Design the Review

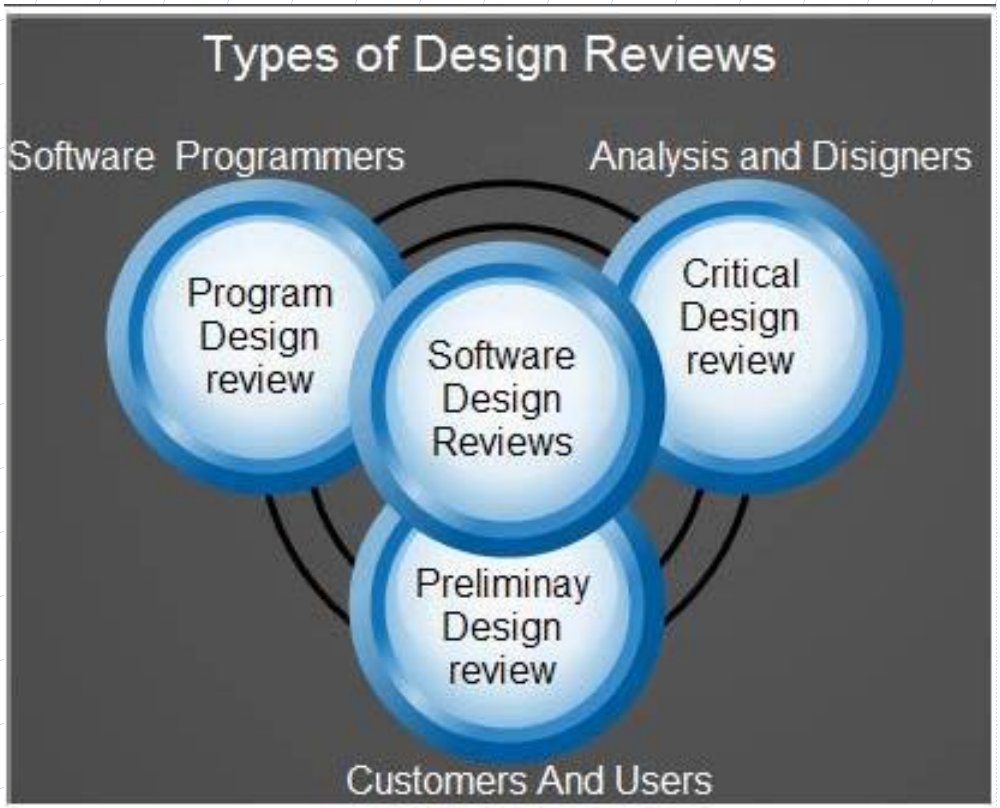
- It is a systematic, and well-documented inspection of design that aims to check whether the specified design requirements are adequate and the design meets all the specified requirements.
- **IEEE** defines software design review as 'a formal meeting at which a system's preliminary or detailed design is presented to the user, customer, or other interested parties for comment and approval.'
- These reviews are held at the end of the design phase to resolve issues related to software design decisions, that is, architectural design and detailed design of the entire software or a part of it.

Design Review Continue

- Elements that must be examined while design Review
  - Design Specifications
  - Verification and Validation Results
  - Testing and Development Plan
  - Project related documents

Types of Software Design Review

- Preliminary Design Review
- Critical Design Review
- Program Design Review



Purpose of Preliminary Design Review

- To ensure that the software requirements are reflected in the software architecture
- To specify whether effective modularity is achieved
- To define interfaces for modules and external system elements
- To ensure that the data structure is consistent with the information domain
- To ensure that maintainability has been considered
- To assess the quality factors.

Purpose of Critical Design Review

- To assure that there are no defects in the technical and conceptual designs
- To verify that the design being reviewed satisfies the design requirements established in the architectural design specifications
- To assess the functionality and maturity of the design critically
- To justify the design to the outsiders so that the technical design is more clear, effective and easy to understand

Purpose of Program Design Review

- To assure the feasibility of the detailed design
- To assure that the interface is consistent with the architectural design
- To specify whether the design is compatible to implementation language
- To ensure that structured programming constructs are used throughout
- To ensure that the implementation team is able to understand the proposed design.

Software Design Review Process

- Entry Criteria
- Activities
  - Select the members, assign them their roles, and prepare schedules for the review.
  - Distribute the software design review package.
  - Participants check the completeness and conformance of the design to the requirements in addition to the efficiency of the design. They also check the software for defects. The recorder documents the defects along with the suggested action items and recommendations.
  - The design team rectifies the defects (if any) in design and makes the required changes in the appropriate design review material.
  - The software development manager obtains the approval of the software design.

Evaluating Software Design Process

- Is the solution achieved with the developed design?
- Is the design reusable?
- Is the design well structured and easy to understand?
- Is the design compatible with other platforms?
- Is it easy to modify or enlarge the design?
- Is the design well documented?
- Does the design use suitable techniques in order to handle faults and prevent failures?
- Does the design reuse components from other projects, wherever necessary?

Version Control

- Version control, also known as source control, is the practice of tracking and managing changes to software code.
- Version control systems are software tools that help software teams manage changes to source code over time.
- As development environments have accelerated, version control systems help software teams work faster and smarter.
- They are especially useful for DevOps teams since they help them to reduce development time and increase successful deployments.

Why Version Control

- A version control system is a kind of software that helps the developer team to efficiently communicate and manage(track) all the changes that have been made to the source code along with the information like who made and what changes have been made.
- A separate branch is created for every contributor who made the changes and the changes aren't merged into the original source code unless all are analyzed as soon as the changes are green signaled they merged to the main source code.
- It not only keeps source code organized but also improves productivity by making the development process smooth.

Benefits of Version Control System

- Enhances the project development speed by providing efficient collaboration.
- Leverages the productivity, expedites product delivery, and skills of the employees through better communication and assistance.
- Reduce possibilities of errors and conflicts meanwhile project development through traceability to every small change.
- Employees or contributors of the project can contribute from anywhere irrespective of the different geographical locations through this **VCS.**

Benefits of Version Control System

- For each different contributor to the project, a different working copy is maintained and not merged to the main file unless the working copy is validated. The most popular example is **Git, Helix core, Microsoft TFS.**
- Helps in recovery in case of any disaster or contingent situation.
- Informs us about Who, What, When, Why changes have been made.

Types of Version Control

- **Local Version Control Systems:**
- It is one of the simplest forms and has a database that kept all the changes to files under revision control. RCS is one of the most common VCS tools.
- It keeps patch sets (differences between files) in a special format on disk. By adding up all the patches it can then re-create what any file looked like at any point in time.
- **Centralized Version Control Systems:**
- Centralized version control systems contain just one repository globally and every user need to commit for reflecting one's changes in the repository. It is possible for others to see your changes by updating.

Types of Version Control

- **Distributed Version Control Systems:**
- Distributed version control systems contain multiple repositories. Each user has their own repository and working copy.
- Just committing your changes will not give others access to your changes. This is because commit will reflect those changes in your local repository and you need to push them in order to make them visible on the central repository.
- Similarly, When you update, you do not get others' changes unless you have first pulled those changes into your repository.

Some well Known Version Control Tools

- GIT
- CVS
- SVN
- Mercurial
- Monotone
- Bazaar
- TFS
- VSTS
- Perforce Helex Core
- IBM Rational Clear Case
- Revision Control System
- Visual Surface Safe

What is Subversion

- SVN stands for Subversion. So, SVN and Subversion are the same. SVN is used to manage and track changes to code and assets across projects.
- Subversion is used for maintaining current and historical versions of projects. Subversion is an open source centralized Version Control System.
- It's licensed under Apache. It's also referred to as a software version and revisioning control system.

How does Subversion work

- For Subversion to work, the SVN setup needs two main elements:
- The **server,** which has all versions of all source files
- A **local copy of the files,** which is on your computer
- The files on your computer are called working files. These are the files in which each user makes edits. Then, users commit their changes to the SVN server.
- Each time a user commits a change, SVN manages and records it by creating a new version. Like most version control options, users typically work with the most recent version. But if an older version is needed, you can revert to an earlier version.

Refactoring

- Refactoring is the process of restructuring code, while not changing its original functionality. The goal of refactoring is to improve internal code by making many small changes without altering the code's external behavior.
- Computer programmers and software developers refactor code to improve the design, structure and implementation of software. Refactoring improves code readability and reduces complexities.

Refactoring

- Refactoring can also help to find hidden vulnerabilities in their software.
- The refactoring process features many small changes to a program's source.
- For example, is to improve the structure of source code at one point and then extend the same changes systematically to all applicable references throughout the program.
- The thought process is that all the small, behavior-preserving changes to a body of code have a cumulative effect.

Purpose of Refactoring

- More efficient by addressing dependencies and complexities.
- More maintainable or reusable by increasing efficiency and readability.
- Cleaner so it is easier to read and understand.
- Easier for software developers to find and fix bugs or vulnerabilities in the code.

When should Code be Refactored

- Refactoring can be performed after a product has been deployed, before adding updates and new features to existing code, or as a part of day-to-day programming.
- it is normally done before developers move on to the next project.
- In the software delivery lifecycle, where the developers have increased availability and more time to work on the source code changes needed.
- Once the new code is added, the developer can refactor the same code again to make it clearer.

Benefits of refactoring

- Makes the code easier to understand and read because the goal is to simplify code and reduce complexities.
- Improves maintainability and makes it easier to spot bugs or make further changes.
- Encourages a more in-depth understanding of code. Developers have to think further about how their code will mix with code already in the code base.
- Focus remains only on functionality. Not changing the code's original functionality ensures the original project does not lose scope.

Techniques of Refactoring

- **Red, green.** This widely used refactoring method in Agile development involves three steps. First, the developers determine what needs to be developed; second, they get their project to pass testing; and third, they refactor that code to make improvements.
- **Inline.** This technique focuses on simplifying code by eliminating unnecessary elements.
- **Moving features between objects.** This technique creates new classes, while moving functionality between new and old data classes.

Techniques of Refactoring

- **Extract.** This technique breaks down code into smaller pieces and then moves those pieces to a different method. Fragmented code is replaced with a call to the new method.
- **Refactoring by abstraction.** This technique reduces the amount of duplicate code. This is done when there is a large amount of code to be refactored.
- **Compose.** This technique streamlines code to reduce duplications using multiple refactoring methods, including extraction and inline.

Best practices of refactoring

- Plan for refactoring.
- Refactor first.
- Refactor in small steps.
- Set clear objectives.
- Test often.
- Automate wherever possible.
- Fix software defects separately.
- Understand the code.
- Refactor, patch and update regularly.
- Focus on code deduplication.

Unit Testing Continue

- **Unit Testing** is a software testing technique by means of which individual units of software i.e. group of computer program modules, usage procedures, and operating procedures are tested to determine whether they are suitable for use or not.
- Unit Testing is defined as a type of software testing where individual components of a software are tested. Unit Testing of the software product is carried out during the development of an application.
- Usually, unit test is performed by developers. Due to the reluctance of developers to test, quality assurance engineers also do unit testing.

Objective of Unit Test

- To isolate a section of code.
- To verify the correctness of the code.
- To test every function and procedure.
- To fix bugs early in the development cycle and to save costs.
- To help the developers to understand the code base and enable them to make changes quickly.
- To help with code reuse.

Advantages of Unit Testing

- Unit Testing allows developers to learn what functionality is provided by a unit and how to use it to gain a basic understanding of the unit API.
- Unit testing allows the programmer to refine code and make sure the module works properly.
- Unit testing enables testing parts of the project without waiting for others to be completed.
- Early Detection of Issues.
- Improved Code Quality.
- Increased Confidence.
- Faster Development.
- Better Documentation
- Facilitation of Refactoring
- Reduced Time and Cost.

Disadvantages of Unit Testing

- The process is time-consuming for writing the unit test cases.
- Unit Testing will not cover all the errors in the module because there is a chance of having errors in the modules while doing integration testing.
- Unit Testing is not efficient for checking the errors in the UI(User Interface) part of the module.
- It cannot cover the non-functional testing.
- Time and Effort
- Dependence on Developers
- Maintenance Overhead
- Difficult in Testing Interactions

Automated Testing

- **Automation Testing** is a software testing technique that performs using special automated testing software tools to execute a test case suite.
- On the contrary, Manual Testing is performed by a human sitting in front of a computer carefully executing the test steps.
- The automation testing software can also enter test data into the System Under Test, compare expected and actual results and generate detailed test reports.

Why Test Automation

- Manual Testing of all workflows, all fields, all negative scenarios is time and money consuming
- It is difficult to test for multilingual sites manually
- Test Automation in software testing does not require Human intervention. You can run automated test unattended (overnight)
- Test Automation increases the speed of test execution
- Automation helps increase Test Coverage
- Manual Testing can become boring and hence error-prone.

Automated Test Cases Process

- **Step 1)** Test Tool Selection
- **Step 2)** Define scope of Automation
- **Step 3)** Planning, Design and Development
- **Step 4)** Test Execution
- **Step 5)** Maintenance

Types of Automated Test Cases

- Smoke Testing
- Unit Testing
- Integration Testing
- Functional Testing
- Keyword Testing
- Regression Testing
- Data Driven Testing
- Black Box Testing