THE SOFTWARE DEVELOPMENT PROCESS

the Software Development Life Cycle (SDLC)

SDLC

- In simpler words, Software Development Life Cycle (SDLC) is the set of processes used to develop and deliver high quality software
- It’s recommended to follow Software Development Life Cycle when an IT project is under development
- Its beneficial to follow SDLC when you need to combine technical and non-technical activities to deliver high quality software
- The Software Development Life Cycle is structured in different stages in the life cycle and can be accomplished using different models

WHAT IS SOFTWARE DEVELOPMENT LIFE CYCLE?

Software Development Life Cycle is generally termed as SDLC and it is the conceptual framework which clearly defines what tasks must be performed at each stage and by whom, within scheduled timeframe and at operational cost

WHY IS SOFTWARE DEVELOPMENT LIFE CYCLE NECESSARY?

Software Development Life Cycle is needed in any of the project for the following reasons:

- Enhance the quality of the software
- Define the goals to the team so that developers know what to build and testers know what and how to test
- Reduce the rate of vulnerabilities (fewer or none)
- Management control
- Effective documentation and reduced dependencies

WHY IS SOFTWARE DEVELOPMENT LIFE CYCLE NECESSARY?

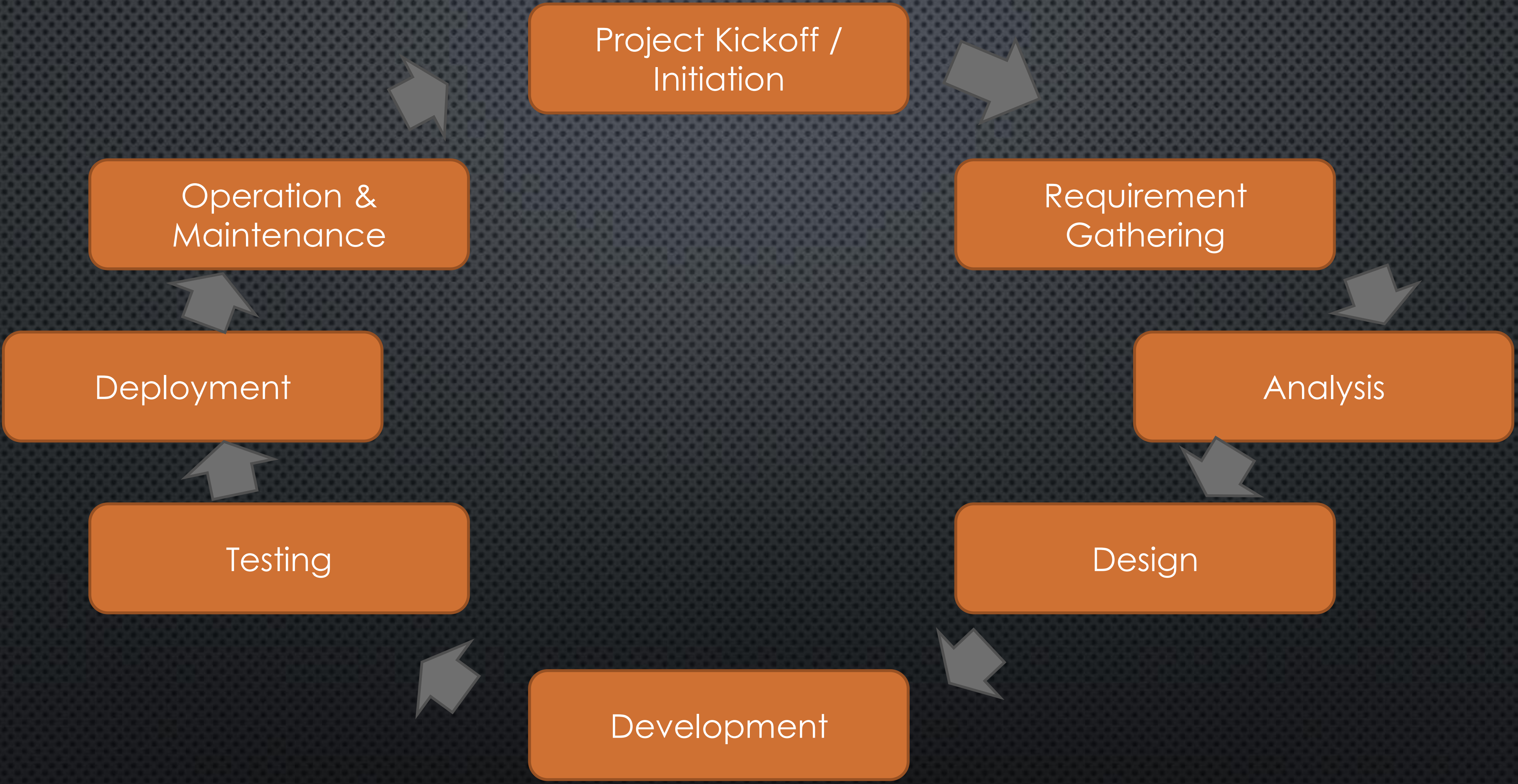
- Effective resource utilization
- Effective cost and time definition
- Ensure the architecture and design are secure
- Define and adhere to the set processes and objectives
- Meet and exceed Customer’s expectations

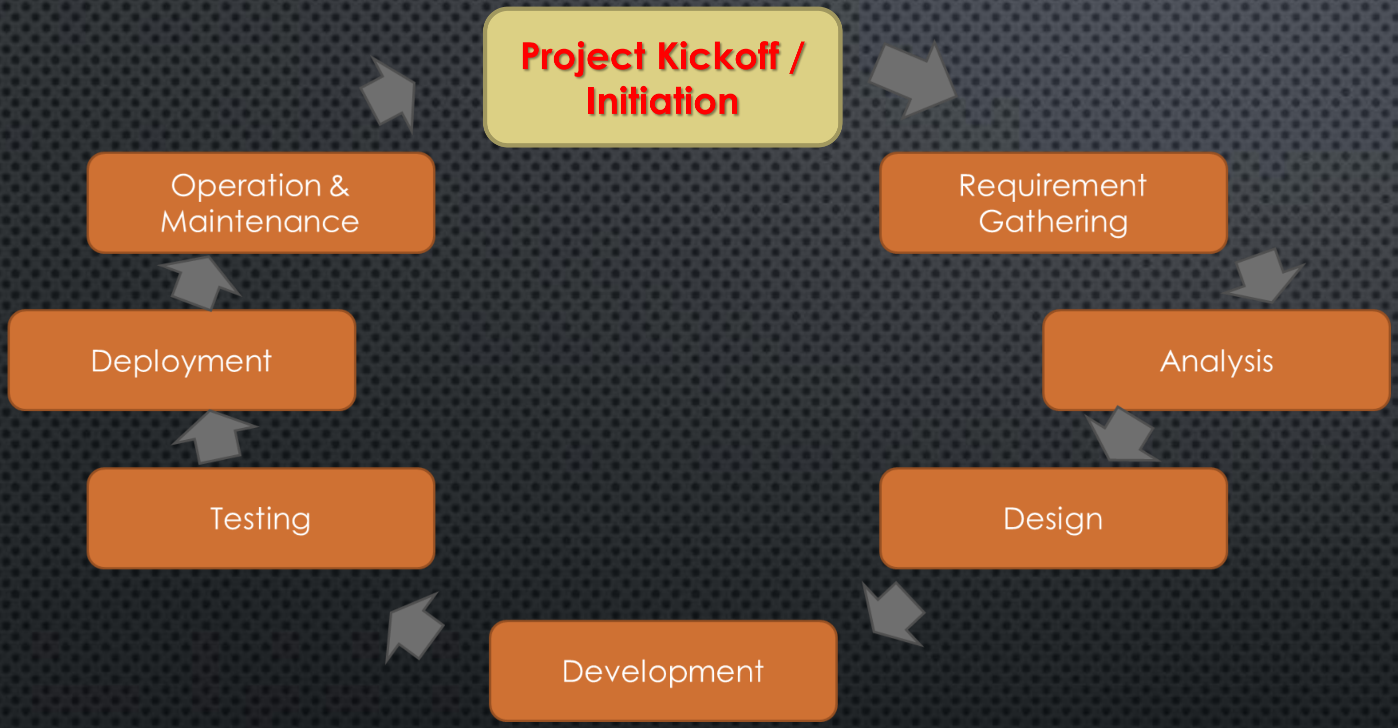
STAGES OF SOFTWARE DEVELOPMENT LIFE CYCLE

- While the term SDLC may indicate that its closely associated with development, a **good software tester** should have sound knowledge of this concept since it will be beneficial in their **software testing career path**
- Right from inception to delivering high-quality software, any software development project follows the series of tasks in the predefined manner
- These tasks are grouped and categorized into stages and the series of these stages forms the Software Development Life Cycle

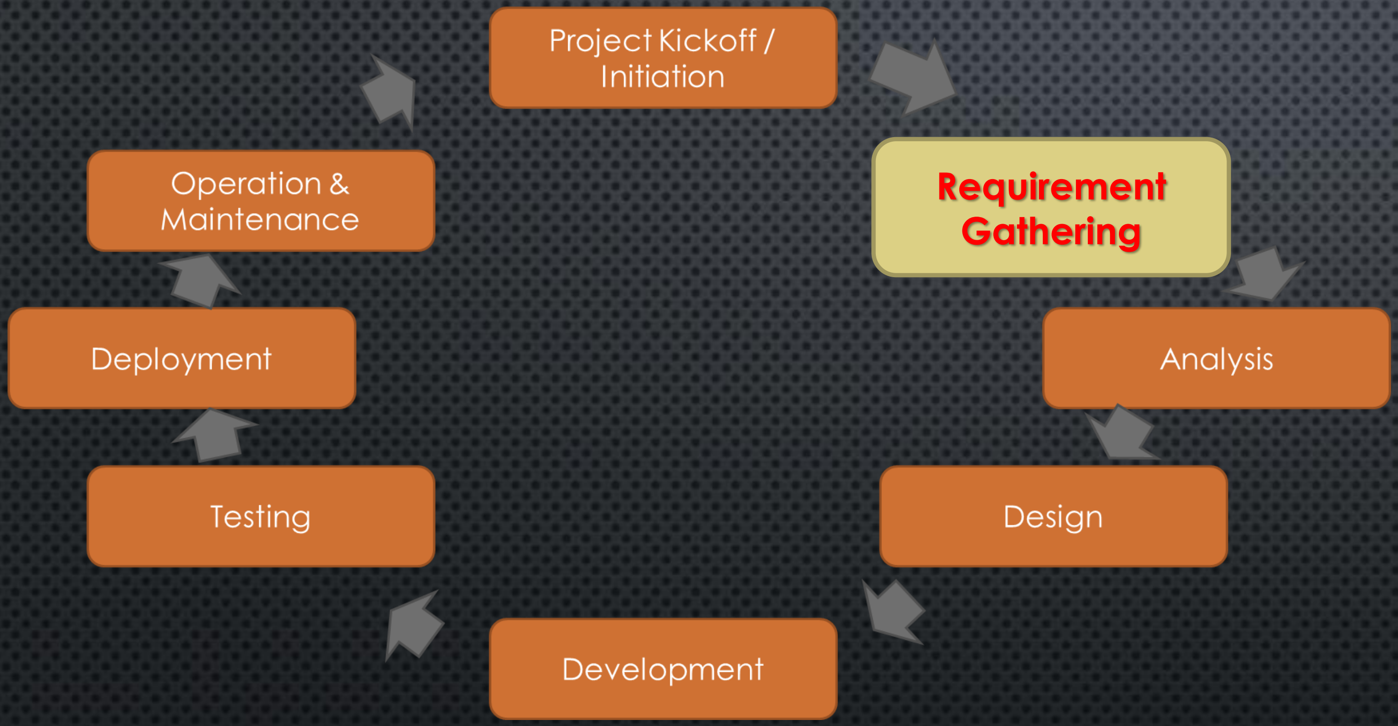
STAGES OF SOFTWARE DEVELOPMENT LIFE CYCLE

- At each of the stage in the Software Development Life Cycle, the responsibilities of the tasks are predefined to the specific roles
- Software Development Life Cycle has many **models** and the project can choose any among them based on the strategies, schedules, complexity and many more business factors
- Note: Testing follows a life cycle called **Software Testing Life Cycle (STLC)**, which is similar to the SDLC but specific to the testing process
- All the Software Development Life Cycle models have almost all the below described stages

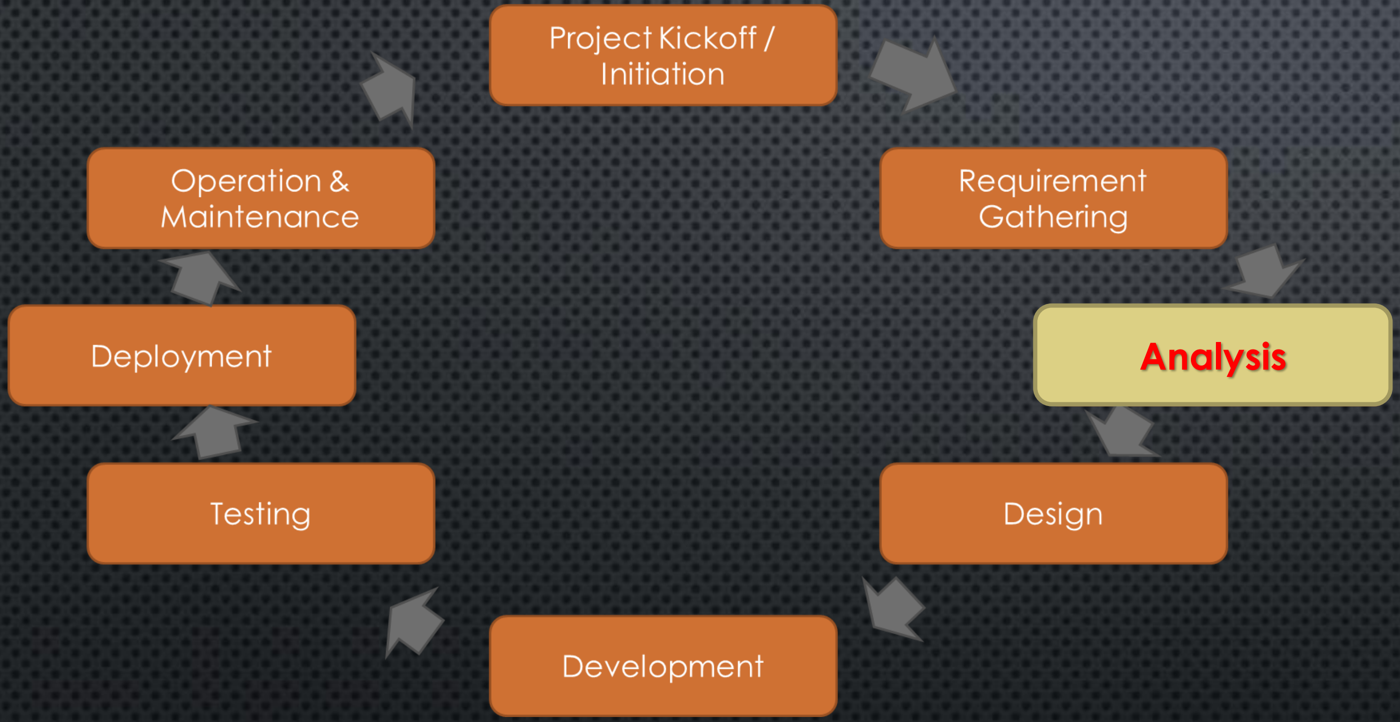

Project Kickoff / Initiation
Requirement Gathering
Analysis
Design
Development
Testing
Deployment
Operation & Maintenance



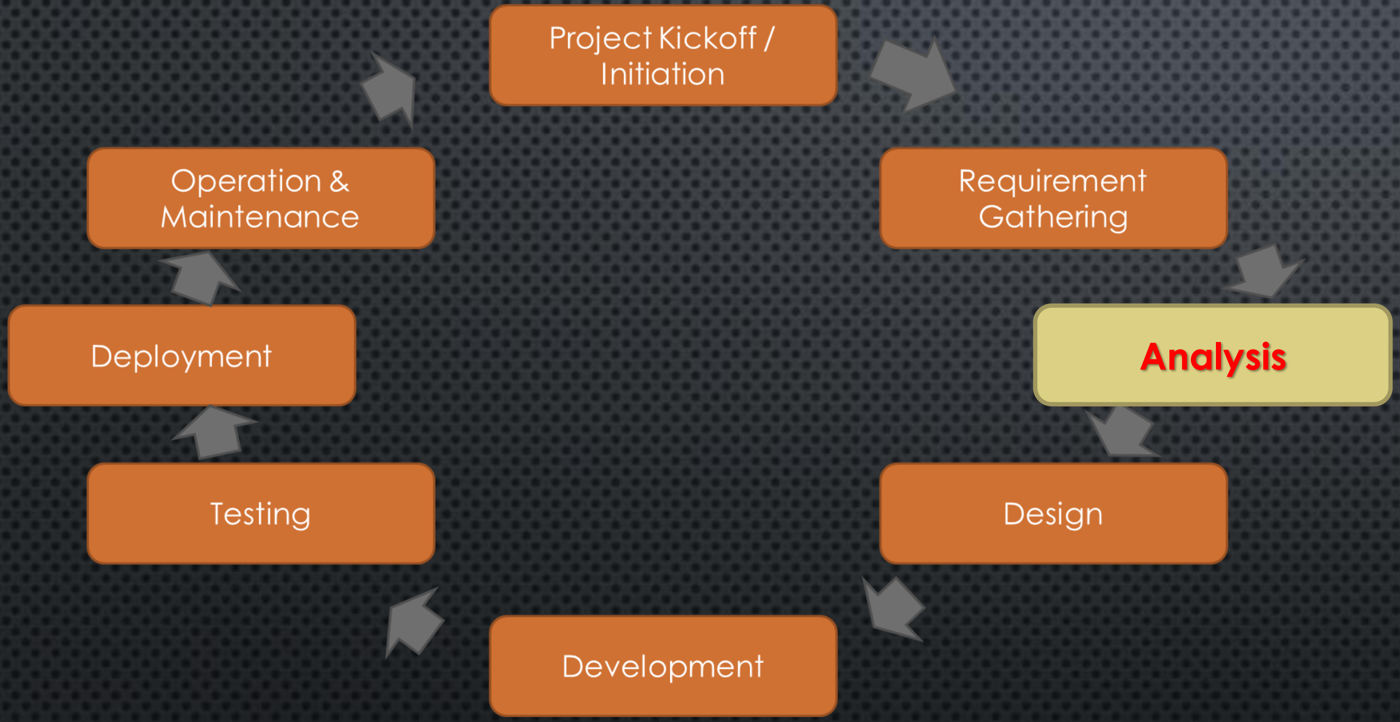
- This is the first stage in Software Development Life Cycle where the project is initiated
- The high level scope, problems and solutions are determined and planning is carried out accordingly for other stages
- Other components that are to be considered in this stage are: Resources, time / schedules, milestones, cost, business benefits and deadlines
- In case of enhancements to existing projects, the strengths and weaknesses of the current software is studied and the improvements are set as goal, along with the collected requirements



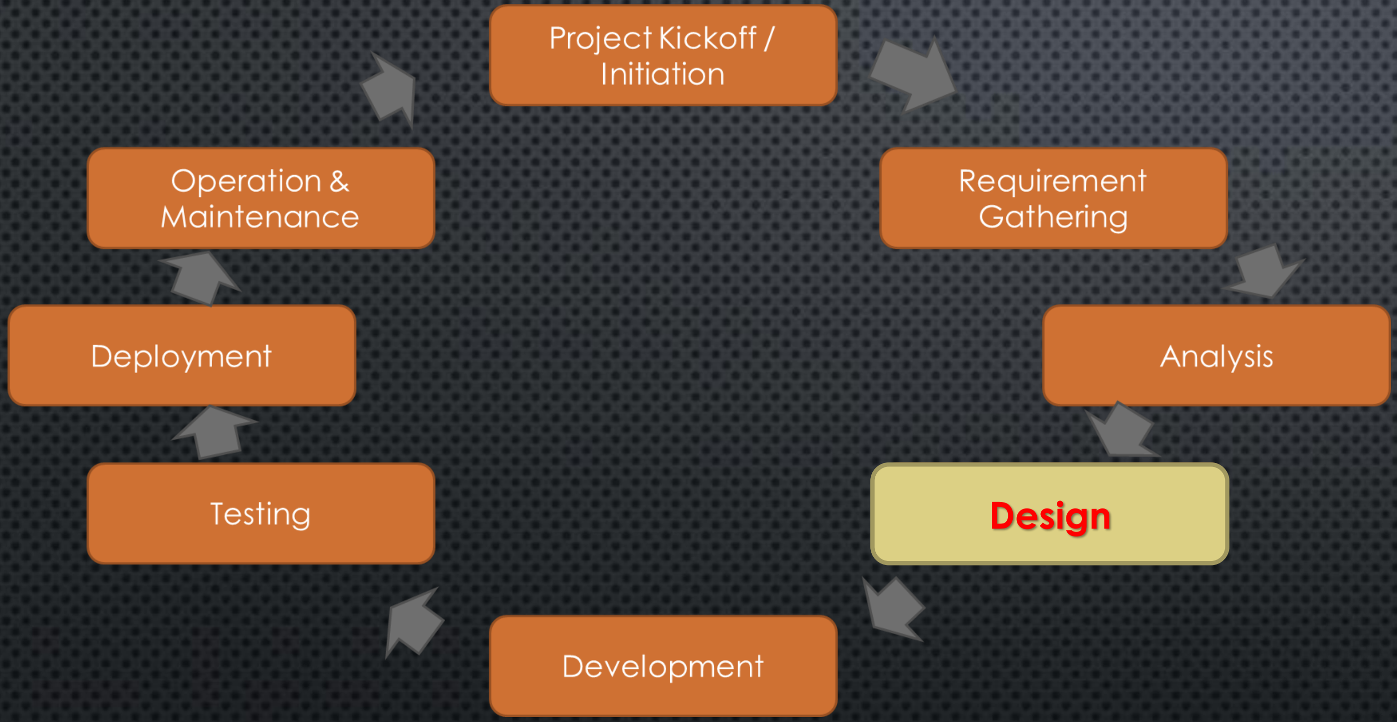
- Here two things are mainly focused:
  - *What is needed?*
  - *What is not needed?*
- In "What is needed?" part, the requirements are further analyzed to understand what are:
  - *Functional Requirements*
  - *Non-functional Requirements*
- End-user requirements from the customer and other stakeholders (sales people, domain / industry experts, etc.) are collected



- In this stage we analyze each and every achievable requirement.
- These are documented as Software Requirements Specifications (SRS) or Functional Requirements Specifications (FRS)
- Risks are predicted and the action items to mitigate those risks are well-planned at this stage
- Also, each and every user requirement are analyzed to ensure that they can be met
- This activity is termed also as System analysis, which helps in enhancing the way the software should behave
- Entire system is divided into smaller feasible chunks, so that the requirements can be prioritized and taken up for development in the order

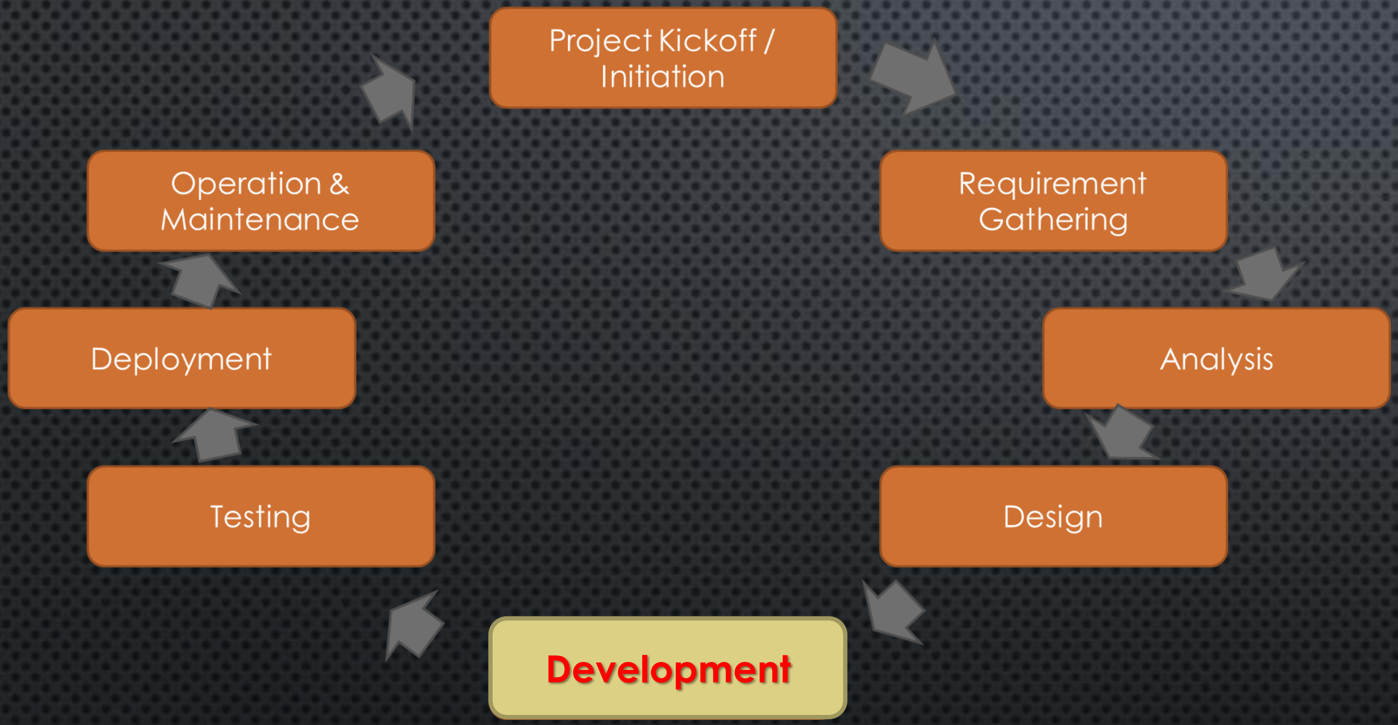


- This is effectively manageable for all the resources (developers, designers, testers, project managers and any other possible roles) to work on the chunks at all the stages in Software Development Life Cycle
- In many cases requirements gathering and analysis can be carried out at the same time



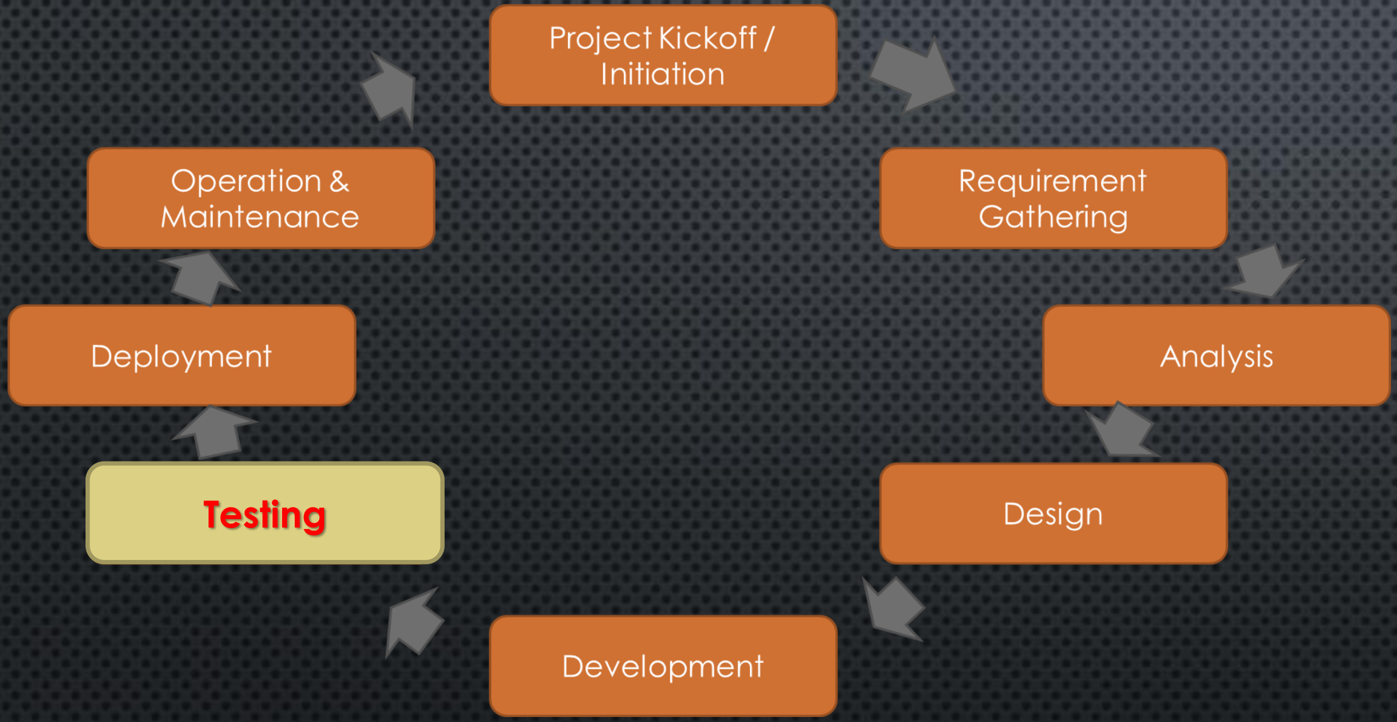
- This is the stage which states "How to achieve what is needed?"
- Software Requirements Specifications (SRS) are now converted to the system design plan, which is commonly known as "Design Specification"
- All the technical details like technologies to use, project constraints, team's capability, etc goes into the design specification document
- The technical architects and developers develop the logical plan of the system which is then reviewed by all the stakeholders
- The feedback and suggestions collected from the stakeholders are again incorporated into the already developed logical plan

The most challenging part in this stage is to ensure that the design has the tight security and has less or no exposure to vulnerabilities
If something goes wrong at this stage, it must be corrected at high priority, failing which has the higher rate of project failure and cost overruns

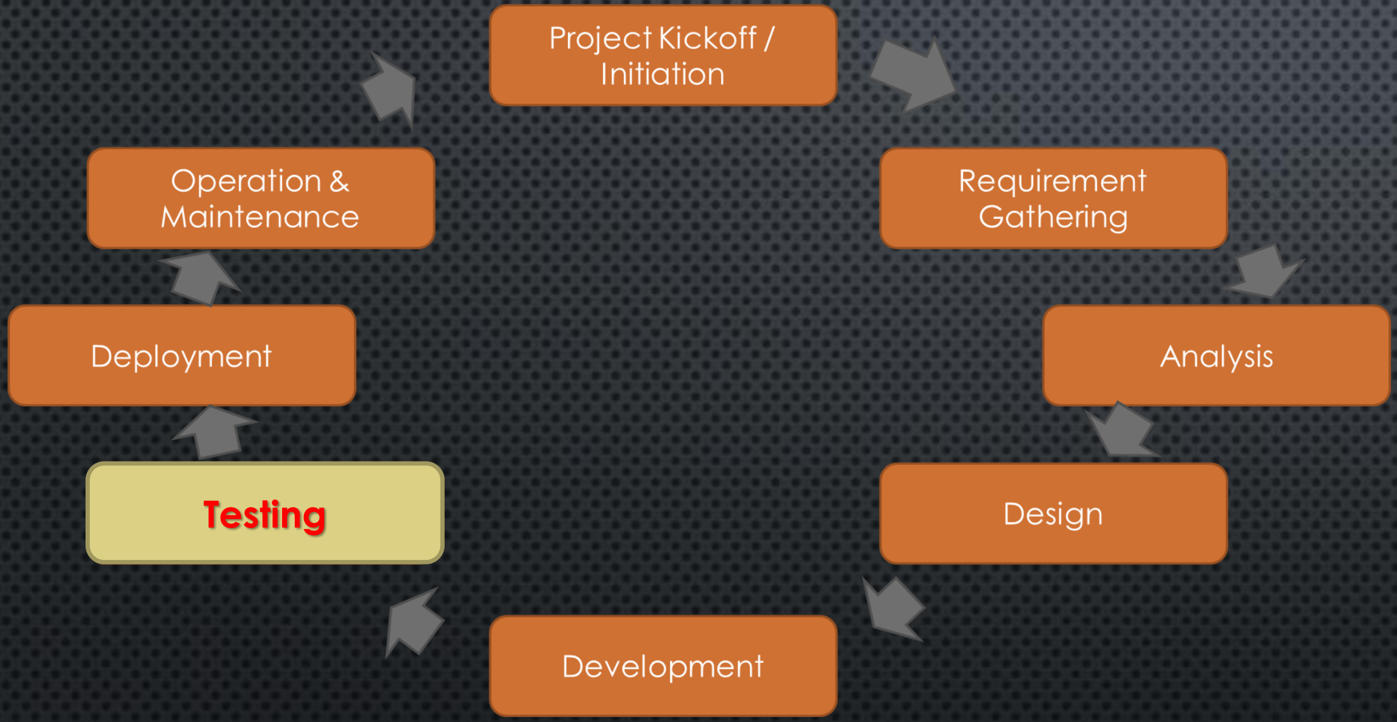


This stage in simpler terms is where the "real work begins" and we "build what is needed"

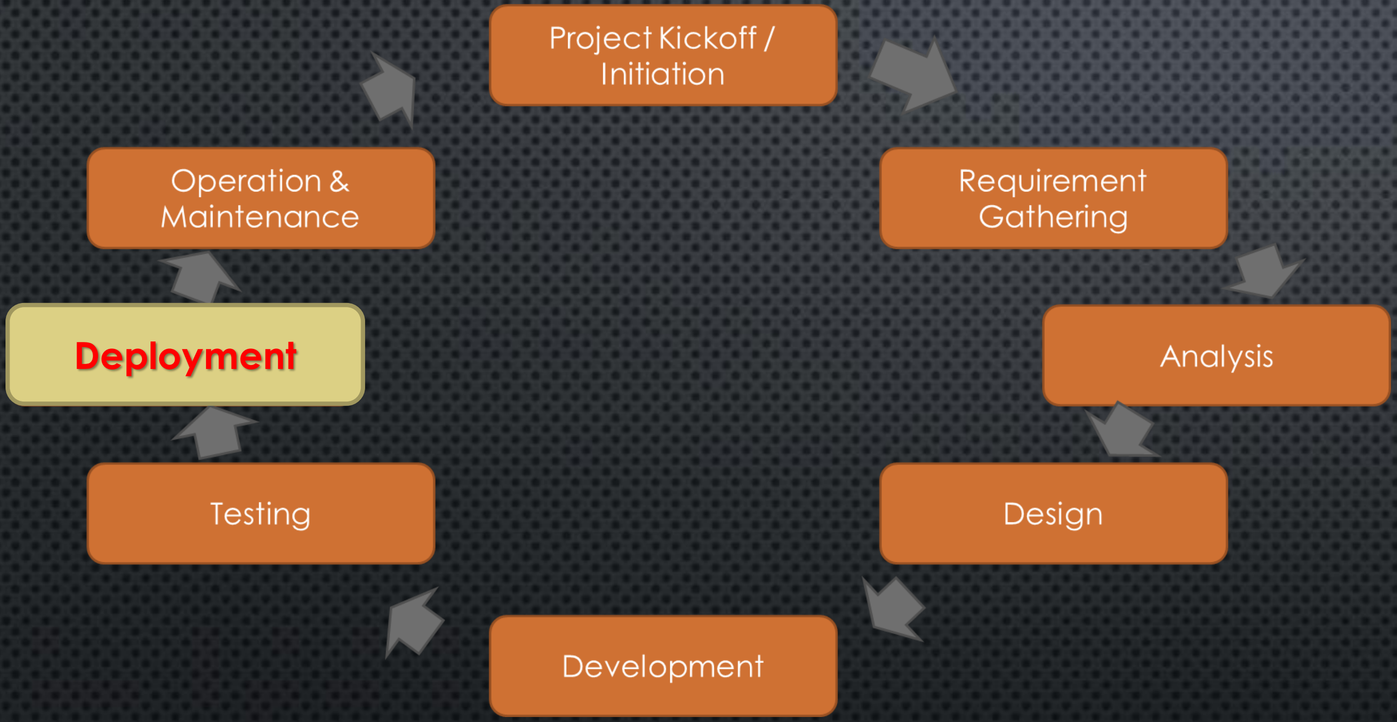
- The developers start to code as per the requirements and the developed design
- Along with the coding, all the other required set-up will begin. i.e., the database set up by database admin, interface and GUI creation by front-end developers, etc.
- Along with coding, it is also important for developers to develop unit tests for their module, peer review other module's unit tests, deploy builds to the intended environment and execute unit tests



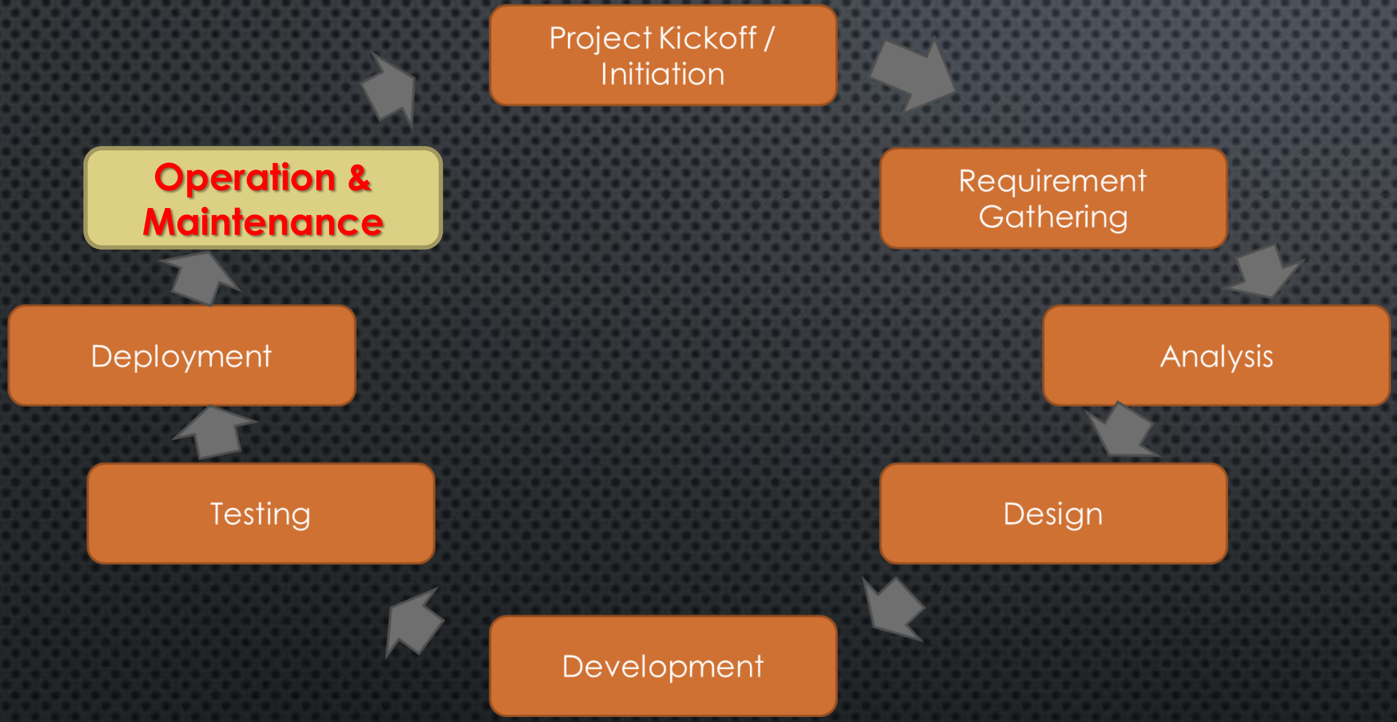
- This stage is the one where the quality check takes place. The developed software is assessed to ensure that all the specified requirements are met
- This is performed by testing team and the focus is to find the defects
- During test case execution, all the defects found are reported in the test management tool and the decision of considering the defect as Valid or Invalid depends on developers
- If the defect is Invalid, it is just rejected and closed
- If the defect is Valid, then it is fixed in the code by the developer and the code fix is provided in the next build for the tester to test whether the defect is fixed or not



- Each defect that is found will have to go through the Defect Life Cycle in the defect management tool
- Again, the testing approach that the project choose depends on various factors: complexity of the project, team's capability, time, etc.



- Once the testing is completed and there are no open high priority issues, then comes the time to deploy the build to the Production environment. This is the environment which is accessible by real users. Real users can then use the software as per their needs
- Deploying the build to production can be a complicated process. If the project is an existing application, technology migration is being carried out etc, it can be an extensive procedure
- Depending on business criticality deployment teams may need to ensure that the application continues to function, while the deployment is in progress
- Due to the high cut-over time, the Production deployment usually takes place during non-peak hours and / or weekends



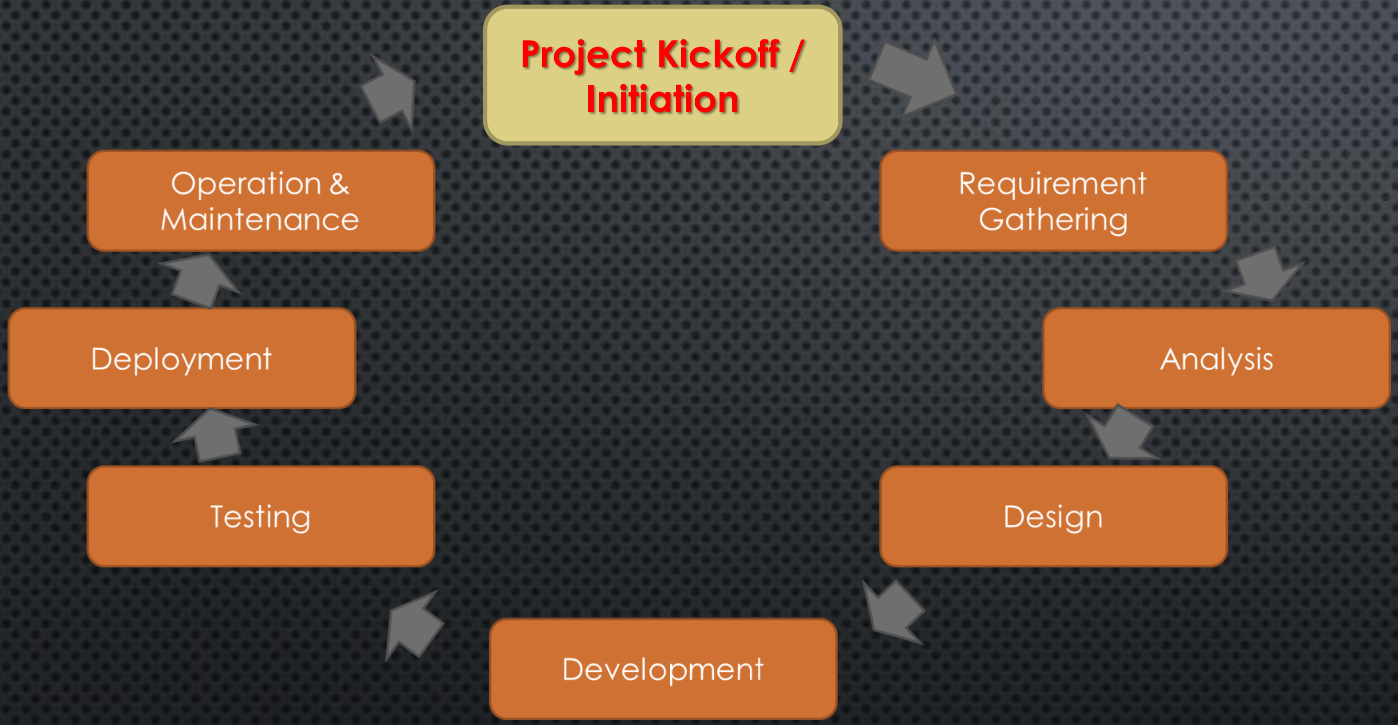
- This stage is when the "fine tuning" of the software takes place
- Once the build is deployed to Production environment, any issues that the real users face are considered as Post-Production issues
- These Post-Production issues are addressed and resolved by the internal team usually termed as Maintenance team
- This stage also addresses minor change requests, code fixes, etc. and deploys them in short intervals

ENTRY AND EXIT CRITERIA FOR EACH OF THE STAGES

- Each stage has specific set of entry and exit criteria
- Entry criteria are the conditions that are required to begin the processing of the current stage and exit criteria are the conditions which sets the stage as completed so that the next stage comes into action
- In general, the exit criteria of the current stage acts as entry criteria to the next stage
- Setting entry and exit criteria helps in determining whether the software development is in the right track and the entire team can focus on the tasks and conditions set for the stage

ENTRY AND EXIT CRITERIA FOR EACH OF THE STAGES

- It also enhances the effectiveness, efficiency and quality of the software to the greater extent, as the common goal to reach exit criteria of the stage is predefined
- Below is the table which at a high level mentions the entry and exit criteria for each of the stage in Software Development Life Cycle

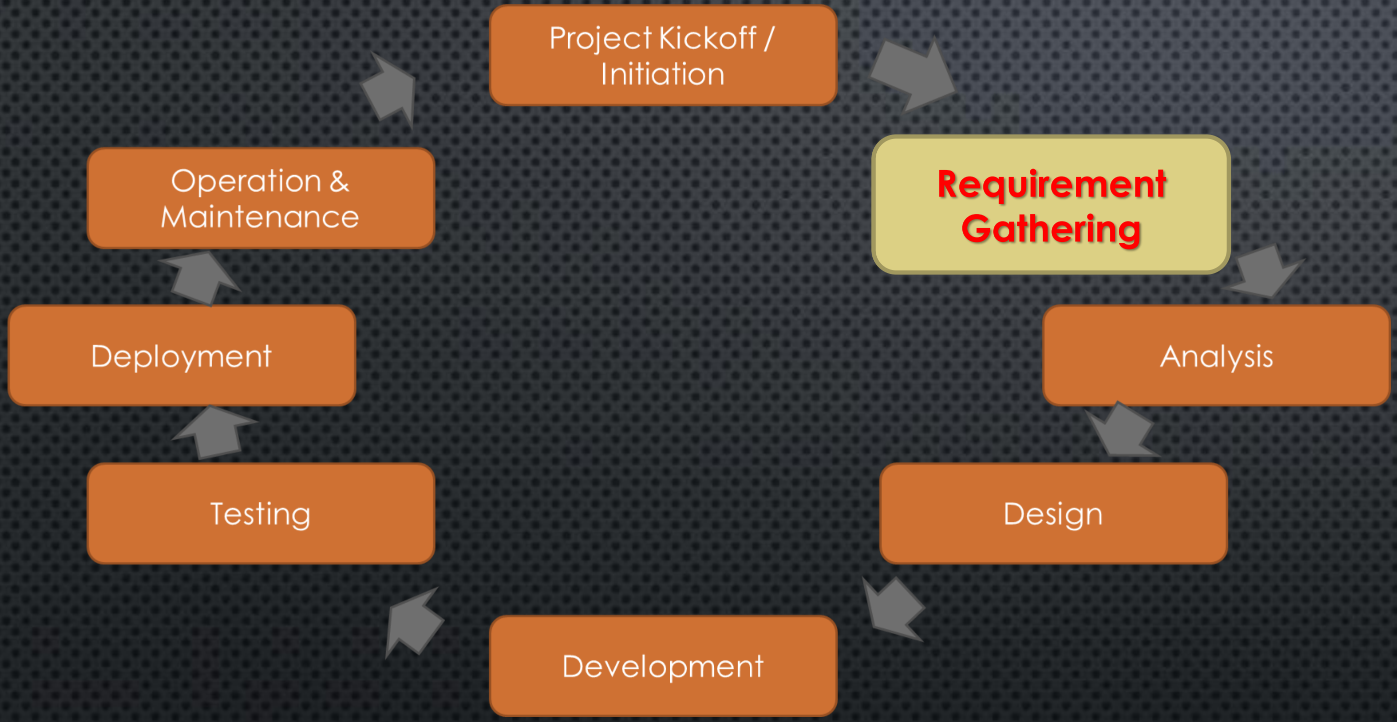


Entry Criteria

- Inputs / user-requirements collection from customers and other stakeholders

Exit Criteria

- Acceptance of the project and planning

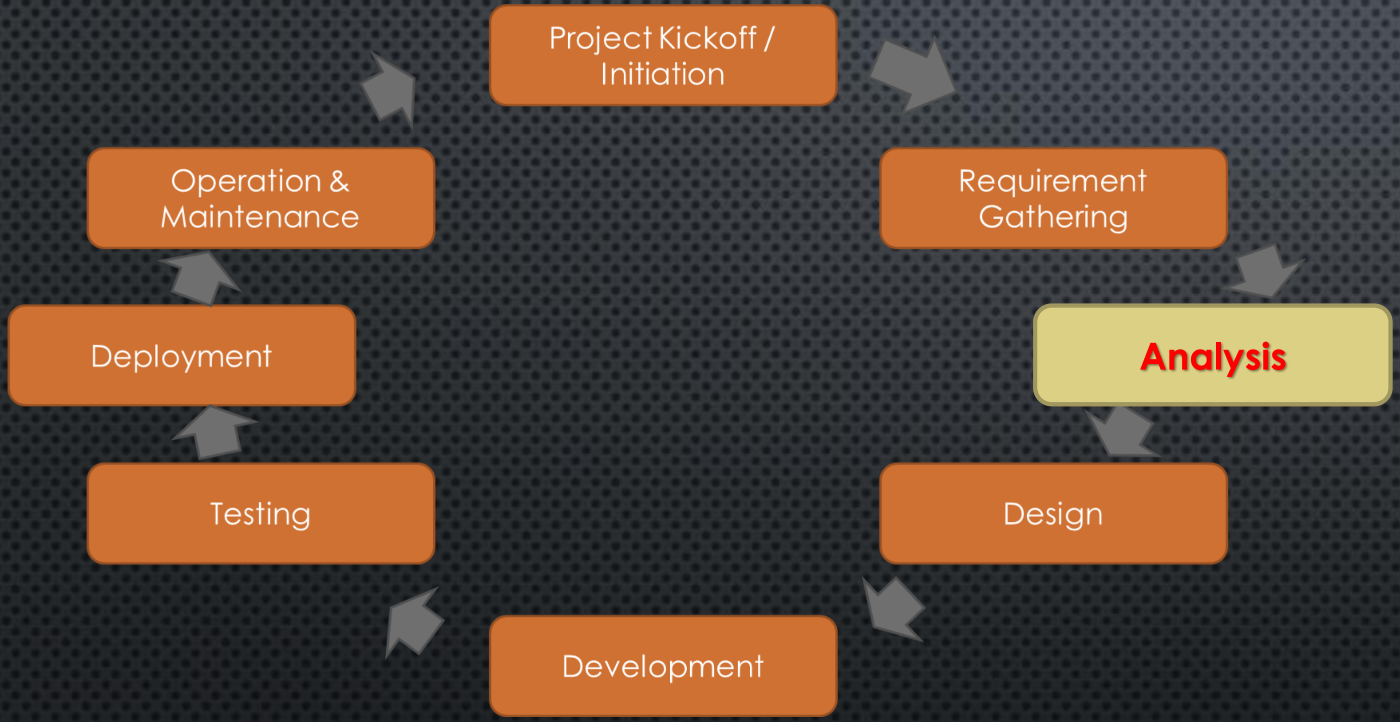


Entry Criteria

- Requirements
- Change requests

Exit Criteria

- Software Requirements Specifications Document

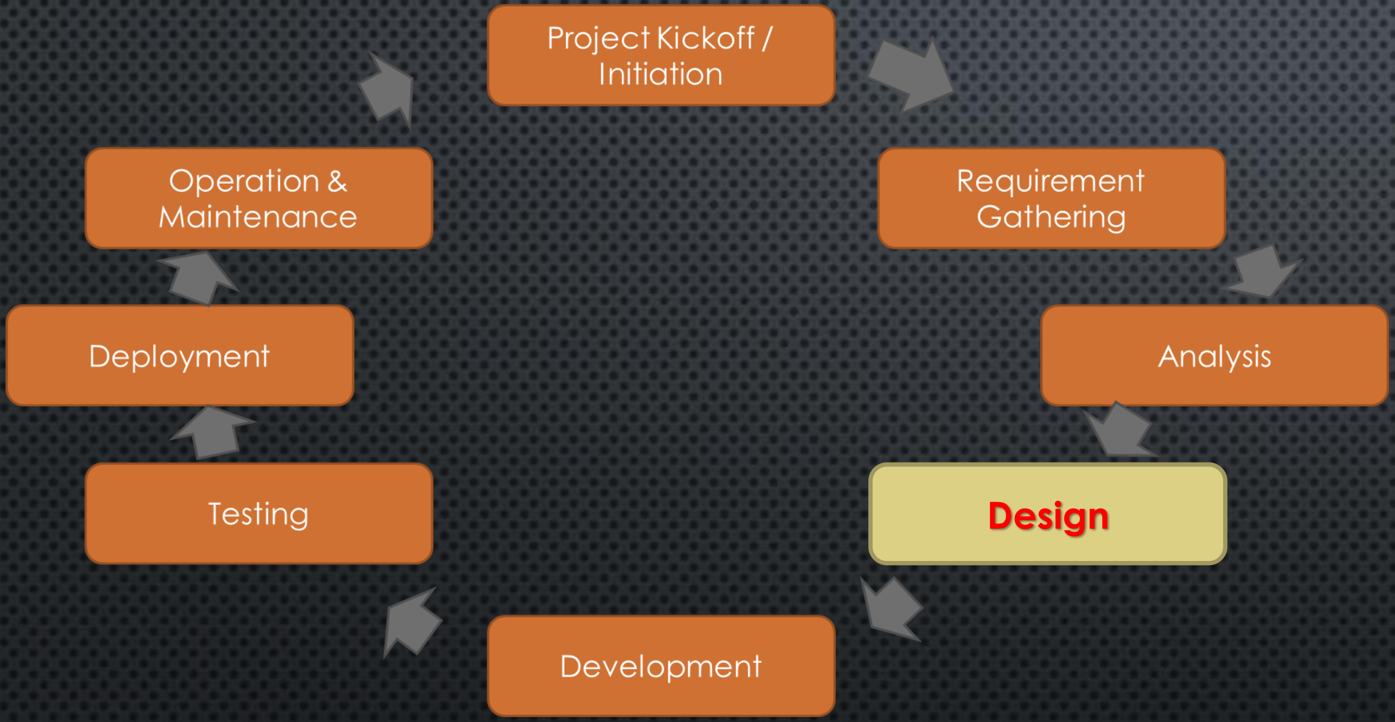


Entry Criteria

- Requirements
- Change requests

Exit Criteria

- Software Requirements Specifications Document

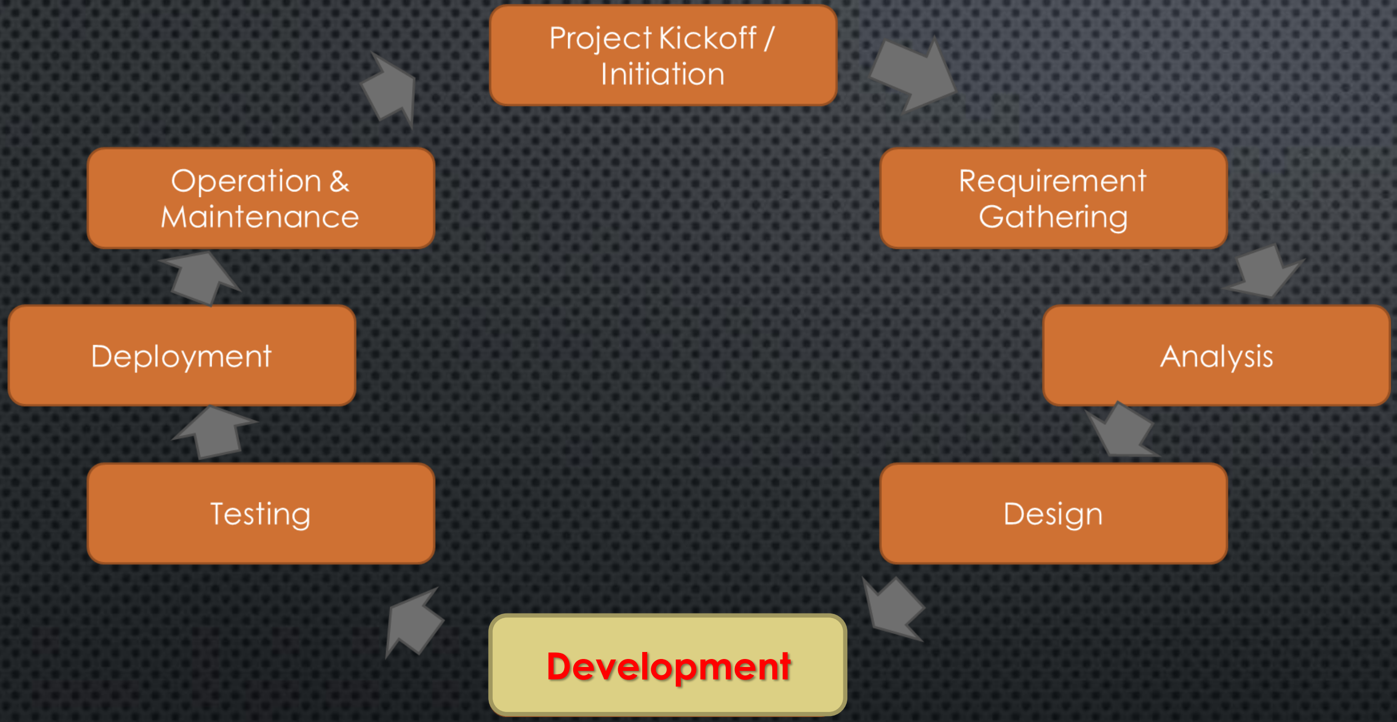


Entry Criteria

- Software Requirements Specifications Document

Exit Criteria

- Design Specifications Document

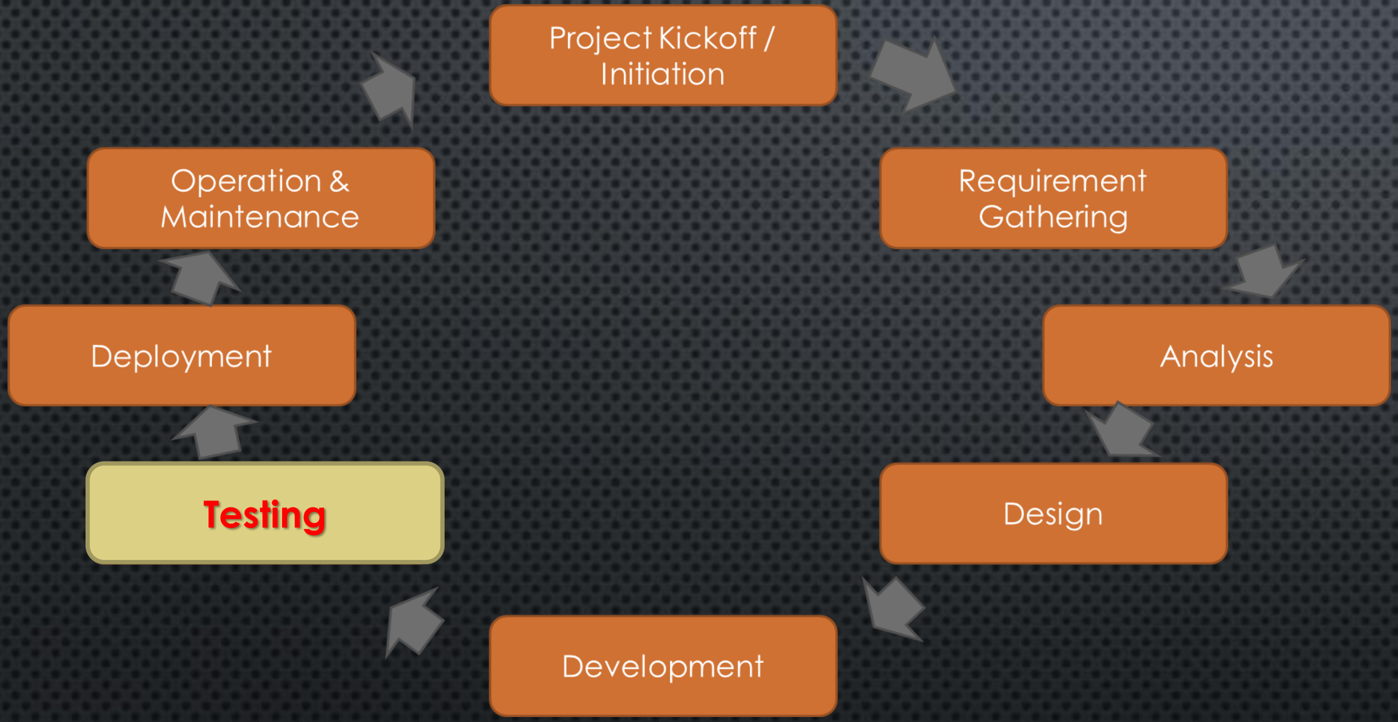


Entry Criteria

- Software Requirements Specifications Document
- Design Specifications Document

Exit Criteria

- Build Unit tests Set-up

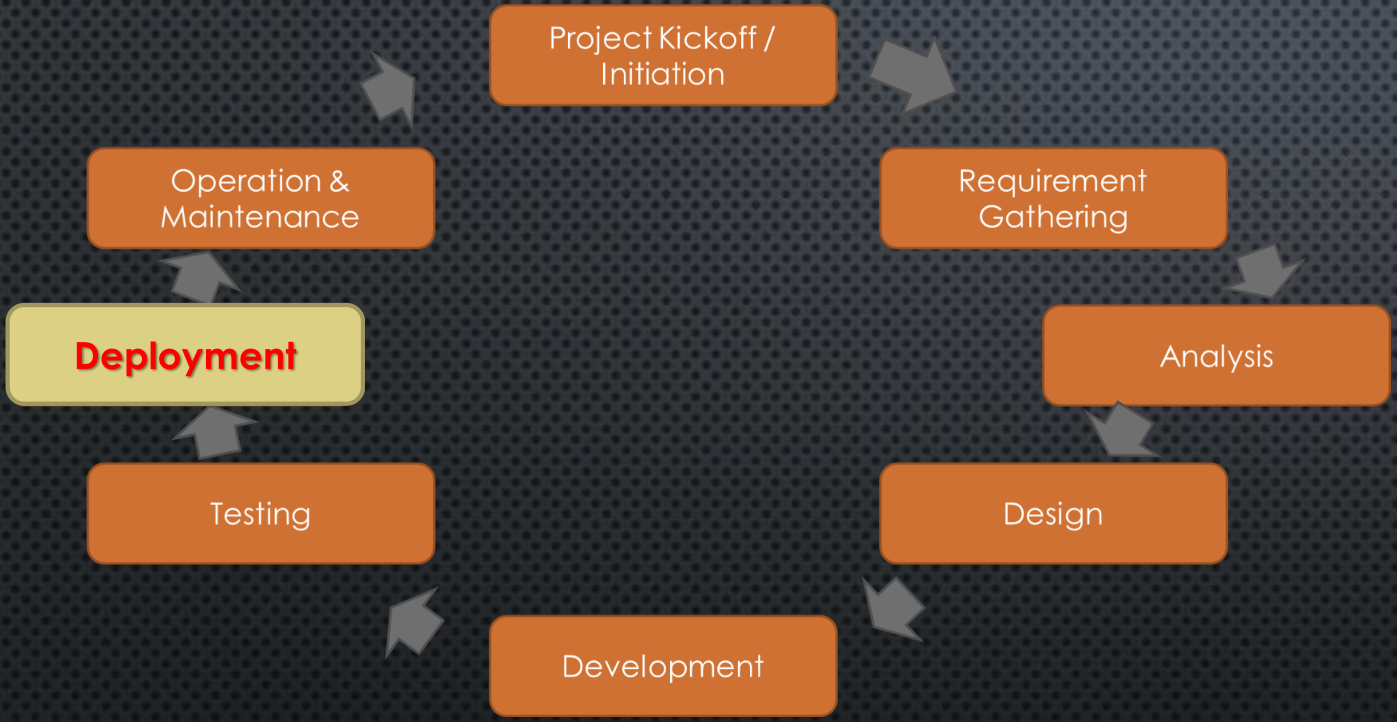


Entry Criteria

- Software Requirements Specifications Document
- Deployed build

Exit Criteria

- Test Cases
- System Testing Defects and their closure

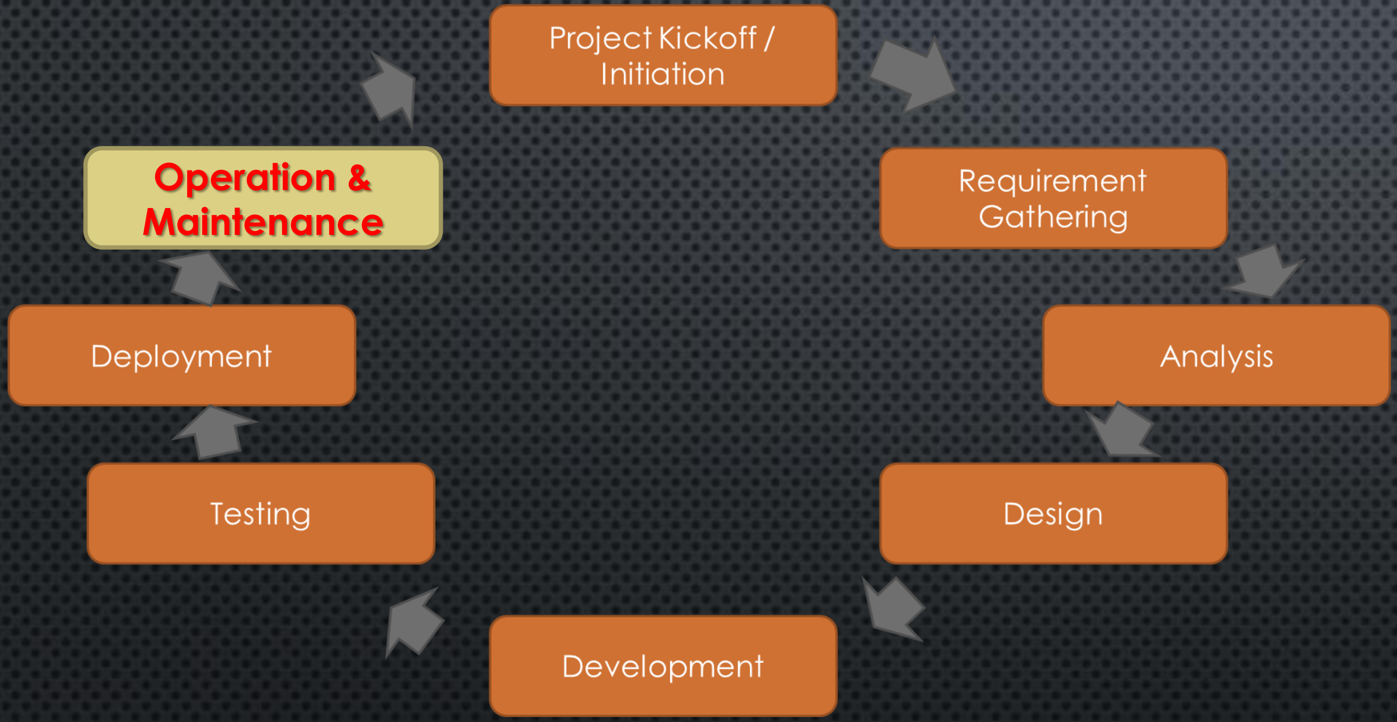


Entry Criteria

- No high priority defects

Exit Criteria

- Build deployment to Production



Entry Criteria

- Real users to use the software and report any issues they find
- Change requests

Exit Criteria

- Rectify the issues and deploy code fix to Production