
10. prepare report and establish follow-up procedure
11. review the review process

1.2 Walkthrough

To find defects, become familiar with development documents.

Done with teams of 2 to 7 people, where material must be distributed in advance, but only presenter must prepare.

Each participant lists potential defects and points that need further explanation. Can be directed by anyone.

Changes only suggested, further investigation and fixes are not in the scope of a walkthrough.

Differences in inspections and walkthrough is different roles, less preparation, and no formal follow-up.

Buddy check is a code walkthrough by 1+ reviewers, they read code and report errors back to the developer.

Pair programming includes a form of buddy check.

1.3 Modern Reviews

Traditional: mandated reviewer checklists and in-person meetings

Modern: lightweight, tool-supported, flexible, and asynchronous

Modern is used in open source systems.

Modern review roles are:

- **author**
  - responsible for correcting problems that are identified during the review
- **reviewer**
  - analyzes and detects problems in the artifacts
  - engineer with expertise in the context the artifact operates within
  - often 2+ reviewers must agree with an artifcat before it is valid

Many more artifacts reviewed than traditional.

2 Metrics

Quality metric:

- quantitative measure of the degree to which an item possesses a given quality attribute

- a function whose inputs are software data and output is a single numerical value that can be interpreted as the degree to which the software possesses a given quality attribute

Goal is to keep track of project-level or organizational-level metrics, not an individual person's performance.

Measure: quantitative indication of the extent, amount, dimension, capacity, or size of some attribute of a product, project, or process (single data point)

Measurement: act of determining a measure

Metric: quantitative measure of the degree to which a system, component, project, or process possesses a given attribute, may relate individual measures

Indicator: metric that provides insight into software process, project, or product (compared against a standard or threshold)

2.1 Objectives

Assist management in:

- control of software development projects and software maintenance
- support of decision making
- initiation of preventative or corrective action

Further objectives are:

1. facilitate managerial control, planning, and intervention based on calculation of metrics regarding deviations of actual from planned for functional (quality) performance and timetable/budget performance
2. identify situations for development or maintenance process improvement (preventative or corrective actions) based on accumulation of metrics information regarding performance of teams, units, etc

Software size measures are used, like:

- **KLOC**: thousands of lines of code
  - problematic since language and programming style dependent and difficult to predict before code is written

- **function points**: measure of development resources (human resources) required to develop a program, based on functionality specified for the software system
  - pre-project estimates of size are stated in terms of required development resources
    - assesment based on requirements document
    - refined during analysis phase
    - not dependent on development tools or programming languages
  - disadvantages
    - detailed requirements may not be available early on
    - subjective results depend on experience
    - domain-dependent, so cannot be universally applied

2.2 Types of Metrics

Process metrics: to improve software process, refer to development, such as defect metrics, effectiveness of defect removal, and productivity

- software process quality metrics, like error density metrics and error severity metrics
- software process timetable metrics
- error removal effectiveness metrics
- software process productivity metrics

Product metrics: characteristics of product, refer to operational phase, such as failure metrics and help desk/maintenance services

Project metrics: project characteristics and execution, such as staffing pattern over life cycle and productivity

Error counted measures are the number of code errors detected by code inspections and testing (NCE) vs the weighted number of code errors detected by code inspections and testing (WCE), where the higher the metrics, the lower the quality. Weights assigned based on severity.

Similarly other measures are number of development (design and code) errors detected during development (NDE) and WDE is the weighted version.

Error density metrics are:

- CED (code error density) where $CED = NCE/KLOC$
- DED (development error density) where $DED = NDE/KLOC$
- WCED (weighted code error density) where $WCED = WCE/KLOC$
- WDED (weighted development error density) where $WDED = WDE/KLOC$
- WCEF (weighted code errors per function point) where $WCEF = WCE/NFP$
- WDEF (weighted development errors per function point) where $WDEF = WDE/NFP$

Error severity metrics are:

- ASCE (average severity of code errors) where $ASCE = WCE/NCE$
- ASDE (average severity of development errors) where $ASDE = WDE/NDE$

Error severity metrics are used when error density metrics are decreasing, to detect adverse situations of increasing numbers of severe errors. More severe errors means lower quality.

For timetables, metrics are milestones completed on time (MSOT), MS (total number of milestones), and TCDAM (total completion delays for all milestones) with delays in time unit of choice and milestones completed before the scheduled date counted as either 0 or minus delays, as long as they are done consistently.

Software process timetable metrics are:

- TTO (time table observance) where $TTO = MSOT/MS$
- ADMC (average delay of milestone completion) where $ADMC = TCDAM/MS$

Higher TTO means greater quality, higher ADMC means lower quality.

For error removal, metrics used are number of software failures detected during a year of maintenance service or any defined period (NYF) and the weighted equivalent (WYF).

Error removal effectiveness metrics are:

- DERE (development errors removal effectiveness) where $DERE = NDE/(NDE + NYF)$
- DWERE (development weighted errors removal effectiveness) where $DWERE = WDE/(WDE + WYF)$

Higher metrics means greater quality.

For software process productivity, metrics used are total working hours invested in development of software system (DevH), number of thousands of reused LOC (ReKLOC), number of reused pages of documentation (ReDoc), and number of pages of documentation (NDoc).

Software process productivity metrics are:

- DevP (development productivity) where $DevP = DevH/KLOC$
- FDevP (function point development productivity) where $FDevP = DevH/NFP$
- CRe (code reuse) where $CRe = ReKLOC/KLOC$
- DocRe (documentation reuse) where $DocRe = ReDoc/NDoc$

Higher DevP or FDevP means lower quality, higher CRe or DocRe means higher quality.

To implement software quality metrics:

1. define an attribute to be measured
2. define the metrics that measure the attribute
3. determine comparative target values (indicators)
4. define method of reporting and metrics data collection

Limitations of software metrics are:

- universal obstacles
  - enough resources allocated for developing and collecting metrics
  - opposition from employees
  - uncertainty regarding data validity, from partial and biased reporting
- software obstacles

- low validity and limited comprehensiveness of metrics
- KLOC is not a very good estimate for development time
- defects detected depend of thoroughness of review/testing and reporting style