www.softwaretestinggenius.com

A Storehouse of Vast Knowledge on Software Testing and Quality Assurance

Design Review

&

Software Testing

Design Review & Software Testing

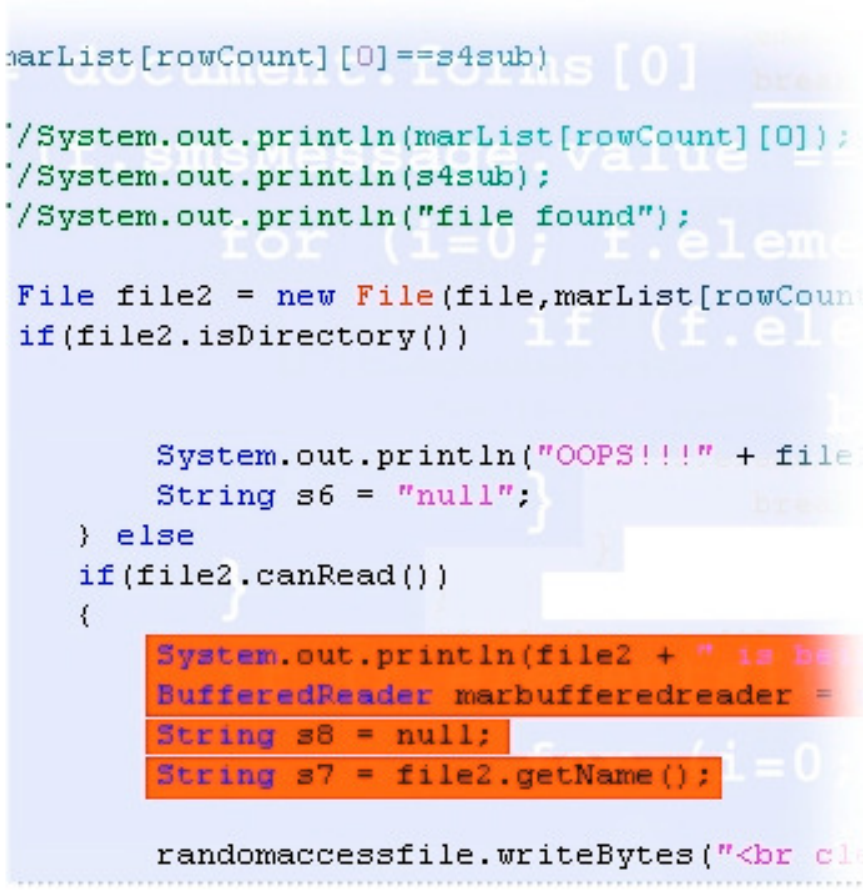
Software Malfunction: An Example

During the 1980s, the Atomic Energy of Canada Limited (AECL) produced a software embedded radiation therapy machine to treat cancer. The machine's software, which wasn't thoroughly tested, malfunctioned and killed at least six patients. The faulty software failed to detect that the machine's powerful electron beam was feeding patients with lethal doses of radiation. As a result, patients died from thermal and radiation burns. The ones that survived initially died later due to radiation poisoning.

Investigation reports revealed that the code of the software used wasn't independently tested. As a result, crucial bugs weren't detected. Later, it was also discovered that the design of the software model was flawed and that the review cycle that followed could not detect the flaws.

Design Review & Software Testing

Need for Review and Testing



No matter how careful the software development process has been, a software product is almost never bug free. Programmers, including those who conduct self reviews, tend to overlook their errors. A software testing team would evaluate the product objectively and help find bugs that would normally be overlooked during the development process. The ultimate goal of the testing phase is to create a product that is high on quality. For these reasons, testing is important, and it is a good practice to document the review and testing processes during the commencement of a project.

Bugs found after the software has been produced are often more expensive to fix. Reviewing the software after every stage of its development life cycle saves rework and expenses. For example, a review after the design stage might highlight flaws that would otherwise have been passed to the next stage and programmed by the coding team.

Design Review & Software Testing

Design Review

The design stage is the first major step in the development of software products and a review of the design is crucial to the further development of the software.

You begin the review process by mapping the completed design to the user requirements and tracking any inconsistencies. This is important if the team that collected the user requirements is different from the one that created the design.

Apart from carrying out a step back to ensure that the requirements have been properly understood, it is also important to ensure that the design is executable. The design should be communicated with the programming team to ensure that it is achievable within the project deadlines.

Design Review & Software Testing

SDLC Models

A design review should include an assessment of all the four components of the software design. These components include: Entity Relationship Diagram (ERD) and table design, program specifications, code review, and input and output layouts.

- ERD and Table Design
- Program Specifications
- Code Review
- Input and Output Layouts

Design Review & Software Testing

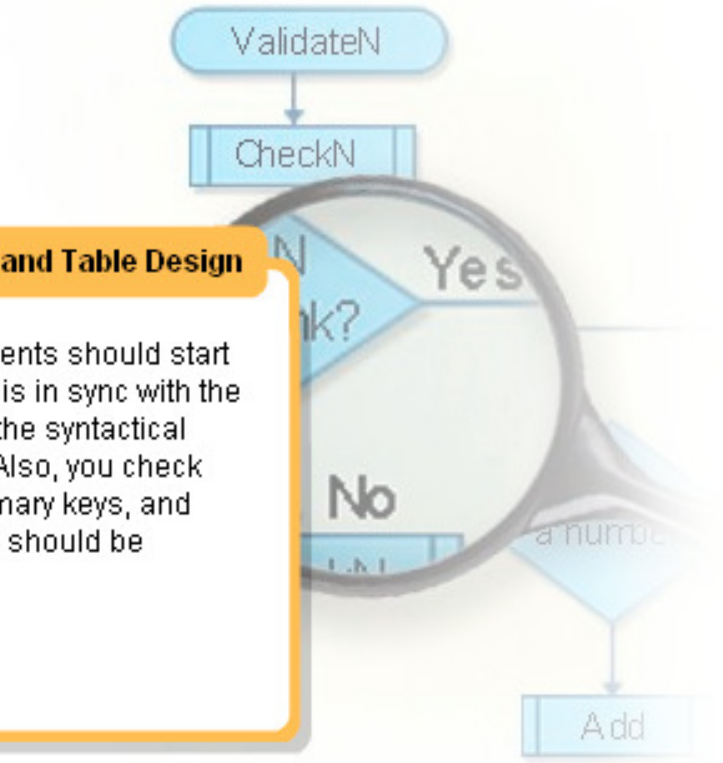
SDLC Models

A design review should include an assessment of all the four components of the software design. These components include: Entity Relationship Diagram (ERD) and table design, program specifications, code review, and input and output layouts.

- ERD and Table Design
- Program Specifications
- Code Review
- Input and Output Layouts

ERD and Table Design

A review of the design components should start with verifying whether the ERD is in sync with the tables. You do so by checking the syntactical rules about drawing the ERD. Also, you check the rules for normalization, primary keys, and foreign keys. Deviations, if any, should be rectified.



Design Review & Software Testing

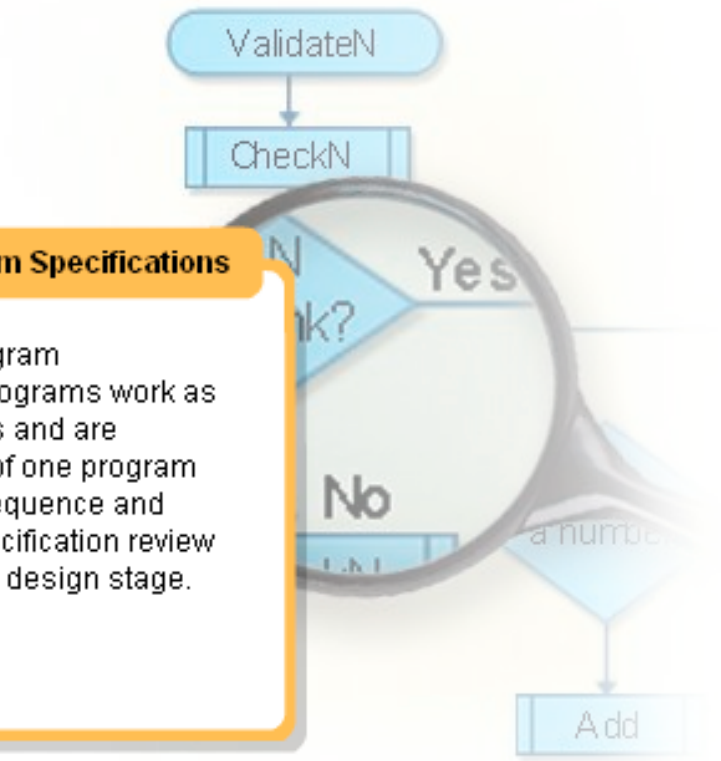
SDLC Models

A design review should include an assessment of all the four components of the software design. These components include: Entity Relationship Diagram (ERD) and table design, program specifications, code review, and input and output layouts.

- ERD and Table Design
- Program Specifications
- Code Review
- Input and Output Layouts

Program Specifications

You should ensure that all program specifications are reviewed. Programs work as a particular sequence of events and are interdependent. The absence of one program specification can disrupt the sequence and introduce bugs. A program specification review helps fix these problems at the design stage.



Design Review & Software Testing

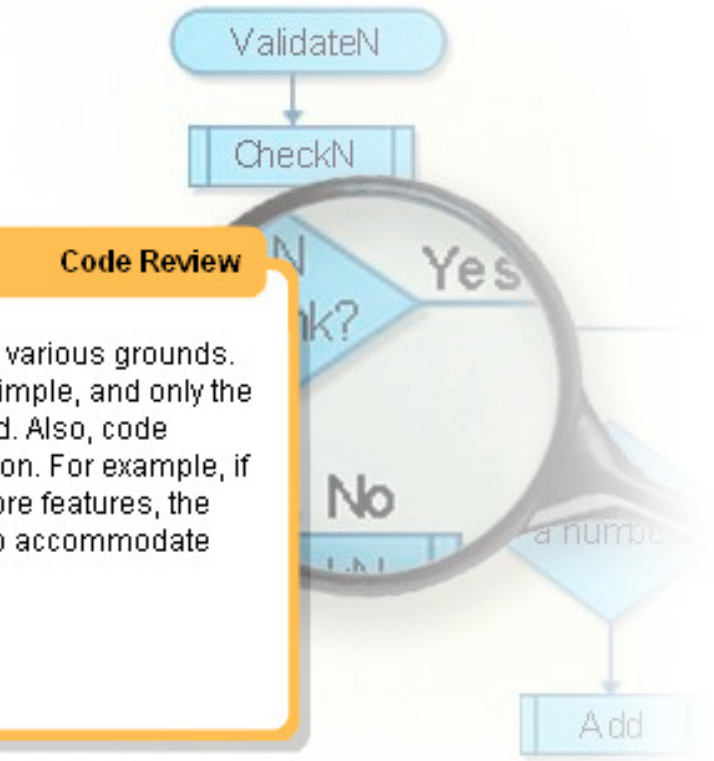
SDLC Models

A design review should include an assessment of all the four components of the software design. These components include: Entity Relationship Diagram (ERD) and table design, program specifications, code review, and input and output layouts.

- ERD and Table Design
- Program Specifications
- Code Review
- Input and Output Layouts

Code Review

You should review the code on various grounds. Code should be meaningful, simple, and only the required code should be added. Also, code should have scope for expansion. For example, if a particular program needs more features, the present code should be able to accommodate them without trouble.



Design Review & Software Testing

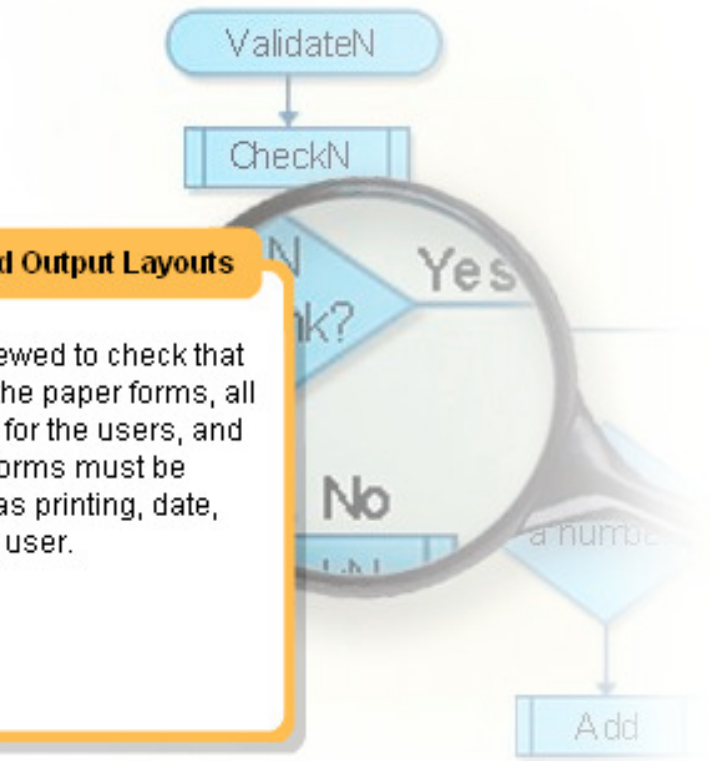
SDLC Models

A design review should include an assessment of all the four components of the software design. These components include: Entity Relationship Diagram (ERD) and table design, program specifications, code review, and input and output layouts.

- ERD and Table Design
- Program Specifications
- Code Review
- Input and Output Layouts

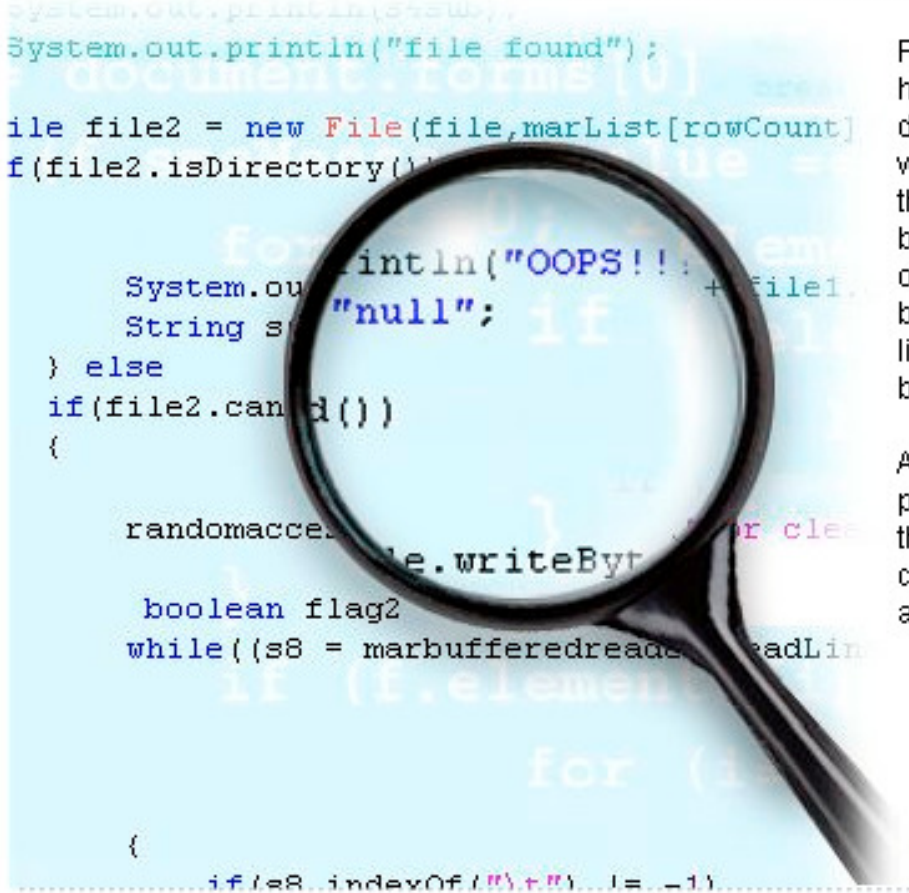
Input and Output Layouts

The input layout should be reviewed to check that input forms map perfectly with the paper forms, all forms have notes and prompts for the users, and all fields are validated. Output forms must be reviewed against factors such as printing, date, time, location, and name of the user.



Design Review & Software Testing

Code Review



Programmers may not find errors in code during a self-review. It's human to be attached to one's creation, and a self review alone does not assure bug-free code. This is because a programmer would follow the same logic when reviewing erroneous code. For the same reason, organizations undertake extensive reviews because a fresh set of eyes always helps find bugs that are overlooked by the programmer. In a research exercise conducted by Edward Yourdon, he and his three colleagues reviewed 200 lines of code and traced as many as 25 bugs. It wouldn't have been possible to trace 5 of these bugs during the testing stage.

A dedicated code review by the testing team should be an integral part of the software development life cycle. It has been observed that code review brings a 15 to 40% improvement in the quality of code. Reviews help remove extraneous code, ineffective logic, and inconsistencies against corporate standards.

Design Review & Software Testing

Types of Code Reviews

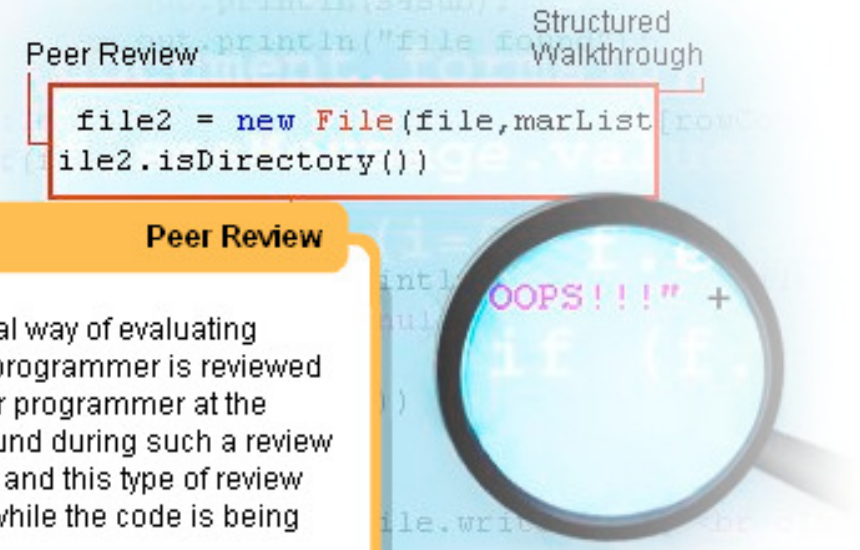
Once a programmer has finished with self-review, the quality of the code can be supported by three types of code reviews: Peer Review, Structured Walkthrough, and Fagan Code Inspection.

- Peer Review
- Structured Walkthrough
- Fagan Code Inspection

Design Review & Software Testing

Types of Code Reviews

Once a programmer has finished with self-review, the quality of the code can be supported by three types of code reviews: Peer Review, Structured Walkthrough, and Fagan Code Inspection.



- Peer Review
- Structured Walkthrough
- Fagan Code Inspection

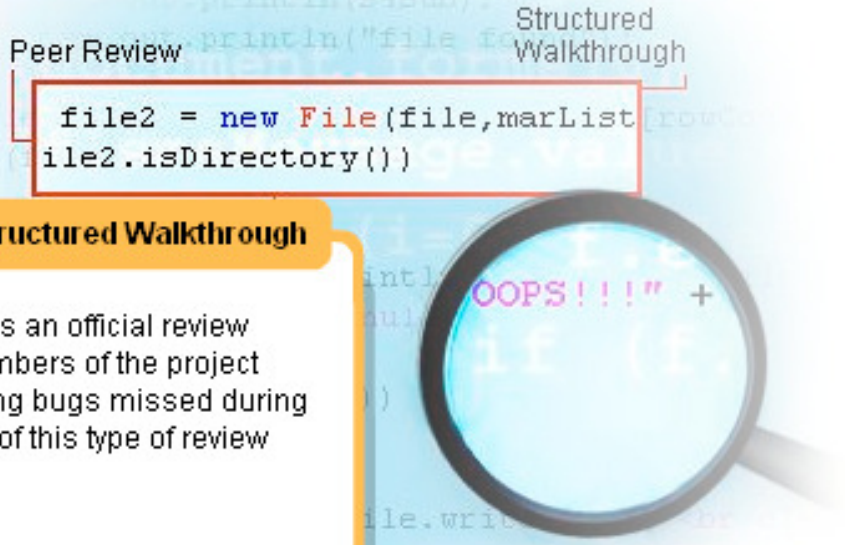
Peer Review

Peer review is an informal way of evaluating code. Code written by a programmer is reviewed by a peer, who is another programmer at the same level. The bugs found during such a review are not reported officially and this type of review helps in rectifying bugs while the code is being written.

Design Review & Software Testing

Types of Code Reviews

Once a programmer has finished with self-review, the quality of the code can be supported by three types of code reviews: Peer Review, Structured Walkthrough, and Fagan Code Inspection.



- Peer Review
- Structured Walkthrough
- Fagan Code Inspection

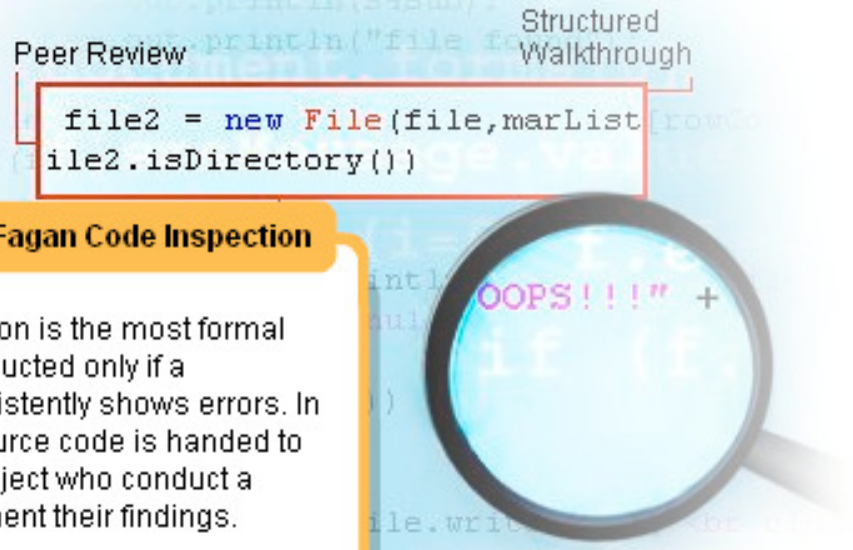
Structured Walkthrough

A structured walkthrough is an official review undertaken by senior members of the project team. It helps in eliminating bugs missed during peer review. The findings of this type of review are documented.

Design Review & Software Testing

Types of Code Reviews

Once a programmer has finished with self-review, the quality of the code can be supported by three types of code reviews: Peer Review, Structured Walkthrough, and Fagan Code Inspection.



- Peer Review
- Structured Walkthrough
- Fagan Code Inspection

Fagan Code Inspection

The Fagan Code Inspection is the most formal type of review and is conducted only if a programmer's code consistently shows errors. In this type of review, the source code is handed to a few members of the project who conduct a detailed study and document their findings.

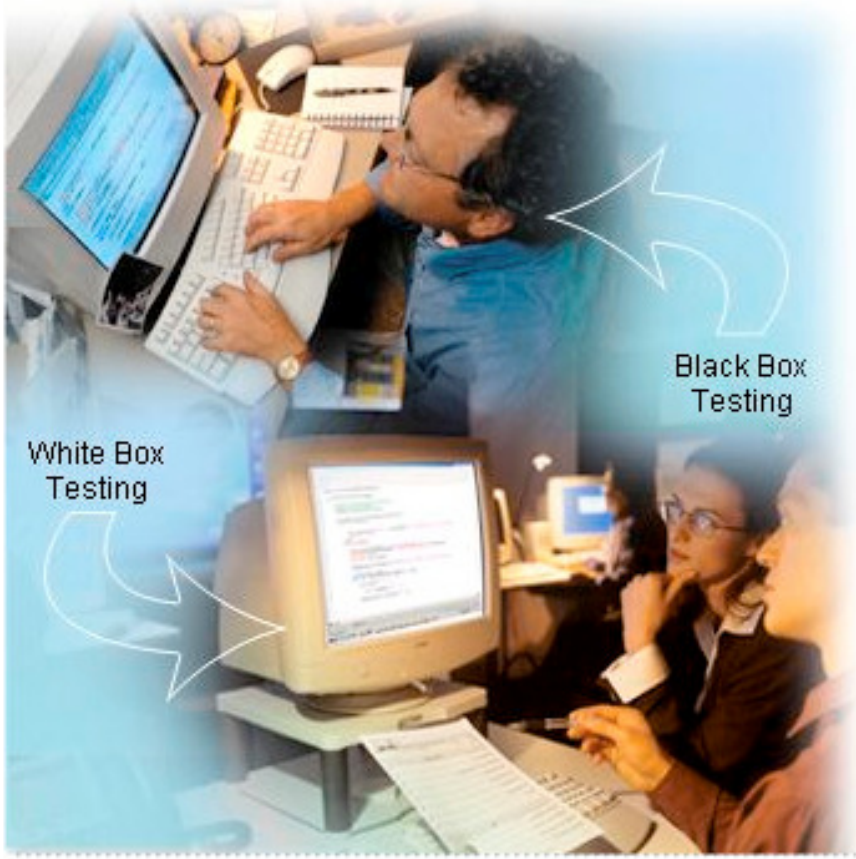
Design Review & Software Testing

Software Testing

After the software has been developed and made ready for final use, it is tested extensively before it is released for commercial use. Testing is the final and most crucial stage of the software development life cycle (SDLC). In fact, many organizations spend months, even years, to test complex software products. No matter how careful the development team has been, all software products always spring unpleasant surprises. For example, a particular software might function flawlessly on Windows XP, but might be sluggish on Windows 2000. Given the intense competition, companies cannot afford to discover bugs after a product has been released for commercial use.

Design Review & Software Testing

Black Box and White Box Testing



After the code has been developed, its functionality and source code are tested. Functionality testing is known as Black Box testing and source code testing is known as White Box testing. Software testing is said to be complete only after these two tests have been conducted.

In both types of testing, the tester must be someone from outside the project team. In Black Box testing, however, the tester should neither have access to the design documents or source code, nor have any prior knowledge of the working of the system. The advantage of Black Box testing is that it is conducted from the user's point of view. Thus, possible end user complaints are identified.

In White Box testing, the tester can access the design documents and source code of the software. In addition, knowledge programming language used is needed. Essentially, White Box testing helps in a review of the source code. The inputs and suggestions of experienced programmers help improve the quality of the system.

Design Review & Software Testing

Testing Process

A unit is the smallest testable part of software applications. Under unit testing, each unit of the software source code is tested to verify that it works independently and generates the expected output. The objective of this test is to verify whether all units of the software function collectively to meet the objective of the software.

Design Review & Software Testing

Testing Process

A unit is the smallest testable part of software applications. Under unit testing, each unit of the software source code is tested to verify that it works independently and generates the expected output. The objective of this test is to verify whether all units of the software function collectively to meet the objective of the software.

Unit Test

A unit is the smallest testable part of software applications. Under unit testing, each unit of the software source code is tested to verify that it works independently and generates expected output. The objective of this test is to ensure that each unit functions correctly as a standalone entity.

Design Review & Software Testing

Testing Process

A unit is the smallest testable part of software applications. Under unit testing, each unit of the software source code is tested to verify that it works independently and generates the expected output. The objective of this test is to verify whether all units of the software function collectively to meet the objective of the software.

Integration Test

It is possible that programs function correctly as independent units, but might throw errors when working as a group. Under integration testing, a group of units are tested collectively to determine whether they function correctly as a group to achieve their combined output.

Design Review & Software Testing

Testing Process

A unit is the smallest testable part of software applications. Under unit testing, each unit of the software source code is tested to verify that it works independently and generates the expected output. The objective of this test is to verify whether all units of the software function collectively to meet the objective of the software.

System Test

Under system test, the entire software is tested. The objective of this test is to test whether all units of the software function collectively to meet the objective of the software. It tests all the functionality of the software from the point of view of the end user.

Design Review & Software Testing

Testing Process

A unit is the smallest testable part of software applications. Under unit testing, each unit of the software source code is tested to verify that it works independently and generates the expected output. The objective of this test is to verify whether all units of the software function collectively to meet the objective of the software.

Regression Test

It is possible that programs that functioned properly during previous tests malfunction after the software has been released. It has also been observed that bugs resurface after they have been fixed due to poor revision control. The objective of a regression test is to find such bugs and fix them.

Design Review & Software Testing

Testing Process

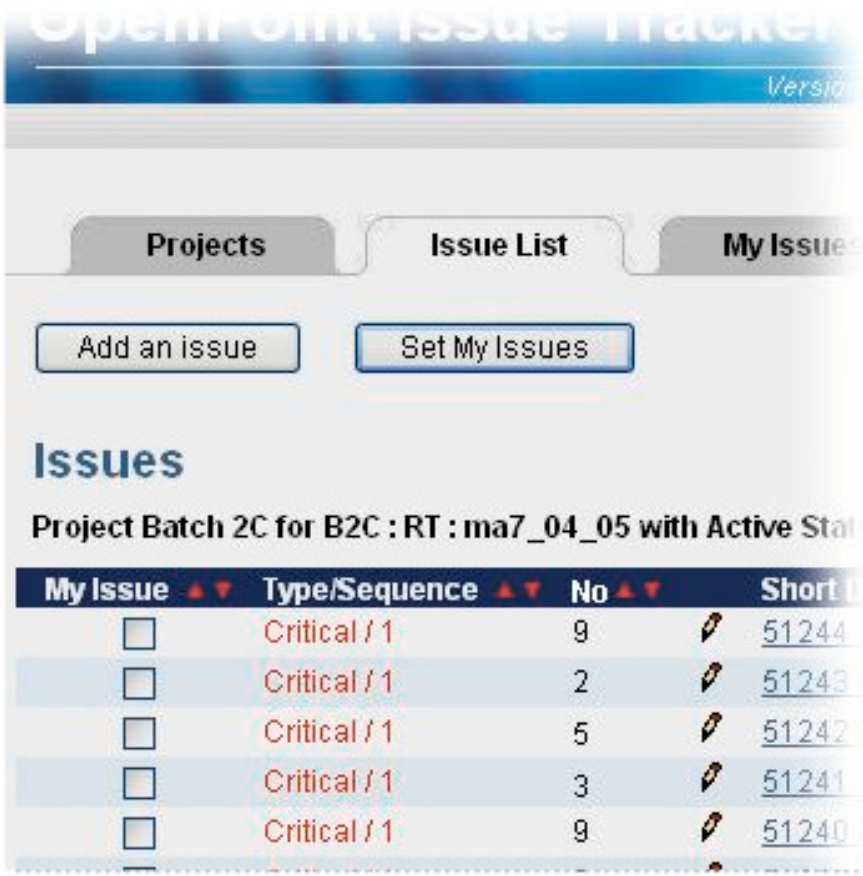
A unit is the smallest testable part of software applications. Under unit testing, each unit of the software source code is tested to verify that it works independently and generates the expected output. The objective of this test is to verify whether all units of the software function collectively to meet the objective of the software.

Acceptance Test

An acceptance test is a Black Box test conducted by the final user of the product. Generally, it is done by the final purchaser of the software to ensure that it is functional and satisfies requirements.

Design Review & Software Testing

Recording Bugs



Testing leads to the discovery of numerous bugs and each observation must be systematically recorded to avoid rework. To ensure that the development team knows exactly what to fix, the tester must record the details of the bugs, including:

- A detailed description of the location of the erroneous code or the steps that led to the bug, such as the name of the dialog box, the data entered, and the keys pressed
- A description of the erroneous outcome as compared to the expected outcome
- A reference to the error log generated by the software
- A screenshot of the error displayed by the software

Design Review & Software Testing

Check Point

Match each type of software test with its description.

Software Test	Description
	Each component of the software is tested individually.
	All the components are tested in a group for a favourable outcome.
	The software is tested as a whole for a favourable outcome.
	The software is tested after it has been released to fix the bugs that resurface.
	This is a functional test conducted by the final purchaser of the software.

- System Test
- Unit Test
- Regression Test
- Integration Test
- Acceptance Test

Design Review & Software Testing

Feedback

Not quite. The correct matches are displayed on the screen.

Software Test	Description
Unit Test	Each component of the software is tested individually.
Integration Test	All the components are tested in a group for a favourable outcome.
System Test	The software is tested as a whole for a favourable outcome.
Regression Test	The software is tested after it has been released to fix the bugs that resurface.
Acceptance Test	This is a functional test conducted by the final purchaser of the software.

Design Review & Software Testing

Check Point

Which of these describes Black Box testing?

- [ ] It is conducted by someone from within the project developing team.
- [ ] It is conducted to test the functionality of the software.
- [ ] It is conducted to test the source code of the software.
- [ ] It is conducted without sharing the design document with the tester.

Design Review & Software Testing

Check Point

Which of these describes Black Box testing?

- [ ] It is conducted by someone from within the project developing team.
- [x] It is conducted to test the functionality of the software.
- [ ] It is conducted to test the source code of the software.
- [x] It is conducted without sharing the design document with the tester.

Feedback

Correct. Black Box testing is conducted without sharing the design document with the tester to test the functionality of the software from the point-of-view of the final user.

Design Review & Software Testing

Check Point

Match each type of design review component to its correct description:

Design Review Component	Description
Code Review	Review of the rules for normalization, primary keys, and foreign keys
Input and Output Layouts	Review to check the flow of the programs to meet user requirements
ERD and Table Design	Review to check that the code is meaningful and as minimum as possible
Program Specifications	Review to check that forms map with the paper forms, forms have notes for the users, and forms are dated properly

Design Review & Software Testing

Check Point

Feedback

Not quite. The correct matches are displayed on the screen.

Design Review Component	Description
ERD and Table Design	Review of the rules for normalization, primary keys, and foreign keys
Program Specifications	Review to check the flow of the programs to meet user requirements
Code Review	Review to check that the code is meaningful and as minimum as possible
Input and Output Layouts	Review to check that forms map with the paper forms, forms have notes for the users, and forms are dated properly

Design Review & Software Testing

Check Point

Match each type of design review component to its correct description:

Design Review Component	Description
Code Review	Review of the rules for normalization, primary keys, and foreign keys
Input and Output Layouts	Review to check the flow of the programs to meet user requirements
ERD and Table Design	Review to check that the code is meaningful and as minimum as possible
Program Specifications	Review to check that forms map with the paper forms, forms have notes for the users, and forms are dated properly

Check Point

Feedback

Not quite. The correct matches are displayed on the screen.

Design Review Component	Description
ERD and Table Design	Review of the rules for normalization, primary keys, and foreign keys
Program Specifications	Review to check the flow of the programs to meet user requirements
Code Review	Review to check that the code is meaningful and as minimum as possible
Input and Output Layouts	Review to check that forms map with the paper forms, forms have notes for the users, and forms are dated properly

Design Review & Software Testing

Conclusion

Because software is rarely bug free, tests and reviews are necessary and crucial steps in the life cycle of software products. After the design is complete, it should be reviewed for feasibility of execution and user requirements. Each component of the design should be evaluated individually.

After a code is written, it should be reviewed by peers and a senior member. If the code still contains bugs, a Fagan Code Inspection should be conducted by elected members for a detailed study. After a software product has been developed, Black Box and White box tests should be conducted on it. Black Box testing involves testing the functionality, and White box test involves testing the source code.

Furthermore, developed software should be subjected to a unit test, integrating test, system test, regression test, and an acceptance test. The bugs found during the testing process should be recorded systematically so that they can be fixed easily.

Thank You