**What is SDLC?**

Software development life cycle is essential a series of steps or phases that provide a model for the development and lifecycle management of an application of piece of software.

**Different phases of SDLC that are**:

1. **Requirement Gathering:**

* Feature
* Usage scenarios
* Requirement will change
* Although requirements may be documented in written form, they may be incomplete, unambiguous or even incorrect.

Functional and Non-functional

1. **Analysis Phase** :

* Ideally**,** the document stales in a clear and precise fashion what is to be built.
* The analysis represents the “what” phase.
* The requirement documentaries to capture the requirements from the customer’s perception.

1. **Design Phase :**

* Design architecture document
* Implementation plan
* Critical priority analysis
* Performance analysis
* Test plan

1. **Implementation phase :**

* In the implementation phase the team builds the component either from scratch or by compositions.
* The implementation phase deals with issue of quality, performance, baseline, libraries and debugging.

1. **Testing phase :**

* Simply stated quality is very important.
* It is much easier to explain to a customer why there is a missing feature than to explain to a customer why the product takes quality.
* The customers satisfied with the quality of a product remain loyal and wait for a new functionality in the next version.
* Quality is distinguishing attribute a system indicating the degree of excellence.

**What is software testing?**

A software testing is a process used to identify the correctness completeness and quality of a developed computer software. It includes a set of activities conducted with the intent of finding error in software so that it could be corrected before the product is released to the end user.

* **Testing Activities:**
* Planning and control.
* Choosing test conditions.
* Designing test cases.
* Checking results.
* Evaluating completion criteria.
* Reporting on the testing process and system under test.
* Finalizing or closer (e.g. after a test phase has been completed.)
* Testing also includes reviewing of documents (including source code) and static analysis.
* **Test Objectives:**
* Finding Defects.
* Gaining confidence in and providing information about the level of quality.
* Preventing Defects.
* Both dynamic testing and static testing can be used as a means for achieving these objectives.
* **Why testing is necessary?**
* Testing is necessary because we all make mistakes.
* Some of mistakes are un-important, but some of the more **Expensive or dangerous.**
* We need to check everything and anything.
* **7 key principles:**
* Testing shows presence f defects.
* Exhausting is impossible.
* Early testing.
* Defect clustering.
* The pesticide paradox.
* Testing id context dependent.
* Absence of error fallacy.

**What is agile methodology?**

* Agile SDLC model is a combination of iterative and incremental process model is with focus on process adaptability and customer satisfaction by rapid delivery of working software product.
* Agile methods break the product into small incremental builds.
* The builds are provided in iterations.
* Each iteration typically lasts from about one to three weeks.
* Every iteration involves cross functional teams. Working simultaneously on various areas like planning, requirements analysis, design coding, unit testing, acceptance testing.
* At the end of the iteration a working product is displayed to the customer and important stakeholder.

**What is agile?**

* Agile through process had started early in the software development and started becoming popular with time due to its flexibility and adaptability.
* Iterative approach is taken and working software builds is delivered after each iteration. Each build is incremental in terms of feature; the final build holds all the features require by the customer.

**Pros of Agile:**

* Is a very realistic approach to software development
* Promotes team work and cross training?
* Functionality can be developed rapidly and demonstrated.
* Resource requirements are minimum.
* Good model for environments that change steadily.

**Cons of Agile:**

* Not suitable for handling complex dependencies.
* More risk of sustainability, maintainability and extensibility.
* Depends heavily on customer interaction, so if customer is not clear, team can be driven in the wrong direction.
* There is high individual dependency, since there is minimum documentation generated.

**What is SRS?**

* SRS stands for “software requirement specification” it is a document prepared by business analyst and system analyst.
* It describes what will be the feature of software and what will be its behavior i.e. how it will perform.
* It is the detail description of software system to be developed with its functional and non-functional requirement.
* The SRS consist of all necessary requirements required for the project development.
* In order to get all details of software from customer and to write the SRS document system analyst is required.
* SRS document is actually an agreement between the client and developer.

**What is oops?**

* Identifying objects and assigning responsibilities to these objects.
* Object communicates to other objects by sending messages.
* Messages are received by the methods of an object.
* An object is like a black box.
* The internal details are hidden.

**Concept of oops:**

* **Class:** Class is a collection of data member and member function.
* **Object:** object gives permission to access the functionality of class.
* **Encapsulation:** Wrapping of data.
* **Polymorphism:** one names multiple forms.
* **Over ridding:** function with same name and same parameter.
* **Over loading:** function with same name and different parameter.
* **Abstraction:** It is the process where only essential information is given and details are hidden.

**Explain phases of waterfall model**

Requirements collection

Analysis

Design

Implementation

Testing

Maintenance

nance

The classical software lifecycle and the software development as a step by step waterfall between the various development phases.

The waterfall is unrealistic for many reasons, especially:

* Requirements must be frozen to early in the lifecycle.
* Requirements are validated too late.

**Applications (when to use? )**

* Requirements are very well documented, clear and fixed.
* Product definition is stable.
* Technology is understood and is not dynamic.
* There are no ambiguous requirements.
* Well understood mile stones.
* Easy to arrange tasks.

**Cons (why not waterfall model)**

* No working software is produced until late during the lifecycle.
* High amount of risk and uncertainty.
* Not a good model for complex and object-oriented projects.
* Poor model for long and ongoing projects.
* It is difficult to measure progress within stages.
* Cannot accommodate changing requirements.

**Pros**

* Simple and easy to understand and use.
* Easy to manage due to the rigidity of the model. Each phase has specific deliverables and review process.
* Phases are processed and completed one a time.
* Works well for smaller projects where requirements are very well understood.
* Clearly defined stages.

**Write phases of spiral model**

**Planning:** Determination of objectives, alternatives and constraints.

**Customer Evaluation:** Assessment of the results of engineering.

**Risk Analysis:** Analysis of alternatives and identification/resolution of risks.

**Engineering:** Development of the “next level” product.

Spiral model is very widely used in the software industry as it is in synch with the natural development process of any product i.e. learning with maturity and also involves minimum risk for the customer as well as the development firms. Following are the typically uses of spiral model:

* When costs there are a budget constraint and risk evaluation is important.
* For medium to high risk projects.
* Long term project commitment because of potential changes to economic priorities as the requirements changes with time.
* Customer is not sure of their requirements which are usually the case.
* Requirements are complex and need evaluation to get clarity.

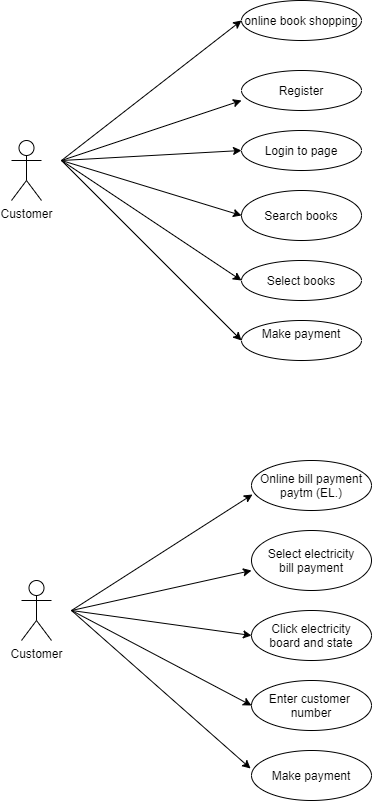
**Pros (why it works)**

* Changing requirements can be accommodated.
* Allows for extensive use of prototypes.
* Requirements can be captured more accurately.
* Users see the system early.
* Development can be divided into smaller parts and more risky parts can be developed earlier which helps better risk management.

**Cons (why doesn’t work)**

* Management is more complex.
* End of project may not be known early.
* Not suitable for small or low risk projects and could be expansive for small projects.
* Process is complex.
* Spiral may go in definitely.
* Large number of intermediate stages requires excessive documentation.

**Draw use-case of online book shopping product and online bill payment**

****

****

