Compiler and SDLC

Dr. Shweta Sharma
Assistant Professor
Dept. of Computer Engineering
NIT Kurukshetra

Compiler

SOURCE CODE

- **Source code** is the term given to a set of instructions that are written in human readable programming language

- Source code must be translated into machine code before a computer can understand and execute it

TRANSLATORS

- Any **program** written in a **high-level language** is known as **source code**. However, computers cannot understand source code. Before it can be run, source code must first be translated into a form which a computer understands.
- A **translator** is a program that converts source code into machine code. Generally, there are three types of translator:
  - **Compilers**
  - **Interpreters**
  - **Assemblers**

COMPILERS

- A compiler takes the source code as a whole and translates it into machine code all in one go

- Once converted, the object code can be run unassisted at any time. This process is called **compilation**.

COMPILERS

Compilers have several advantages:

- Compiled programs run quickly, since they have already been translated.
- A compiled program can be supplied as an **executable** file. An executable file is a file that is ready to run. Since an executable file cannot be easily modified, programmers prefer to supply executable rather than source code.
- Compilers **optimise** code. Optimised code can run quicker and take up less **memory** space.

COMPILERS

Compilers also have disadvantages:

- The source code must be re-compiled every time the programmer changes the program.
- Source code compiled on one platform will not run on another - the machine code is specific to the processor's architecture.

INTERPRETERS

- An interpreter translates source code into machine code one **instruction** at a time
- It is similar to a human translator translating what a person says into another language, sentence by sentence, as they speak
- The resulting machine code is then executed immediately. The process is called **interpretation**

INTERPRETERS

Interpreters have several advantages:

- Instructions are executed as soon as they are translated.
- Errors can be quickly spotted - once an error is found, the program stops running and the user is notified at which part of the program the interpretation has failed. This makes interpreters extremely useful when developing programs.

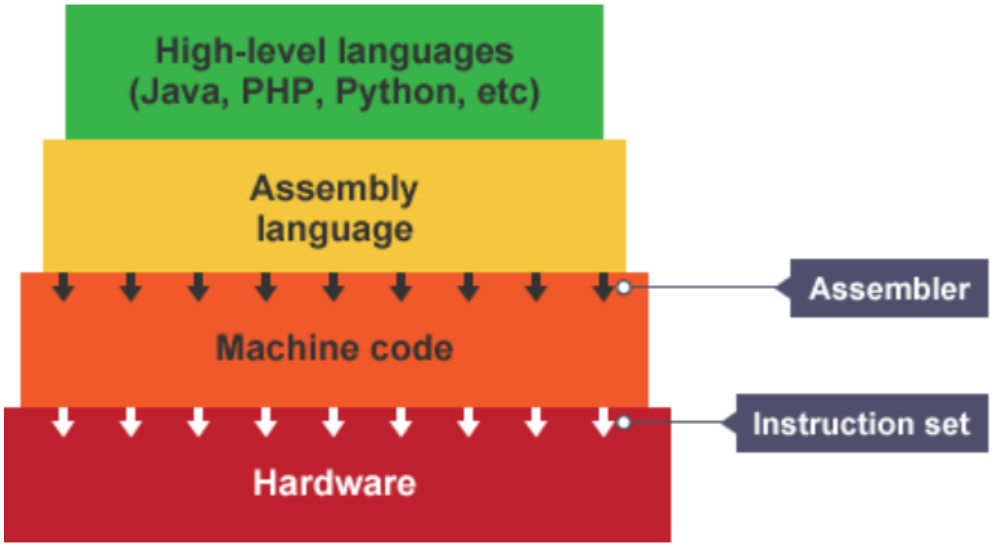
INTERPRETERS

Interpreters also have several disadvantages:

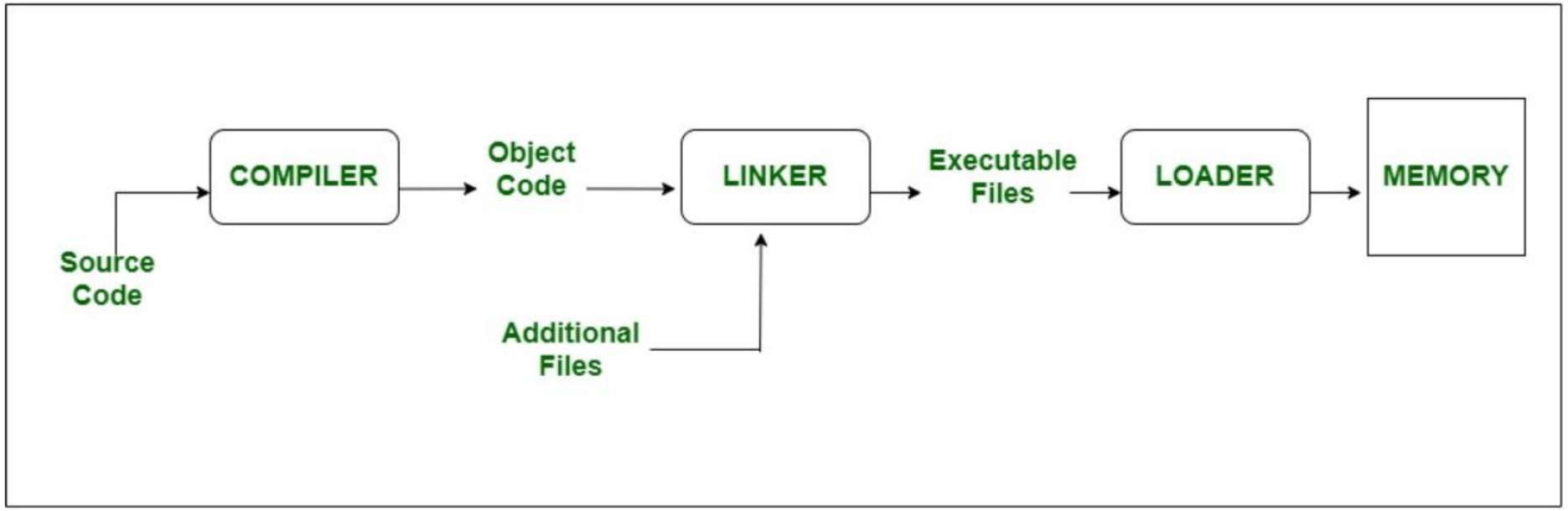
- Interpreted programs run slowly as the processor has to wait for each instruction to be translated before it can be executed
- Additionally, the program has to be translated every time it is run
- Interpreters do not produce an executable file that can be distributed. As a result, the source code program has to be supplied, and this could be modified without permission
- Interpreters do not optimise code - the translated code is executed as it is

ASSEMBLERS

- Assemblers are a third type of translator
- The purpose of an assembler is to translate **assembly language** into machine code
- Whereas compilers and interpreters generate many machine code instructions for each high-level instruction, assemblers create one machine code instruction for each assembly instruction



LINKER AND LOADER



LINKER AND LOADER

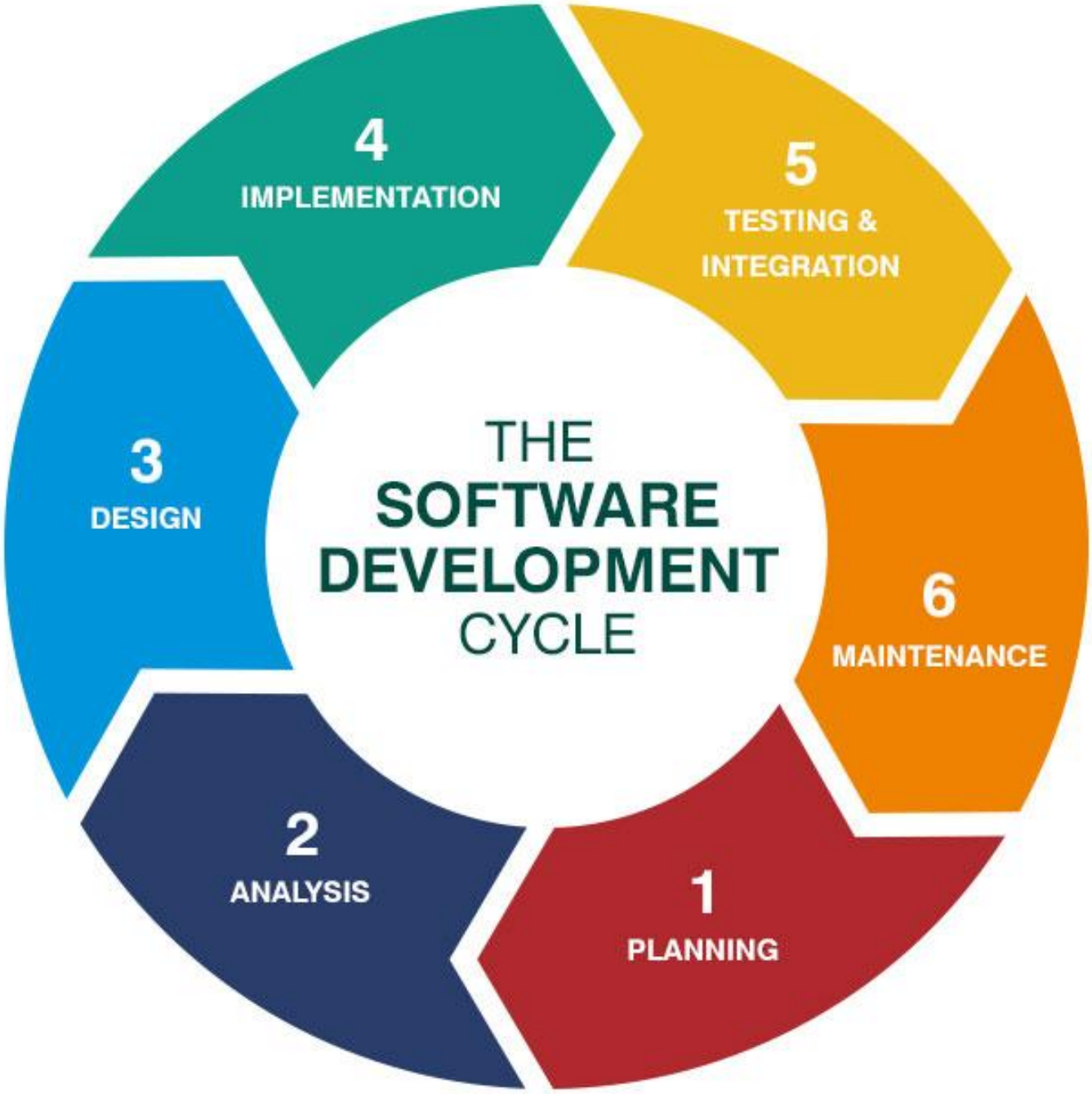
- **Linker** or link editor
- A computer utility program that takes one or more object files generated by a compiler and combines them into a single executable file, library file, or another 'object' file

LINKER AND LOADER

- **Loader** is the part of an operating system that is responsible for loading programs and libraries
- Loader allocates memory space to program
- It is one of the essential stages in the process of starting a program, as **it places programs into memory and prepares them for execution**
- It is a special program that takes input of executable files from linker, loads it to main memory, and prepares this code for execution by computer

Software Development Life Cycle (SDLC)

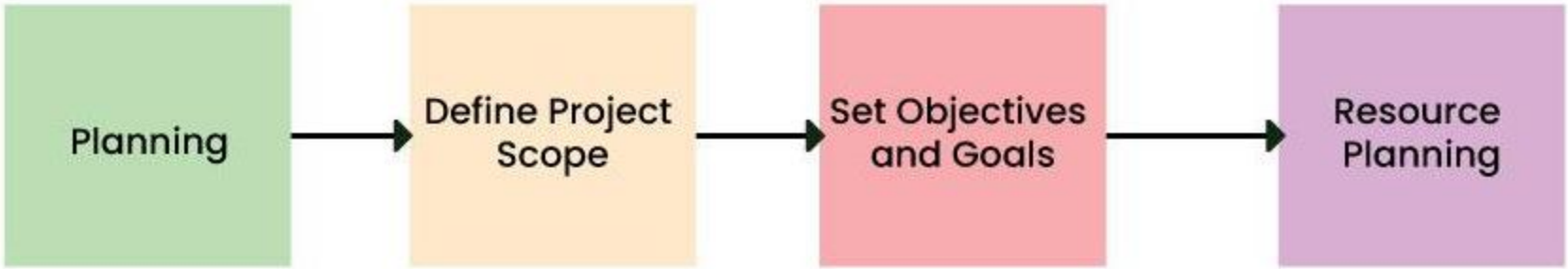
Methodology that defines the entire procedure of software development step-by-step


4
IMPLEMENTATION
5
TESTING &
INTEGRATION
3
DESIGN
THE
SOFTWARE
DEVELOPMENT
CYCLE
6
MAINTENANCE
2
ANALYSIS
1
PLANNING

SDLC

Stage-1: Planning:

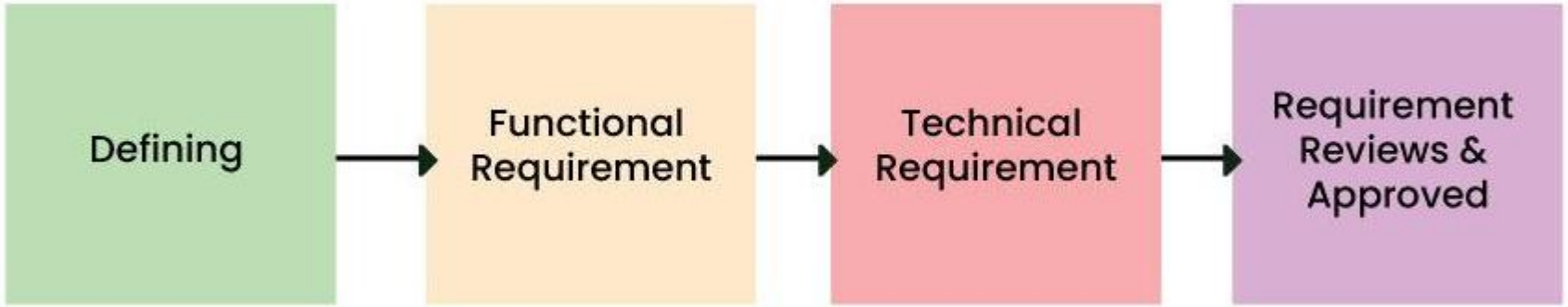
Team estimates cost, creates a schedule, and has a detailed plan to achieve the goals



SDLC

Stage-2: Analysis:

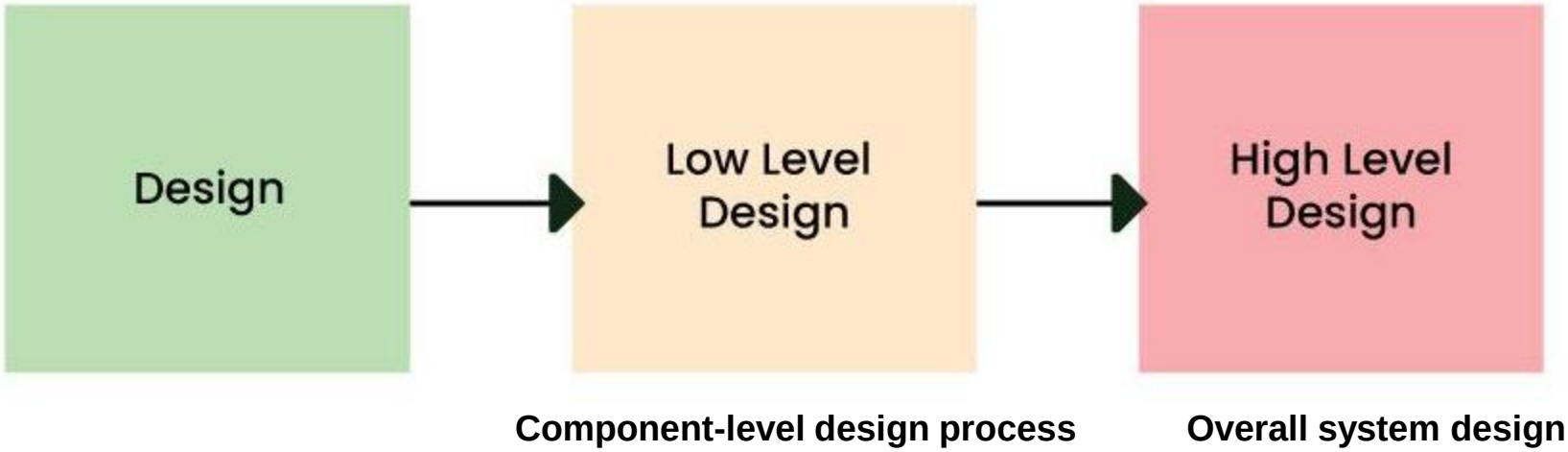
Utilizing SRS (Software Requirement Specification)
A document that specifies all those things that need to be defined and created during the entire project cycle



SDLC

Stage-3: Design:

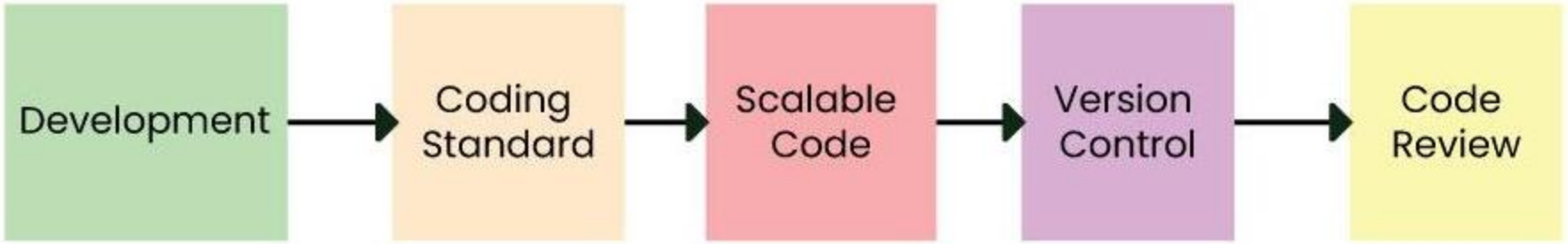
Analyze requirements and identify the best solutions to create the software



SDLC

Stage-4: Implementation:

The development team codes the product
This phase often requires extensive programming skills and knowledge of databases



SDLC

Stage-5: Testing & Integration:

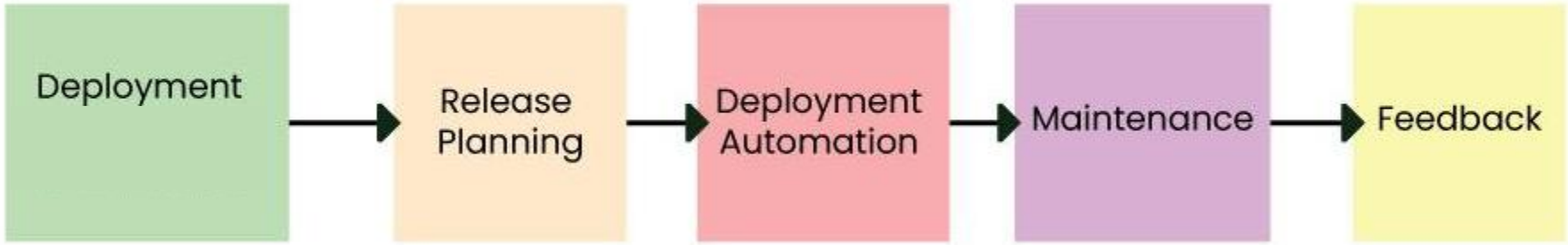
Testing the software for errors and checking if it meets customer requirements

Testing phase often runs parallel to the development phase

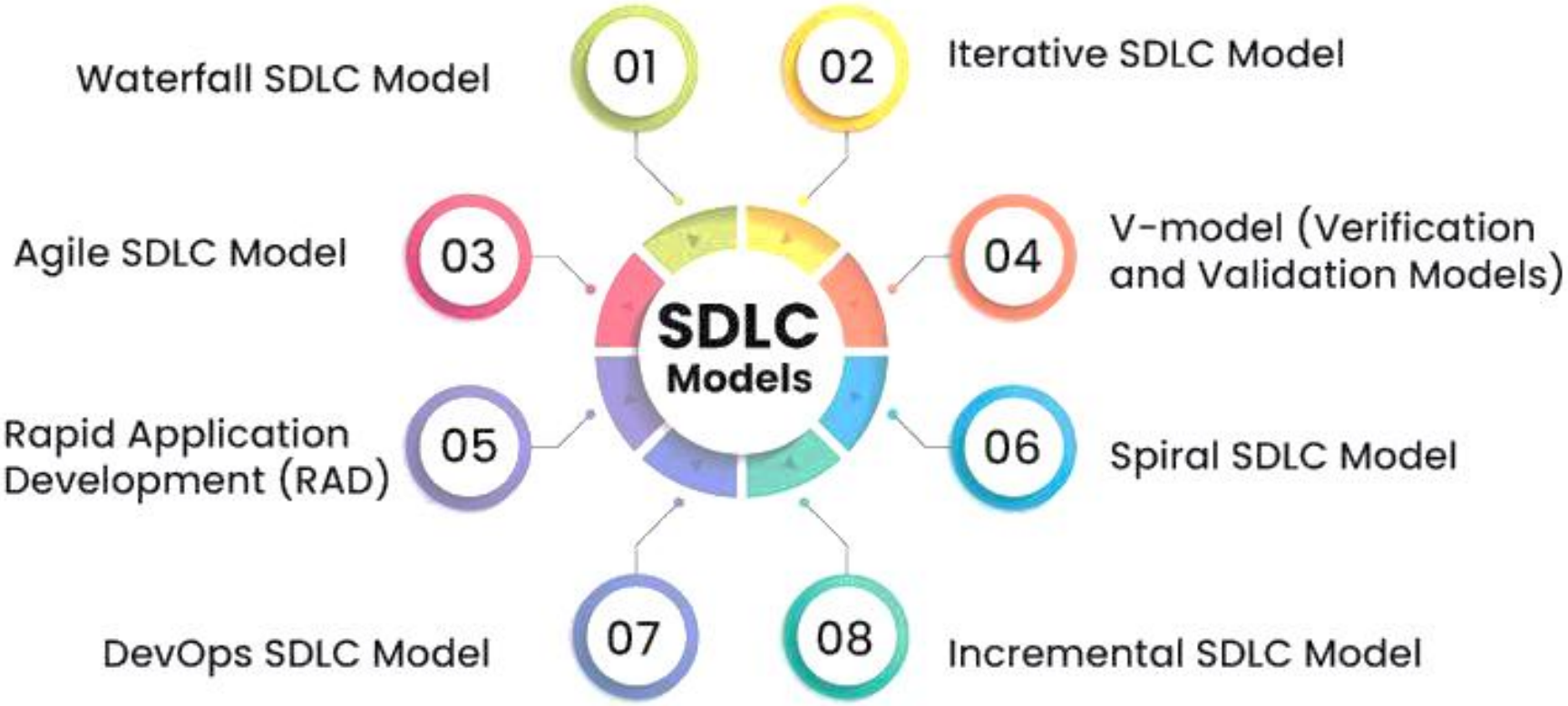
SDLC

Stage-6: Maintenance:

The team fixes bugs, resolves customer issues, and manages software changes
updating an existing software product to fix bugs and ensure reliability
It can also include adding new features or functionality to a current product



SDLC Models


SDLC
Models
01
Waterfall SDLC Model
02
Iterative SDLC Model
03
Agile SDLC Model
04
V-model (Verification and Validation Models)
05
Rapid Application Development (RAD)
06
Spiral SDLC Model
07
DevOps SDLC Model
08
Incremental SDLC Model

Thank You!