8.2

1. Search for a tool that provides static code analysis for that language. FindBugs, it is a static code analysis tool for java based program

2. Read the specification for this tool and write a brief document with the type of anomalies that it detects.

FindBugs uses static analysis to inspect Java bytecode for occurrences of bug patterns.  Static analysis means finding bugs by simply inspecting a program's code without executing the program. FindBugs works by analyzing Java bytecode (compiled class files), so developer don't even need the program's source code to use it. It is based on the concept of bug pattern: Difficult language features; Misunderstood API methods; misunderstood invariants when code is modified during maintenance; typos, use of the wrong Boolean operators. It mainly detects the Data faults, control faults and input/output faults, such as array out of bounds, unused variables for data faults, and dead code for control faults. The multiple output variables is detected as bad practice code warning in the tool, bad practice means violation of recommended and essential coding practice. The parameter type and mismatch are detected by java compiler during compilation. For storage management faults, it probably would detect the null pointer that is part of Pointer arithmetic error.

8.3

Read the document with the notes on Verification and Validation (vandv\_notes)

1. What is the difference between validation and verification?

Validation refers to make sure a product complies with its requirements. Verification refers to checking that the product functionality executes correctly.

Validation is to check software function as user expect and those expectation (requirements) are implemented, verification is often an internal process of checking the compilation with specification and requirements.

1. What are the two objectives of the Validation and Verification stage of Quality Assurance?

Detection of defects and assess if the system is ready to be used.

The discovery of defects in a system. The assessment of whether or not the system is usable in an operational situation. It build the confidence and good enough for use.

1. What are the main aspects that affect the level of confidence of the Verification and validation process?—page 3

The user expectations, how critical is the software for the company, and how soon the product must be in the market.

Depends on system’s purpose, user expectations and marketing environment.

1. The tools for automatic software verification are not only used for Dynamic Verification (software testing), but also can be used for supporting Static verification, such as software inspections.
2. What is the difference between static and dynamic verification techniques

*Software inspections are static verifications, and software tests are dynamic verifications.*

1. Consider a product divided into the following product parts: Requirements, High-level design, Formal specification, Detailed design, and program (the code). Which of these parts are not considered for static verification? Those items can be static verified.
2. If a program has a very large number of tests, and all of them passed, we may conclude that:

The tests did not discover any errors. However, it may not comply with software or even functional requirements in the software requirements, since it may not contains the test cases consistent with those requirements. In addition, we cannot always say test cases are adequacy for covering all scenarios, they are limited by test cases design.

1. Software inspections are typically applied to any representation of the system.—page 4

It includes implementation details and design documents as well as other component in the system.

1. Which of these documents is needed for a code inspection?

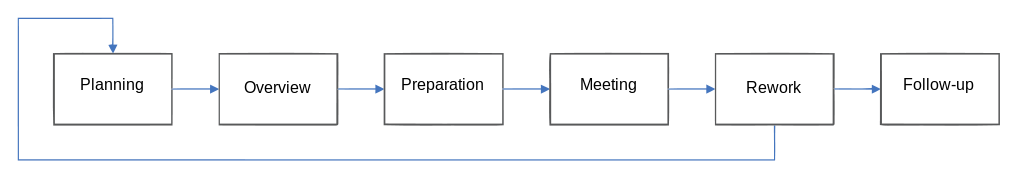
The code; Description of the system; Error checklist

1. Automatic Static Analysis are tools that provide information about errors that can be deduced by parsing the code.

Function execute incorrectly can be detected by software testing activity, lack of paper documentation in a function can be detected by review process, but it is often treated as not good engineering practice, performance problem in a function could be also detected by software performance testing, not always be detected by static analysis.

8.4, 8.5

1. Search for information describing the Fagan Code Inspection procedure.

**

A. Planning: Preparation of materials; Arranging of participants; Arranging of meeting place; Goal of this inspection

B. Over-View: a moderator should state any special scrutiny those areas that were reworked since the errors found in design inspections or any changes made in the design, prepare the code listings and design specifications.

C. Preparation: each inspection team member studies the ranked distributions of errors types found by recent inspections, which could make them concentrate on the most the error-prone areas, and also the clues of finding this errors should also be studied.

D. Inspection: An implementer should describe the implementation of each piece of logic, and all high-level documentation and logic specifications should be presented as well. In this activity, the objective is to find errors, the moderator should note the classification and severity. The activity should take place no longer than 2 hours on average.

E. Rework: the step in software inspection in which the defects found during the inspection meeting are resolved by the author, designer or programmer. On the basis of the list of defects the low-level document is corrected until the requirements in the high-level document are met. (Designers or coders resolve noted issues/errors)

F. Follow-Up: an inspection report should be produced and pass to each team members. The report is to ensure the all raised issues will be addressed in the reworks.

Roles in the inspection:

Author/Designer/Coder: the person who wrote the low-level document

Reader: paraphrases the document

Reviewers: reviews the document from a testing standpoint

Moderator: responsible for the inspection session, functions as a coach

2. Document with three advantages and disadvantages of the Fagan inspection.

Advantages:

A. Errors are categorized and well documented. It allow members to maintain the cumulative totals of each type and modules of errors, easy to review the errors on later stage and could be referenced in the future.

B. The detailed feedback is on a relatively real-time basis.

C. Programmer could able to make some improvements on later work by receiving the feedback, such as avoiding those addressed errors during the development, reduce the cost of fixing the errors on later stage of the project.

D. High chance to identify the defects and analysis the errors.

E. Inspections could shorten delivery time by reducing the time of the integration and system test and debug processes (Cost improvement because of less resources needed in later phases)

F. Provide a good way to check the completeness of the project against exit criteria.

G. Help developers to learn coding skills from different people.

H.The standard method can guild people to get a solution, consider and avoid some mistakes.

I. A formal code inspection process that can find errors and resolve them effectively

J. Significantly decrease errors in final release

K. The formal stage of follow-up seems like it will ensure that no found defects are left unresolved.

L. The formal stage of overview and preparation seem like they will ensure the participants are fully equipped to review the code.

Disadvantages:

A. Since inspectors are from different teams, they might not have much knowledge of the system, they would need to spend time to learn the system.

B. It might not suitable for a relatively small project, since it costs resources on carrying out those inspections, such as effort on planning, bring overhead for those small projects.

C. may only suitable for critical work job, not for trivial work.

D. Some common errors would repeatedly being reviewed.

E. Extra effort tracking the inspected problems.

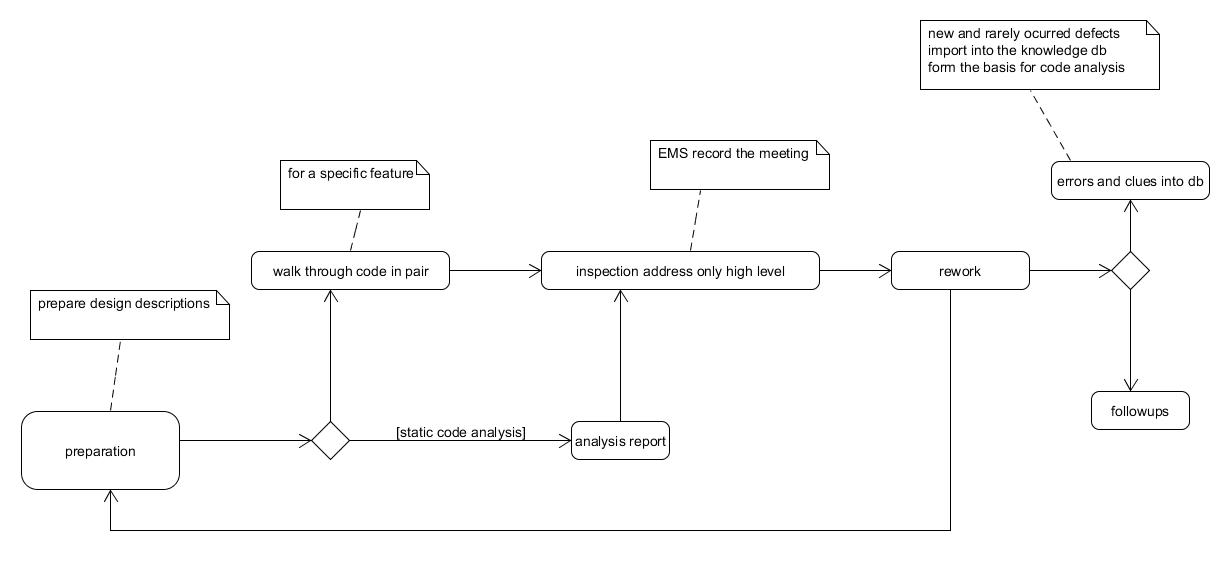
F. Difficult to start the Fagan's inspection process - have to initially resolve the problems with the input documents as the initial input material likely has many errors and defects

3. Solution:

Static code analysis tools (FindBugs, pc-lint, SonarQube) could increase the efficiency with a great support of automatic detecting error by parsing the code, allow inspector concentrate on rarely occurred and complex errors other than trivial errors.

Those tools have up-to-date error patterns in the knowledge database and can produce a well-categorized error detection report for supplement of inspections. They could learn new type of errors by import patterns into the database. Import the document including issues into tools after each inspection, developer resolve issues and update the information after the rework, doing so can save times in the planning with less efforts of preparing material.

4. Propose a re-design of the Fagan inspection that maintains the advantage and removes the disadvantage previously chosen. The team is free to adopt any new measure (even a total re-design of the process).

**

4. What is the difference between a “code inspection” and a “code walk-through”?

Code inspection: a formal planned review bases on regular rules/procedures and checklists, which intend to find defects and suggest any improvements, it usually need to product the well-structured reports and need follow-up, keep track of reworks.

Code walk-through: an informal review people go through the code line-by-line and may not have official guidelines/procedures, so results could vary widely. Some errors could also address while people explain the code. It is usually for the specific unit of the system and familiarize developer with code, not quite suitable for a feature.

8.6

Review the content of the Validation and Verification notes (vandv\_notes) with special attention to:

Static and Dynamic V&V

Program testing and debugging

V&V Planning: The software test plan

Inspections and automatic static analysis