Process Instruction : Any statement in a language to perform a task.

Eg.

- Declaring a variable
- Assigning a constant to a variable
- Some arithmetic expression
- A comment
- Defining a function
- Calling a predefined function etc.

Program : A set of process instructions

Software : A set of programs of a real time applications.

Eg. Banking app, Healthcare app, Payroll app, Inventory control app etc.

Software Evolution:

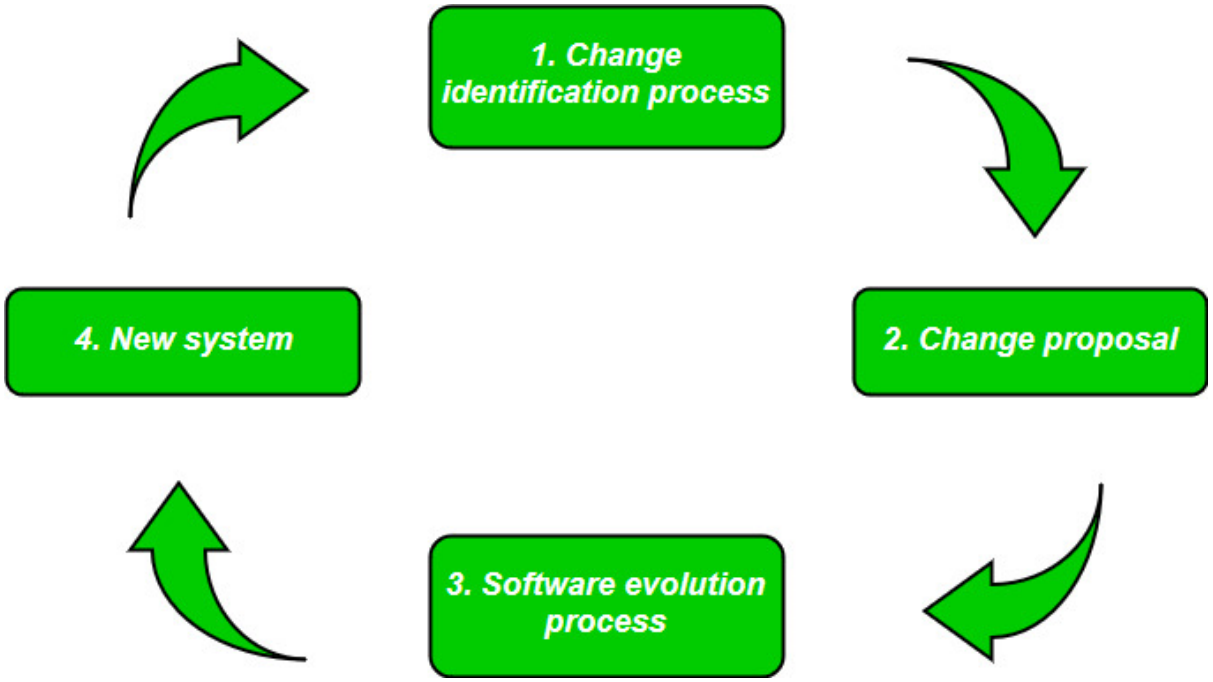
Software Evolution is a term that refers to the process of developing software initially, and then timely updating it for various reasons, i.e., to add new features or to remove obsolete functionalities, etc.

Identify the importance and challenges of different types of software maintenance, and learn the seven phases of maintenance for large software systems.

Software changes are inevitable because there are many factors that change during the life cycle of a software.

Software evolution involves the overall development and improvement of software according to specific purposes.

vidyasagar.bandaluppi@gmail.com



Software Evolution

---

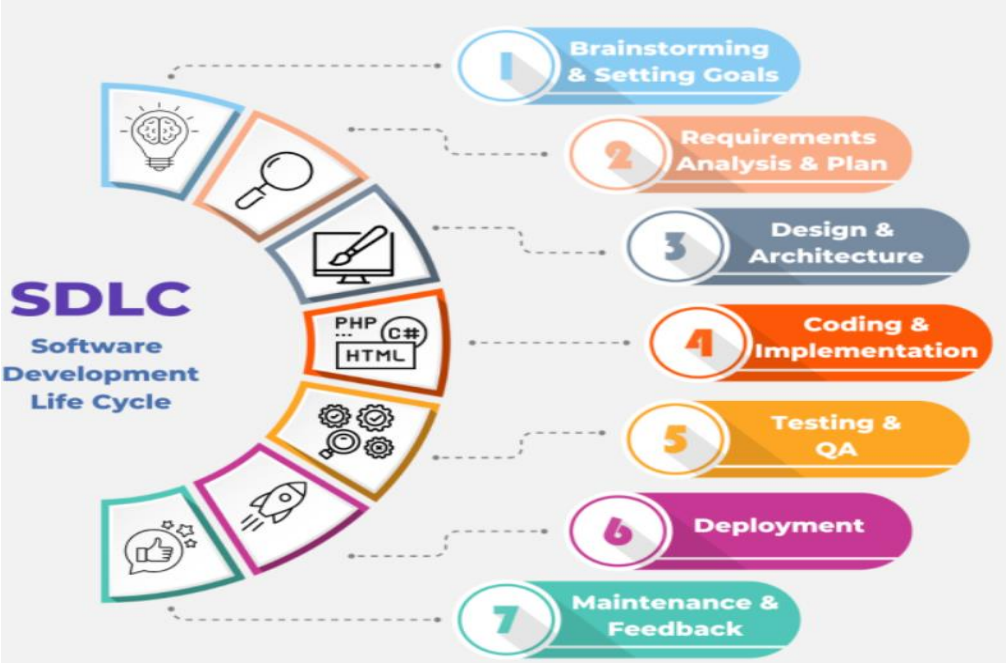
vidyasagar.bandaluppi@gmail.com

SDLC (Software Development Life Cycle)

A systematic approach that generates a structure for the developer to design, create and deliver high-quality software based on customer requirements and needs.

The primary goal of the SDLC process is to produce cost-efficient and high-quality products.

SDLC Phases :



Stage 1: Project Planning , Brain Stroming

- The first stage of SDLC is all about “What do we want?”
- Project planning is a vital role in the software delivery lifecycle since this is the part where the team estimates the cost and defines the requirements of the new software.
- Brainstorming marks the outset of the software development life cycle, when teams join together to explore possibilities and conceive inventive solutions.

vidyasagar.bandaluppi@gmail.com

Stage 2: Gathering Requirements & Analysis

- The second step of SDLC is gathering maximum information from the client requirements for the product.
- Discuss each detail and specification of the product with the customer.
- The development team will then analyze the requirements keeping the design and code of the software in mind.
- Further, investigating the validity and possibility of incorporating these requirements into the software system.
- The main goal of this stage is that everyone understands even the minute detail of the requirement.
- Hardware, operating systems, programming, and security are to name the few requirements.

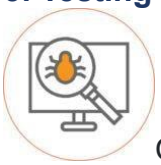
Stage 3: Design

- In the design phase (3rd step of SDLC), the program developer scrutinizes whether the prepared software fulfils all the requirements of the end-user.
- Additionally, if the project is feasible for the customer technologically, practically, and financially.
- Once the developer decides on the best design approach, he then selects the program languages like Java, python etc., that will suit the software.
- Once the design specification is prepared, all the stakeholders will review this plan and provide their feedback and suggestions.
- It is absolutely mandatory to collect and incorporate stakeholder's input in the document, as a small mistake can lead to cost overrun.

Stage 4: Coding or Implementation

- Time to code! It means translating the design to a computer-legible language.
- In this fourth stage of SDLC, the tasks are divided into modules or units and assigned to various developers.
- The developers will then start building the entire system by writing code using the programming languages they chose.
- This stage is considered to be one of the longest in SDLC. The developers need certain predefined coding guidelines, and programming tools like interpreters, compilers, debugger to implement the code.
- The developers can show the work done to the business analysts in case if any modifications or enhancements required.

Stage 5: Testing



- Once the developers build the software, then it is deployed in the testing environment.
- Then the testing team tests the functionality of the entire system.
- In this fifth phase of SDLC, the testing is done to ensure that the entire application works according to the customer requirements.
- After testing, the QA and testing team might find some bugs or defects and communicate the same with the developers.
- The development team then fixes the bugs and send it to QA for a re-test.
- This process goes on until the software is stable, bug-free and working according to the business requirements of that system.

Stage 6: Deployment

- The sixth phase of SDLC: Once the testing is done, and the product is ready for deployment, it is released for customers to use.
- The size of the project determines the complexity of the deployment.
- The users are then provided with the training or documentation that will help them to operate the software.
- Again, a small round of testing is performed on production to ensure environmental issues or any impact of the new release.

Stage 7: Maintenance

- The actual problem starts when the customer actually starts using the developed system and those needs to be solved from time to time.
- Maintenance is the seventh phase of SDLC where the developed product is taken care of.
- According to the changing user end environment or technology, the software is updated timely.

---

vidyasagar.bandaluppi@gmail.com

Pre-code Planning:

- Pre-code planning plays an important role in creating an algorithm or a program, helping programmers plan how to write the codes.
- Pseudocode is one of the ways flowcharts can be helpful in programming.

Pseudo-code:

- Pseudocode is a false code used to describe how a computer program or an algorithm works.

- It uses annotations and text written in English because it's meant for humans to read instead of computers.

- Specifically, it consists of statements in English, selected keywords, and mathematical notations.

- One can make use of pseudo code to present the implementation of an algorithm.

Eg.

The information needed for pre-code planning includes the following:

- Input: Student grades
- Output: The average grade
- Processing: Sum the grades, find the total student count, calculate the average grade.

vidyasagar.bandaluppi@gmail.com

```
Initialize Counter and Sum to 0
Do While there are more data
    Get the next Grade
    Add the Grade to the Sum
    Increment the Counter
Loop
Computer Average = Sum / Counter
Display Average
```

Verifying an algorithm:

- An important aspect of any algorithm is that it is **correct**: it always produces the expected output for the range of inputs and it eventually terminates.
- As it turns out, it's difficult to prove that an algorithm is correct. Programmers often use **empirical analysis** to find faults in an algorithm, but only **formal reasoning** can prove total correctness.

Empirical analysis

- An "empirical" analysis is one based on actual experimentation and observation of the results.
- In the world of algorithms, that means the algorithm must actually be translated into a programming language and executed on a computer.
- Let's conduct an empirical analysis of an algorithm that finds the maximum value in a list of numbers.
- Here's pseudocode that expresses that algorithm:

```
maxNum ← -1
FOR num IN numbers {
 if (num > maxNum) {
  maxNum ← num
```

```
 }
}
```

- Next, we'll translate that into the JavaScript language, for example:

```
var maxNum = -1;

for (var i = 0; i < numbers.length; i++) { // repeats loop for length(size) no. of times
 if (numbers[i] > maxNum) {
   maxNum = numbers[i];
 }
}
```

  - Then we need to feed input into the algorithm and observe how it performs.
  - For our first experiment, let's give it an array of 4 numbers, [13, 4, 24, 7] and see if it outputs the max, 24.

```
1  var numbers = [13, 4, 24, 7];
2  var maxNum = -1;
3  for (var i = 0; i < numbers.length; i++) {
4    if (numbers[i] > maxNum) {
5      maxNum = numbers[i];
6    }
7  }
8  println(maxNum);
9
```

  - Hooray, it worked!
  - Can we declare this to be a perfectly correct algorithm and move on with life?
  - It's not quite that easy...

It's time for experiment #2.

- This time, let's make all the numbers in the array negative, [-13, -4, -24, -7].
- This time, the maximum value should be -4, the smallest negative number in the list.

```
var numbers = [-13, -4, -24, -7];
var maxNum = -1;
for (var i = 0; i < numbers.length; i++) {
  if (numbers[i] > maxNum) {
    maxNum = numbers[i];
  }
}
println(maxNum);
```

- Uh-oh, the code outputted -1 instead of -4.
- That's because the initial value for maxNum is -1, and the loop never finds a value in the list greater than that.
- Our algorithm definitely does *not* work correctly for negative numbers.
- At this point, we need to modify our algorithm and conduct empirical analysis on the (hopefully) improved algorithm.
- Let's try out a version of the algorithm that initializes maxNum to the first number in the list:

```
var numbers = [-13, -4, -24, -7];
var maxNum = numbers[0];
for (var i = 0; i < numbers.length; i++) {
  if (numbers[i] > maxNum) {
    maxNum = numbers[i];
  }
}
println(maxNum);
```

That works!

---

vidyasagar.bandaluppi@gmail.com

Flowchart:

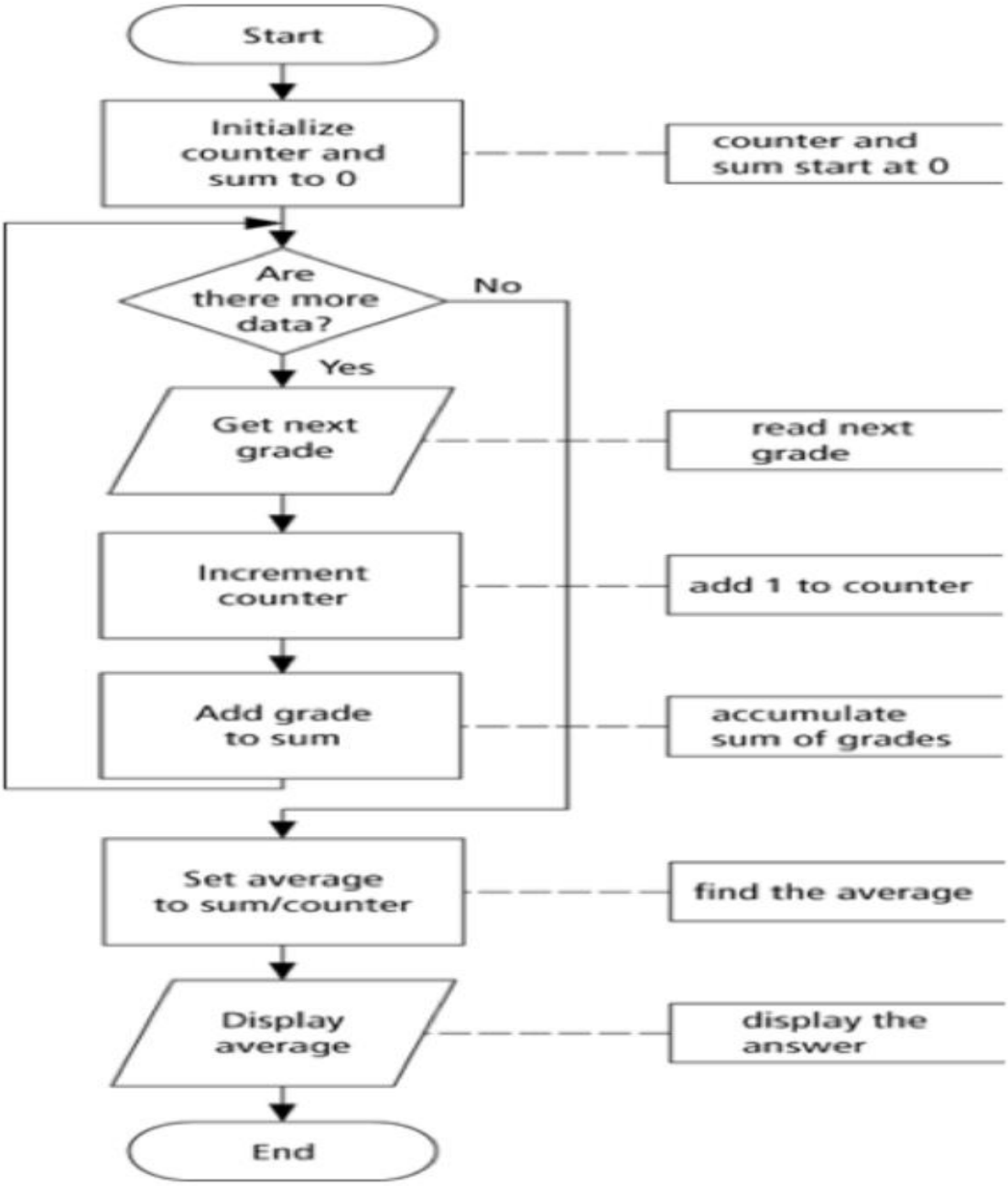
- A flowchart is a type of diagrammatic representation using shapes and flow lines to illustrate a computer program, an algorithm, or a process.

- The process symbol used in a flow chart includes ovals, rectangles, diamonds, flow lines, and more to indicate various types of steps.

Eg. The question that needs solving is calculating the average grade-point of all students in a class.

The flowchart is shown in this picture.

vidyasagar.bandaluppi@gmail.com



---

vidyasagar.bandaluppi@gmail.com