Static Analysis

Angelo Gargantini

Testing e verifica del software

AA 2018-19

Software Inspection

- Examine representation of a software system with the aim of discovering anomalies and defects
  - "Check software artifacts for constructs that are known to be problematic from past experience"
- Systematic & detailed review technique
  - Peer review (not author or his boss but inspection team)
- Applicable to all kinds of software artifacts
  - Requirement specification, design documents, source code, ...
- Defined in the 70's by Fagan (IBM)
  - Several alternatives & extensions proposed that vary in rigorousness of the review and focus on particular goals
  - Pair programming can be seen as a light & informal instance

Software Inspection

- Can be applied to essentially any document
  - requirements statements
  - architectural and design documents
  - test plans and test cases
  - source code
- May also have secondary benefits
  - spreading good practices, shared standards of quality
  - pair-programming
- Limitations
  - takes a considerable amount of time
  - re-inspecting a changed component can be expensive
- Used primarily in areas
  - where other techniques are inapplicable or ineffective
  - where other techniques do not provide sufficient coverage

Formal code reviews

- A formal code review is the process under which **inspection** is performed.
- Can be a simple one-on-one meeting or a detailed rigorous code inspection.
- May be organized by the programming or the testing team.

Why have Code Inspections

- Several eyes are better than one pair
- Not all problems are detected by automated tools
- Tools cannot decide whether the problem is real.
- Tools cannot decide whether the problem is serious. [Worth fixing.]

Code Inspection

- Code Inspection is the most formal type of review, which is a kind of static abalysis to avoid the defect multiplication at a later stage.
- The main purpose of code inspection is to find defects and it can also spot any process improvement if any.
- An inspection report lists the findings, which include metrics that can be used to aid improvements to the process as well as correcting defects in the document under review.
- Preparation before the meeting is essential, which includes reading of any source documents to ensure consistency.
- Inspections are often led by a trained moderator, who is not the author of the code.

Code Inspection

- The inspection process is the most formal type of review based on rules and **checklists** and makes use of entry and exit criteria.
- It usually involves peer examination of the code and each one has a defined set of roles.
- After the meeting, a formal follow-up process is used to ensure that corrective action is completed in a timely manner.

Code review checklist

- Design and Architecture errors
- Computation errors
- Comparison errors
- Control flow errors
- Subroutine parameter errors
- Input/Output errors
- Memory allocation errors
- Error discovered from previous code reviews
- Other checks
  - Does your code pass the lint test? E.g., How about gcc compiler warnings?
  - Is your code portable to other OS platforms?
  - Does the code handle ASCII and Unicode?

Inspections vs. Testing

- What attributes are well-handled by inspections but not testing?
- “Fuzzy” non-functional properties
  - Maintainability, evolvability, reusability
- Other properties tough to test
  - Scalability, efficiency Security, integrity Robustness, reliability, exception handling Time sensitive, real-time actions
- Requirements, architecture, design documents
  - Cannot “execute” these as a test

Inspection team

Inspections are characterized by roles, process, and reading techniques, i.e., who the inspectors are, how they organize their work and synchronize their activities, and how they examine the inspected artifacts. Inspection is not a full-time job: Many studies indicate that inspectors' productivity drops dramatically after two hours of work, and suggests no more than two inspection sessions per day. Thus, inspectors are usually borrowed from other roles: junior and senior software and test engineers, project and quality managers, software analysts, software architects, and technical writers. The same studies highlight the delicate relation between inspectors and developers: The efficacy of inspection can vanish if developers feel they are being evaluated. In classic approaches to inspection, managers and senior engineers who participate in inspection sessions are often borrowed from other projects to avoid misinterpreting the goals of inspection.

Checklists

Checklists are a core element of classic inspection. They summarize the experience accumulated in previous projects, and drive the review sessions. A checklist contains a set of questions that help identify defects in the inspected artifact, and verify that the artifact complies with company standards. A good checklist should be updated regularly to remove obsolete elements and to add new checks suggested by the experience accumulated in new projects. We can, for example, remove some simple checks about coding standards after introducing automatic analyzers that enforce the standards, or we can add specific semantic checks to avoid faults that caused problems in recent projects.