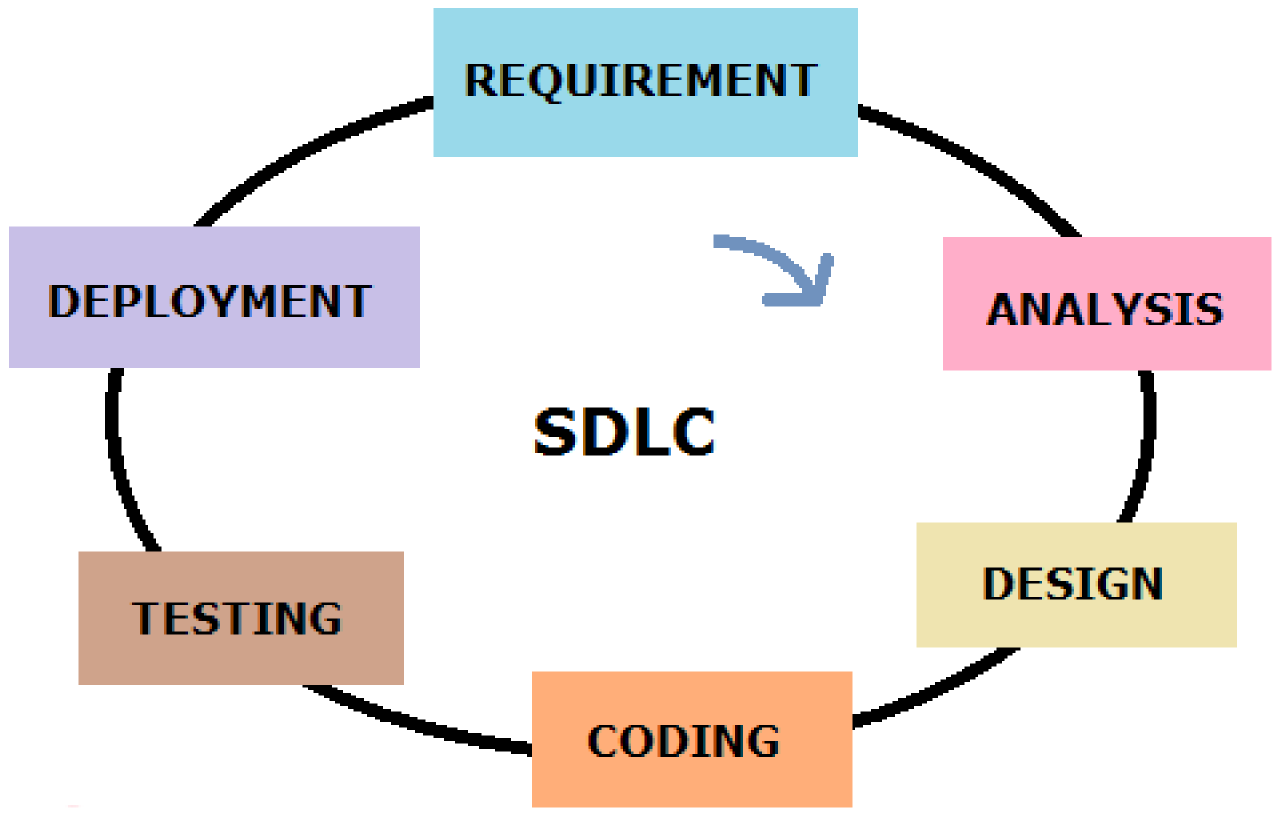
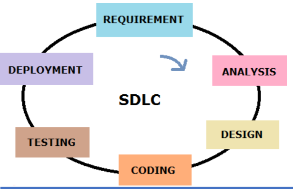
Software Development Life Cycle



Software Development Life Cycle (SDLC): A process that ensures good software is built.

Software: The **programs** and other operating information used by a computer.

Project: A series of tasks that need to be completed in order to reach a specific outcome. It can be done by an individual or a group of people.



Software Development Life Cycle

Requirement

Requirement engineering → Gathering + Analyzing + Documenting the requirements.

Carried out by **PO (Product Owner)** and **BA (Business Analyst).** **PM (Project Manager)** can also be with them)

The goal → is to develop **documents…**

Types of Requirement Documents →
- Business Requirements (BR)
- System Requirements Specification (SRS)
- Market Requirements (MR)
- Functional Requirements (FR)
- Non-Functional Requirements (NFR)

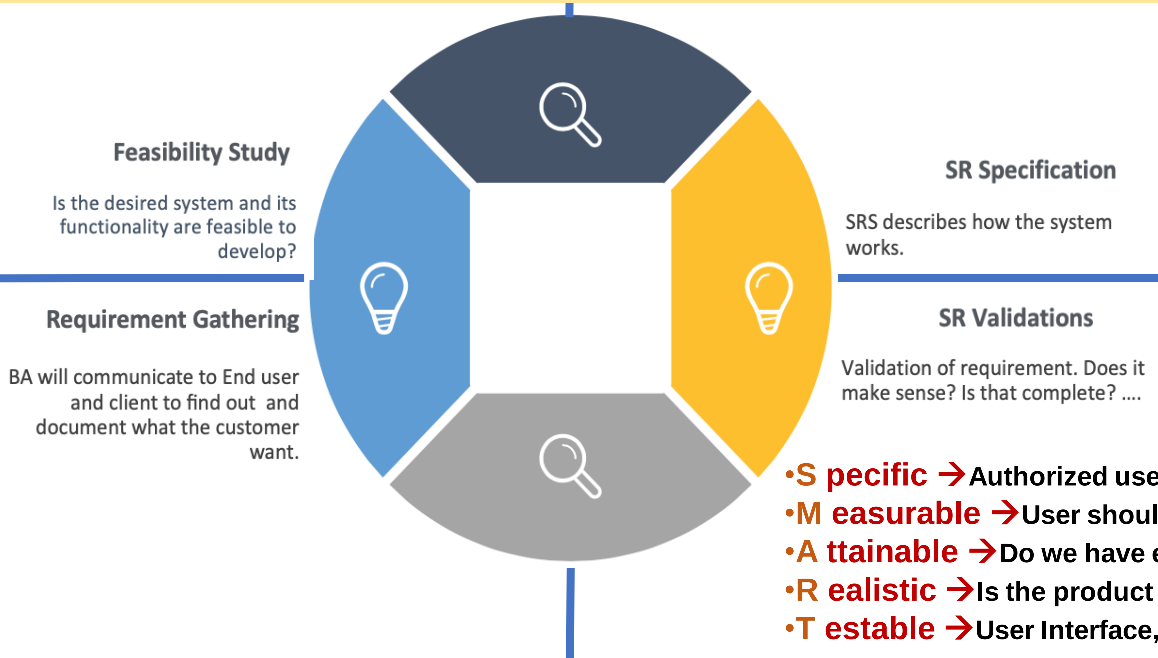
Once requirements are done, PO or BA will deliver documents to the designers.

Verification→
The process of evaluating products in the development phase to determine whether they meet the specified requirements for that phase.
Evaluation Items → Plans, Requirement Specs, Design Specs, CodeTest Cases

Validation→
The process of evaluating products during or at the end of the development phase to determine whether it satisfies specified business requirements.
Evaluation Items → The actual product or software

Testing activity should start with testing the requirements against to good requirement characteristics

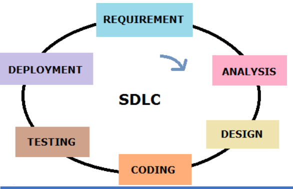
Is the product actually in demand?
Is the product marketable?
What is the return of investment?
What is the likelihood for success?


Feasibility Study
Is the desired system and its functionality are feasible to develop?
SR Specification
SRS describes how the system works.
Requirement Gathering
BA will communicate to End user and client to find out and document what the customer want.
SR Validations
Validation of requirement. Does it make sense? Is that complete? ....

Good Requirement Characteristics
- Clear
- Correct
- Consistent
- Unambiguous
- Modifiable

Requirement Sources
- Customers
- Partners
- End-Users
- Domain Experts
- Industry Analysts
- Reverse Engineering

Criterias
- **S pecific →**Authorized user with valid username and password should able to login.
- **M easurable →**User should able to login in 2 second after clicking login button.
- **A ttainable →**Do we have enough Budget / Resources / Time?
- **R ealistic →**Is the product realistic based on our resources and constraints
- **T estable →**User Interface, Graphic Design, Mock- Up



Software Development Life Cycle

Requirement

Requirement engineering → Gathering + Analyzing + Documenting the requirements.

Carried out by PO (Product Owner) and BA (Business Analyst).
PM (Project Manager) can also be with them.

The goal → is to develop documents…

Types of Requirement Documents →
- Business Requirements (BR)
- System Requirements Specification (SRS)
- Market Requirements (MR)
- Functional Requirements (FR)
- Non-Functional Requirements (NFR)

PO writes documents

PO writes user stories – each has its own acceptance criteria

All user stories are put to the product backlog in JIRA, Bitbug, etc.

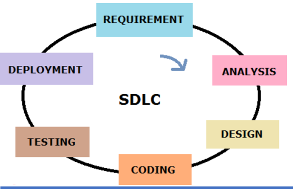
PO decides if the user story is epic or not.

- As a user, I should be able to buy an item from Amazon.

This story is very big, so we can create several user stories out of it. We call this type of user stories epic.

PO has to think which user story should be built first (prioritization by PO)

- Register
- Login
- Logout
- Start
- Play
- Pause
- Stop
- Restart



Software Development Life Cycle

Design

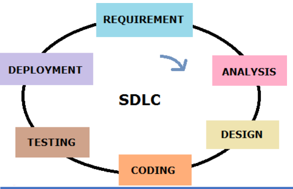
Executed by Designers and Business Team (PO, BA and PM)
They analyze the risks, write the documents and make the plan.

Design documents includes→

- Details on computer programming languages and environments
- Machines, packages, application architecture
- Distributed architecture layering, memory size, platform
- Algorithms
- Data structures, global type definitions, interfaces
- Many other engineering details…

Types of Design Documents →

- Architecture Document
- Implementation Plan
- Critical Priority Analysis
- Performance Analyses
- Test Plan



Software Development Life Cycle

Coding

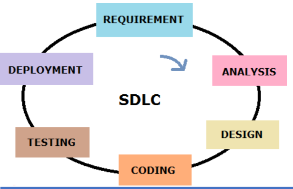
In this phase developers write code to build the application.
They work in the dev environment.

Dev Environments → Used by developers
www.dev-amazon.com

Test/QA Environments → Used by testers (QA1 for manuel testing, QA2 for automation team 1, QA2 for automaiton team 2, etc.)
www.qa-amazon.com
www.test0-amazon.com

Staging/Pre-production Environment → It is for regression and UAT testing and used by testers and UAT team
www.pre.amazon.com
www.stg.amazon.com

Production Environments
www.amazon.com



Software Development Life Cycle

Testing

Testing: Take measures to check the quality, performance, or reliability of something, especially before putting it into widespread use or practice. Software testing is a process of executing a program or application with the intent of finding the software bugs.

Software Testing: An activity to check whether the actual results match the expected results, to ensure that the software is defect/bug free, to evaluate the quality of the product, and to improve the product quality.

- Expected result → The client's request
- Actual result → Test results

There are two kinds of items that can be tested in an IT company

- Documents
- Application/Project

Testing Activities →

- Write Test Cases → Testers writes test cases based on SRS documents
- Execute Test Cases → Testers executes test cases, generally manually
- Log Defect → If they find a defect in a function, they inform the developers
- Retest Executed Defect → After developers correct the defect, retest is executed
- Automate Test Case → Test cases for automated test
- Execute Automated Test Cases → Executing automated test cases

The main goal of testing → Validating the software if it meets customer expectations!...

Why do bugs occur in software?

Software is built by human beings:

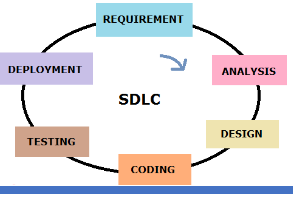
- who know something, but not everything
- who have skills, but aren't perfect
- who do make mistakes (errors)

Under increasing pressure to deliver to strict deadlines:

- no time to check everything
- assumptions may be wrong systems may be incomplete

Test Docs →

- Test Plan
- Test Scenario
- Test Case
- Traceability Matrix



Software Development Life Cycle

Testing

Software Quality Assurance →

- **Starts with verification process.**

Quality of Software can be Assured by performing Software Testing.

What's Quality? →

- **Depends on point of view.**

How can you avoid bad consequences? →

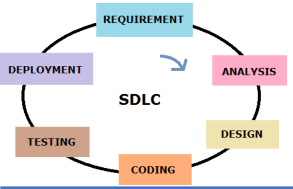
- **Quality is Perception.**
- **It depends on person's mind and expectations.**
- **Quality is a Mental Image.**
- **Understanding what quality means for others is more important than developing a "good quality" product.**

Software Quality Assurance is understanding →

- **What's the purpose of the software?**
- **The end user of the software?**
- **The Value Proposition of the software to the end user.**
- **What does it offer?**
- **What makes the software unique than others in the market.**
- **End - User's expectation.**
- **Stakeholder's expectation.**
- **Making sure the software Meets the Expectations**

Process of validating and verifying a software program, application or product should →

- **Meet the business and technical requirements that guided its design and development**
- **Work as expected**
- **Make sure 100% customer and end-user satisfaction.**



Software Development Life Cycle

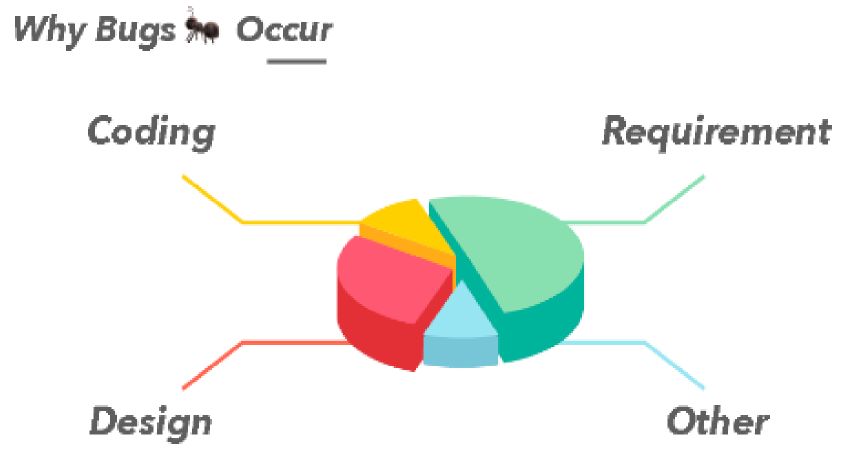
Testing

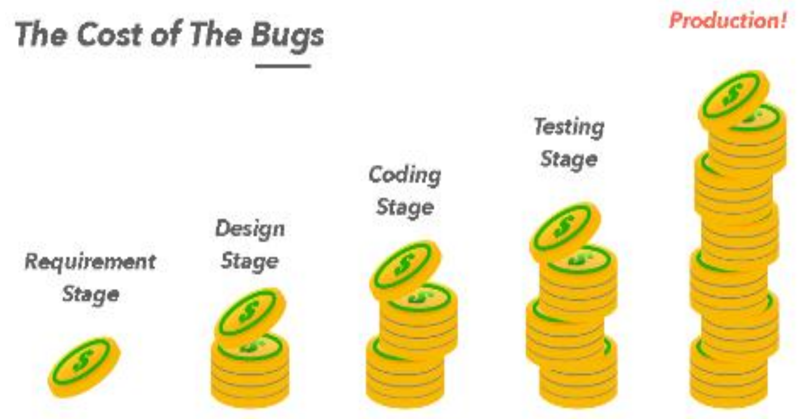
The Origin of the Bug →

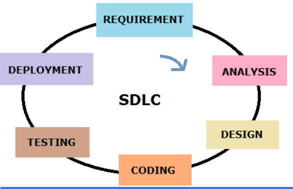
Grace Hopper was released from active service after the war in 1946 when she joined the Harvard Faculty at their Computation Laboratory. Here she continued her work on the Mark II and Mark III computers. On the 9th September 1947 Grace traced an error on the Mark II to a dead moth that was trapped in a relay. The insect was carefully removed and taped to the logbook and the term computer bug was coined. Henceforth the term "Bug" was used to describe any errors or glitches in a program.

The Definition of Bug →

Software testing is a process of executing a program or application with the intent of finding the software bugs. Bug is Software Functionality doesn't meet expectation.







Software Development Life Cycle

Testing

What Testers Do →

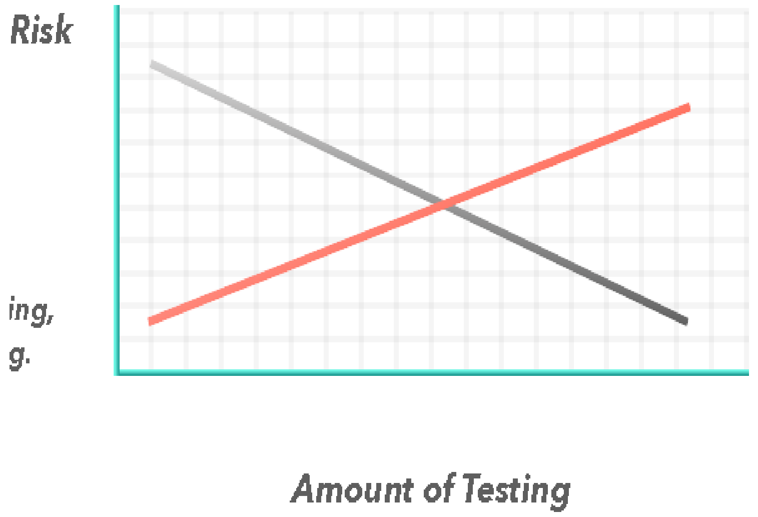
- The goal of a software tester is to find bugs.
- There are a lot of “testers” out there. They are only there for confirm things “working fine”. If you set up your tests only for the things that should work, they will pass, and you will never find a bug.
- Testers should have test to break approach instead of test to pass approach.
- The goal of a software tester is to find bugs and find them as early as possible.
- The earlier you find the lower the cost of the bug for the project.
- Good testers save the project tons of money by finding the bug before anyone else.
- The goal of a software tester is to find bugs and find them as early as possible and make sure they get fixed.
- Good testers should not afraid the pressure from developers and business analyst, they should be able to back up their opinion to make sure deliver top quality software.

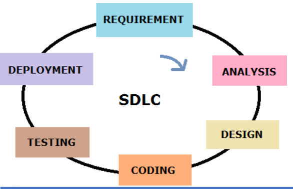
Is 100% Testing Possible → NO

- The number of possible inputs are Unlimited.
- The number of possible combinations of scenarios are Unlimited.
- The number of possible outputs are Unlimited.
- It is impossible to test every single scenario even in a simple calculator.
- 100% Testing is not possible, but 100% Customer & Client satisfaction is possible.

Risk Based Testing →

The more we test the less risk we have.
Since we can’t test everything, we can prioritize our testing.





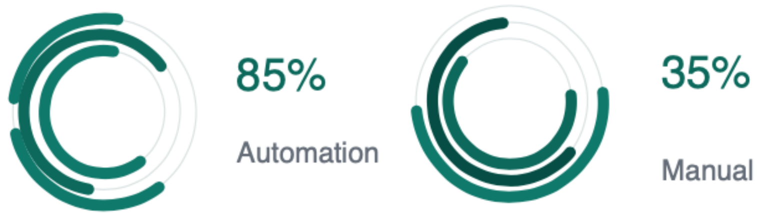
Software Development Life Cycle

Testing

Manual Testing

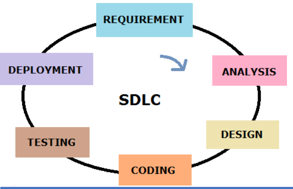
- *Manually testing is performed by Human*
- *Use manual testing tools like Jira, FireBug, Postman, Jmeter, …*
- *Manual testing helps testers understand the whole problem*
- *Any* ***new application must be manually tested*** *before its testing can be automated.*

Automation testing

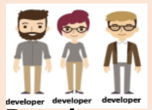
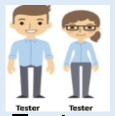


- *Automation testing is performed based on test cases(document)*
- *Use automation testing tools like Selenium, TestNG, RestAssured…*
- *Once the test is automated, no human intervention is required.*

Cybertek



Software Development Life Cycle Testing

Phase	Test Item	Test Phase	Test Types		
Requirement	★ Testing Starts		Review	Client PO BA Architecture	
Design	Documents	Verification Static Testing	Walkthrough Inspection		
Coding	Code Application	Validation Dynamic Testing	 Developers White Box (Glass)Testing	Unit Testing Integration Testing	
Testing	Code Application	Dynamic Testing Validation	Black Box Testing  Testers	System Testing UAT	→ GUI Testing Usability Testing Smoke Testing Regression Testing

Alpha Testing

Beta Testing

Static Testing

Review
Walkthrough
Inspection

Dynamic Testing

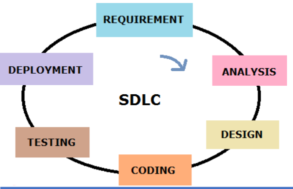
➢**Functional** → Clickable? Able to select? Able to check?

➢**Non-Functional**→ Security, installation, handling stress.

Verification	*Validation*
1. **Verification** is a static practice of verifying documents, design, code and program.	1. **Validation** is a dynamic mechanism of validating and testing the actual product.
2. It does not involve executing the code.	2. It always involves executing the code.
3. It is human based checking of documents and files.	3. It is computer based execution of program.
4. **Verification** uses methods like inspections, reviews, walkthroughs, and Desk-checking etc.	4. **Validation** uses methods like black box (functional) testing, gray box testing, and white box (structural) testing etc.
5. **Verification** is to check whether the software conforms to specifications.	5. **Validation** is to check whether software meets the customer expectations and requirements.
6. It can catch errors that validation cannot catch. It is low level exercise.	6. *It can catch errors that verification cannot catch. It is High Level Exercise.*
7. Target is requirements specification, application and software architecture, high level, complete design, and database design etc.	7. Target is actual product-a unit, a module, a bent of integrated modules, and effective final product.
8. **Verification** is done by QA team to ensure that the software is as per the specifications in the SRS document.	8. ***Validation** is carried out with the involvement of testing team.*
9. It generally comes first-done before validation.	9. It generally follows after **verification**.

Levels of software testing

➢Level one: Unit testing
➢Level two: Integration testing
➢Level three: System testing
➢Level four: User Acceptance Testing(UAT)



Software Development Life Cycle

Static Testing

Review →

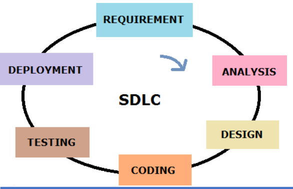
- **Typically used to find and eliminate errors or ambiguities in documents.**
- **Individual review, Group review, Requirement review, Design review, etc.**
- **Review also happen in developing and Testing phase of SDLC like Code review, Test plan review, Test case review, etc.**

Walkthrough →

- **Formal then review**
- **Discuss at peer level**
- **It is led by the authors**
- **Author guide the participants through the document according to his or her thought**
- **Process to achieve a common understanding and to gather feedback**
- **Useful for the people if they are not from the software discipline, who are not used to or cannot easily understand software development process.**
- **Is especially useful for higher level documents like requirement specification, etc.**

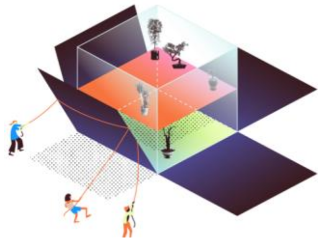
Inspection →

- **It is the most formal review type**
- **It is led by the trained moderators**
- **During inspection, the documents are prepared and checked thoroughly by the reviewers before the meeting**
- **It involves peers to examine the product**

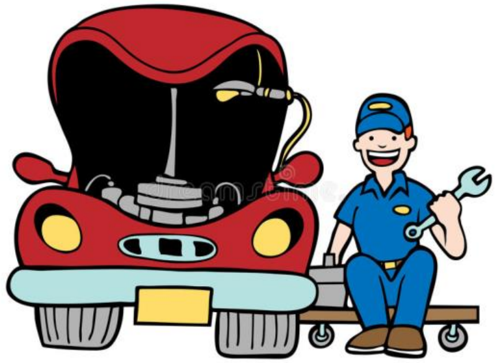


Software Development Life Cycle

Dynamic Testing



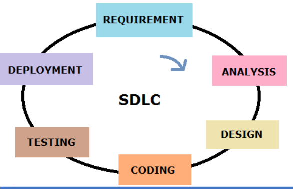
White Box Testing →
Do developers know what is wrong?
Yes, because they built it, they know inner logic.



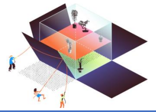
A mechanic knows what is the issue and how to fix it.

Black Box Testing →
Do testers know where to is wrong?
No, testers may not have knowledge of internal code or design .

I only know how to drive a car.
I know how to test break and gas paddle.
When there is an issue, I do not know about it.



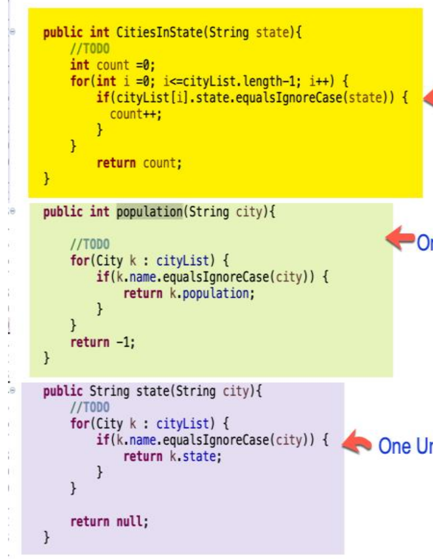
Software Development Life Cycle



White Box Testing

Unit Testing → 1st level of software testing

- A unit is the smallest testable part of an application like functions, classes, procedures, interfaces.
- Done by developers to make sure eliminate the bug before testers find it.
- Everyone are responsible for their own Unit Test.
- Could be offered as value proposition to the developers interviewing you.
- Before you deploy from develop environment to QA environment.
- It can be named as spelling check.
- It is preventive, proactive.



```
public int CitiesInState(String state){
    //TODO
    int count =0;
    for(int i =0; i<=cityList.length-1; i++) {
        if(cityList[i].state.equalsIgnoreCase(state)) {
          count++;
        }
    }
        return count;
}
```

This is just one unit of the code. Test only this single part is called Unit Testing

```
public int population(String city){

    //TODO
    for(City k : cityList) {
        if(k.name.equalsIgnoreCase(city)) {
            return k.population;
        }
    }
    return -1;
}
```

One Unit

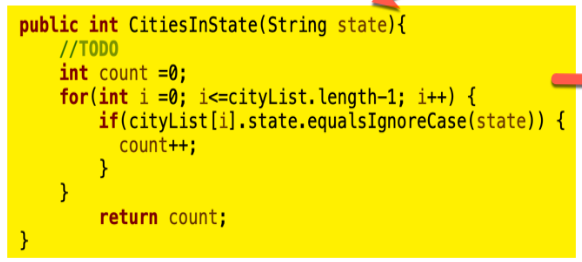
```
public String state(String city){
    //TODO
    for(City k : cityList) {
        if(k.name.equalsIgnoreCase(city)) {
            return k.state;
        }
    }

    return null;
}
```

One Unit

In this example, total 3 unit testing can be done by Developers

Integration Testing → 2nd level of software testing

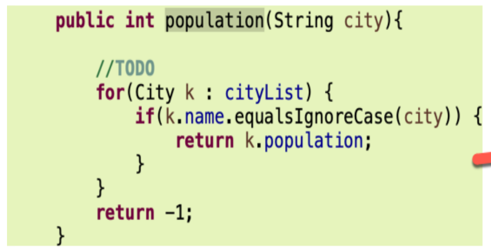
- Integration testing tests after integrating multiple component if the system works fine or not.
- Single component might work fine by itself but when integrated with multiple component it might fail.



```
public int CitiesInState(String state){
    //TODO
    int count =0;
    for(int i =0; i<=cityList.length-1; i++) {
        if(cityList[i].state.equalsIgnoreCase(state)) {
          count++;
        }
    }
        return count;
}
```

When a user enter a state name, the code will print all cities in that State.

Input : State
Output : Cities



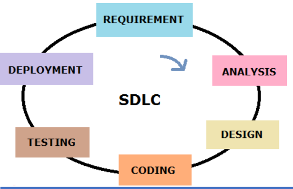
```
public int population(String city){

    //TODO
    for(City k : cityList) {
        if(k.name.equalsIgnoreCase(city)) {
            return k.population;
        }
    }
    return -1;
}
```

If we take a city's name from above code, and enter to this code, it will print the population.

Input : a city's name
Output : Population

Integration testing :
Combination of above 2 unit tests , take the 1st code's output as the second code's input, and check if they work together or not.

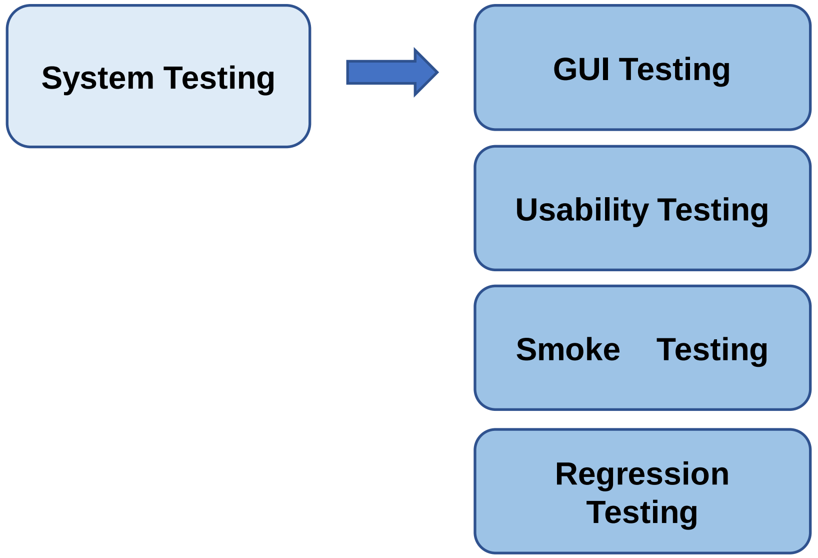
(Can conbine more then 2 unit tests)



Software Development Life Cycle

Black Box Testing

System Testing → 3rd level of software testing

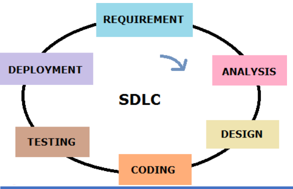


UAT Testing → 4th level of software testing

Graphical User Interface testing

Mainly test the application's:

Spelling error
Color
error messages Format
Size of an image
Etc.



Software Development Life Cycle

System Testing

GUI Testing →

Graphical User Interface testing

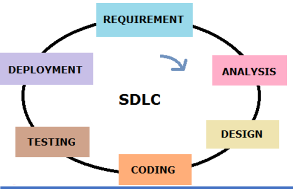
Mainly test the application's:

Spelling error
Color
Error messages
Format
Size of an image
etc.

Usability Testing →

Evaluates the application on how easy it is for use it.
Check if the application is provided "Help" or not.

It can be a module of the application, or separate documentation.



Software Development Life Cycle

System Testing

Smoke Testing →

Only the major functionalities of the application will be tested automatically with the help of some tools.

The codes run before you start your work, and send the result to your team member's email automatically.

- **Login**
- **Order food**
- **See menu**
- ***Do payment***

Example:

First month your team developed *Registration*, *login* and *logout* functionality;
Second month your team developed *food ordering* and *payment* functionality;
Third month your team developed *receive conformation emails and write comments.*

Regression Testing →

Whenever a new functionality added to the application, we have to make sure it shouldn't affect the existing functionalities/features.

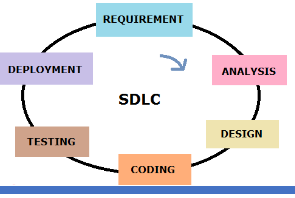
So far, we finished 150 user stories:
In the new sprint, we will add one more functionality, and we will add 5 more User stories.

As a user, I should be able to collect my favorite foods and drinks in "Favorites" module.

We add "Favorites" functionality, login-logout, food ordering and all existing functionalities should work as expected.

So we test all 150 + 5 requirements at the same time.

(Some companies only choose half of them as regression because it takes a long time to test or they do not have enough automation tester, or no tools to help running regression faster)



Software Development Life Cycle

Functioanl/Non-Functional Testing

Functional Testing →

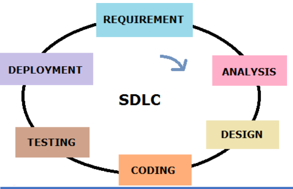
is performed to verify that a software application performs and functions correctly.

What is a functionality of an application/project? → what the application is capable of or is used for. (Login, Logout, Search food ,do payment, write comments, etc.)

Non-Functional Testing →

- Performance testing → (mainly test the speed of the application)
  - Load Testing →
    - 10/20 costumer use the Cyberburger at the same time
    - Every time increase the number of user a little bit
    - Observe/test if the application is become slower or not
  - Stress Testing → Test the respond of the application under the stress. Immediately increase or decrease load, or number of user, from 10 to 90, and give stress to the application.
  - Volume Testing → How much of data that the application is able to handle.
- Security Testing →
- Installation Testing →

An IT Company has a team called "*Performance team*"
Performance team responsible for non-functional testing.



Software Development Life Cycle

Other Tests

Exploratory Testing →

- Exploratory testing is about exploring, finding out about the software, what it does, what it doesn't do, what works and what doesn't work.
- The tester is constantly making decisions about what to test next and where to spend the (limited) time.
- This is an approach that is most useful when there are no or poor requirement specifications and when time is severely limited.
- This test can be done without any knowledge of the application, or any documentation.

Ad Hoc/Random/Monkey Testing →

• Performed without proper planning and documentation.

• Testing is carried out with the knowledge of the tester about the application

End to End Testing →

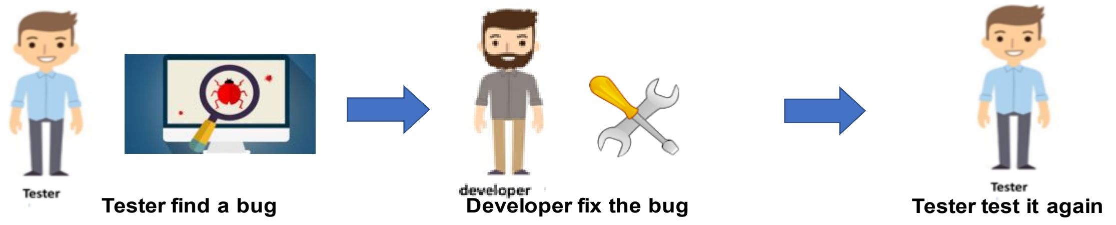
Testing the overall functionalities of the system including the data integration among all the modules is called end-to-end testing.

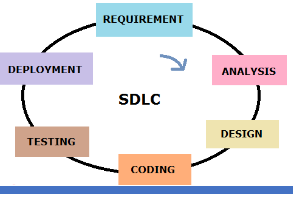
Example:
As a user, once I login, I should be able to see food menu and order food, and then I can make a payment, then I should be able to logout.

Re-Testing →

When there is a bug, a tester writes a bug report, and the developer fix it.

After developer fixing, testers should test it again.





Software Development Life Cycle

Other Tests

Component Testing →

- Stand alone, only one part, engine in the car, separately from the car itself.
- Component testing is a method where testing of each component in an application is done separately.
- Component testing is also known as module and program testing. It finds the defects in the module and verifies the functioning of software.
- Before integrating the components of the software
- the single component has to be working fine.

Continuous Integration →

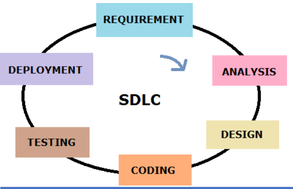
- We have 5 developers. Each of them writes code.
- When we compile the code at the end of the day, the new change in the code could unintentionally break other functionalities.
- Now, we have to run regression for all to find whose fault is that?
- Chi Ching - Continues Integration was invented…
- Each Developer Checks in the code, CI Server will trigger

Positive Testing →

Positive testing can be performed by testing the application with valid input.
It is also called “Happy Path” Testing.
If you try to login with valid username and password it is positive testing.

Negative Testing →

Negative Testing can be performed on the system by providing invalid data as input.
It checks whether an application behaves as expected with the negative input.
This is to test the application that does not do anything that it is not supposed to do so.
How many children do you have? 3.5
(trying to input this value to the application is negative test)
Login with invalid username and password



Software Development Life Cycle

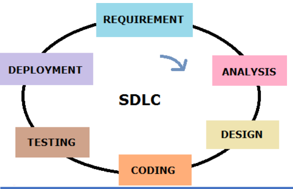
Other Tests

Equivalence Class Partitioning →

- Assume you are buying grape and you are not sure it is sweet or not, what would you do?
- Would you eat entire grapes and then come to the conclusion to buy the grapes?
- You would take one grape and taste it.
- If the one that you have tasted is sweet, you would think the entire basket of grape is sweet.
- You have just used Equivalence Class Partitioning method to buy the grape.
- Then you will buy the grapes

- In this method the input domain data is divided into different equivalence data classes. This method is typically used to reduce the total number of test cases to a finite set of testable test cases, still covering maximum requirements.
- If there is input box call "Enter your age" and will only take numeric value 0 to 99.
- You would have 100 test cases to execute.
- With EP Method I will only use below sets of data test that input box.
- 0, 10, 50, 99.
- If the input box is able to take above four numeric value, I would assume it will work from 0 to 99 with no problem.
- Now instead of 100 test cases I have only executed 4 test cases and I am still getting the same result.

- If team doesn't have enough time, we can pick first row and last row as simple.
- If first and last row are accurate we assume the rest of the rows are accurate.
- Based on equivalence class partitioning method.

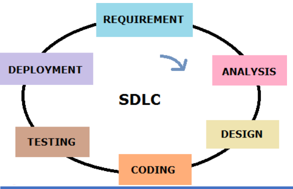


Software Development Life Cycle

Other Tests

Boundary Value Analysis →

- You have a credit card and it has spending limit of $5000.
- You should at least spend $50 to avoid the monthly fee.
- The $50 is Lower Boundary or Lower Limit.
- The $5000 is Higher Boundary or Higher Limit.
- The will try to keep all the transactions between $50 and $5000.
- It means you will be always inside the Boundary.
- Less than $50 will make you pay fee.
- Higher than $50000 will make your credit maxed out.
- Boundary testing is make sure the software accepts valid data inside the valid Boundary.
- Boundary testing is make sure the software rejects invalid data outside the valid Boundary.
- If you have $5000 spending limit and when you spend $6000 and system did not stop, you or give you proper warning message it means there is error or bug in the system.
- If you spend less than $50 and your credit card still did not charge you for monthly fee it means, there is a defect in the credit card application.
- Upper Limit Reached on Commitment ID
- $200 Million lots of transaction in one day.
- Upper Limit Jan 31
- Includes Jan 31
- Developer Forgets
- Enrollment >= Jan 31



Software Development Life Cycle

Other Tests

Hot Fix →

- A small piece of code developed to correct a major software bug or fault and released as quickly as possible.
- Parallel Execution of Regression Suite

508 Compliance Testing →

- Section 508 is a requirement for government websites, and in general, for Federal agencies' electronic and information technology to be accessible to people with disabilities.
- Basically some websites like healthcare.gov must perform 508 testing to make sure disable people can use it as well.
- 508 Compliance is mandatory for government websites.

Browser Compatibility Testing →

- The web application needs to work as expected on all the major web browsers.
- Why Need Automation?
- 1000 Test Cases become 4000 with browser compatibility testing.

Requirement Traceability Matrix →

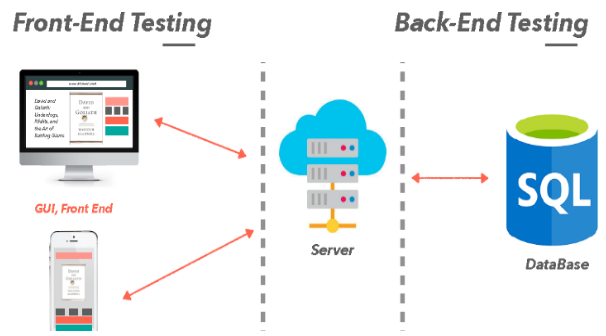
- Test Case Coverage
- JIRA-User Stories
- ALM

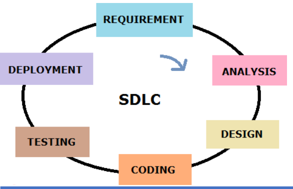
Penetration Testing →

- A penetration test, colloquially known as a pen test or pen-test, is an authorized simulated cyberattack on a computer system, performed to evaluate the security of the system

Front-End/Back-End Testing →

- Front-End: Graphical User Interface – GUI
- Back_End: Database





Software Development Life Cycle

Product

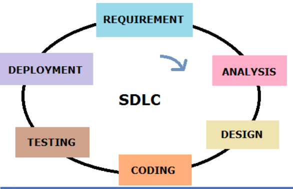
Once the product is tested, it is deployed in the production environment. The application is on the market, and end user can use it.

Dev Environments → Used by developers
www.dev-amazon.com

Test/QA Environments → Used by testers (QA1 for manuel testing, QA2 for automation team 1, QA2 for automaiton team 2, etc.)
www.qa-amazon.com
www.test0-amazon.com

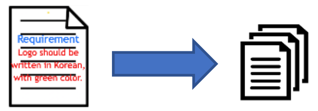
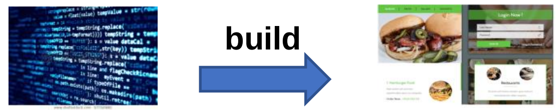
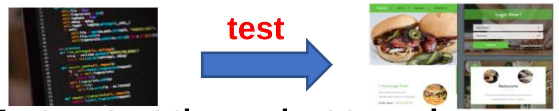
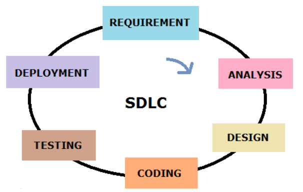
Staging/Pre-production Environment → It is for regression and UAT testing and used by testers and UAT team
www.pre.amazon.com
www.stg.amazon.com

Production Environments
www.amazon.com



Software Development Life Cycle

Summary

SDLC steps	Input for the step	Involved roles	Output from the step
Requirement	Client's requirements	PO +BA	 Requirement Logo should be written in Korean, with green color. All kinds of requirement documents are written
Design	Requirement documents	Architecture	Design documents & Risk analysis are done
Coding	Requirements	Developers	 Developers write code to build the project
Testing	Project is built, and ready to test	Testers	 Testers test the project to make sure client and end users use bug-free application
Product	Tested bug-free application	Developers + Testers	End user now can use the application
Maintinance	• Client wants any changes? • Any hidden bugs appeared while the end user are using the application? • Unidentified real-world problems?	The team	 Repeat the SDLC steps again