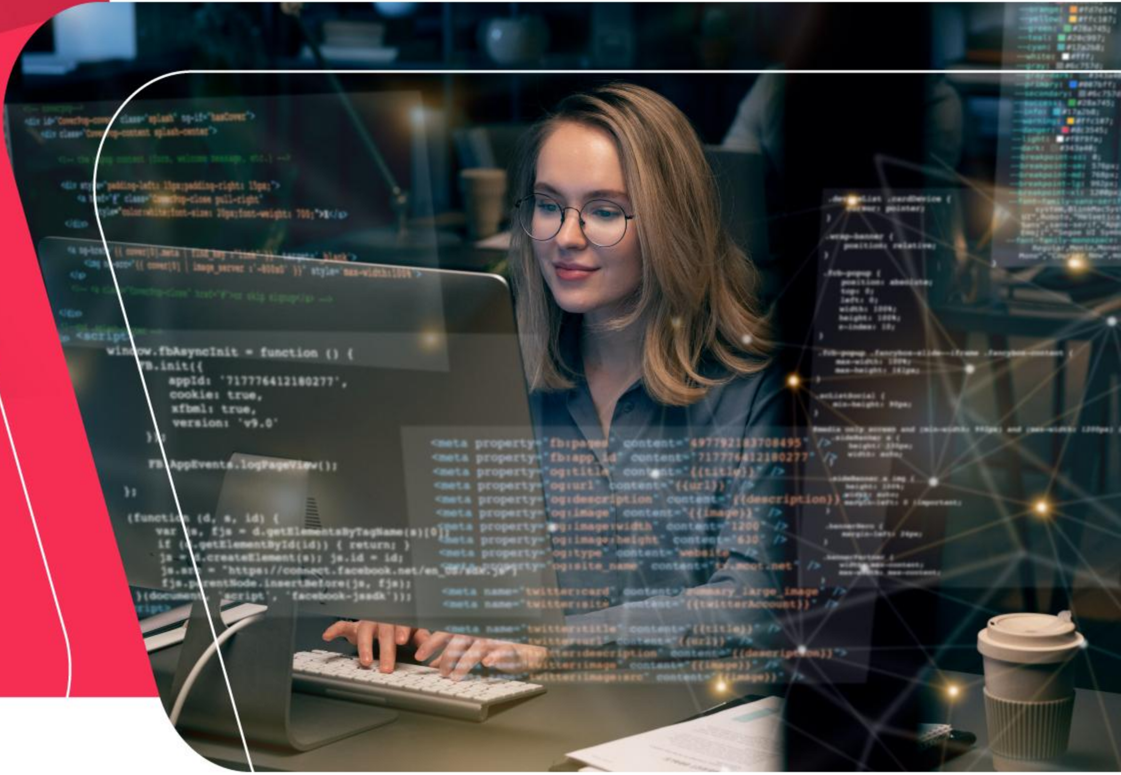
TIA-Academy

Java Bootcamp

Day 01



SDLC

TIA-Academy

What is SDLC?

- **SOFTWARE DEVELOPMENT LIFECYCLE (SDLC)** is a systematic process for building software that **ensures the quality and correctness of the software built.**
- SDLC process aims to **produce high-quality software** that meets customer expectations.
- The system development **should be complete in the pre-defined time frame and cost.**
- SDLC consists of a detailed plan which explains **how to plan, build, and maintain specific software.**
- Every phase of the SDLC life cycle has its **own process and deliverables that feed into the next phase.**

Why SDLC?

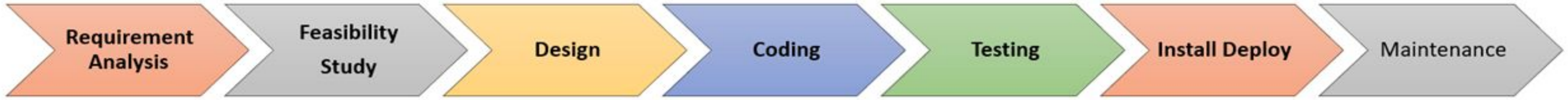
Here, are prime reasons why SDLC is important for developing a software system.

- It offers a **basis** for project planning, scheduling, and estimating
- **Provides a framework** for a standard set of activities and deliverables
- It is a **mechanism** for project tracking and control
- **Increases visibility** of project planning to all involved stakeholders of the development process
- **Increased and enhance** development speed
- **Improved** client relations
- Helps you to **decrease** project risk and project management plan overhead

TIA-Academy

SDLC Phases

The entire SDLC process divided into the following stages:

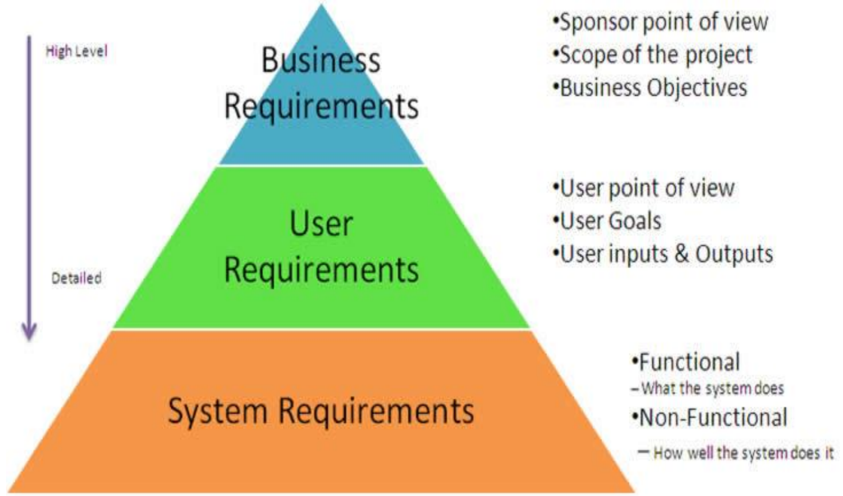


- Phase 1: Requirement collection and analysis
- Phase 2: Feasibility study
- Phase 3: Design
- Phase 4: Coding
- Phase 5: Testing
- Phase 6: Installation/Deployment
- Phase 7: Maintenance

TIA-Academy

Phase 1: Requirement collection and analysis:

- The requirement is the first stage in the SDLC process. It is **conducted by the senior team members with inputs from all the stakeholders** and domain experts in the industry.



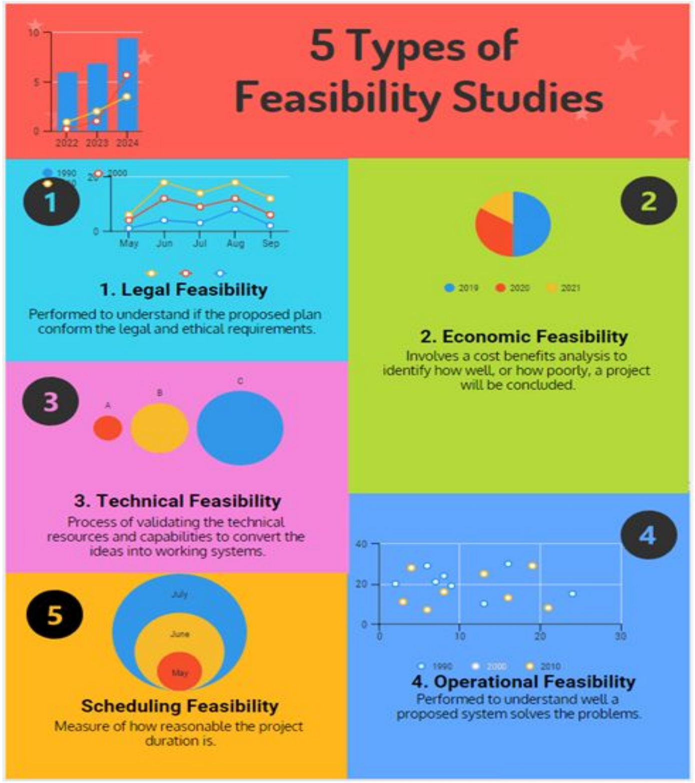
TIA-Academy

Phase 1: Requirement collection and analysis:

- Planning for the quality assurance requirements and re-cognization of the risks involved is also done at this stage.
- This stage gives a **clearer picture of the scope of the entire project and the anticipated issues, opportunities, and directives which triggered the project.**
- Requirements Gathering stage need teams to get detailed and precise requirements. This helps companies to **finalize the necessary timeline to finish the work of that system.**

TIA-Academy

Phase 2: Feasibility study:


5 Types of Feasibility Studies

1. Legal Feasibility
Performed to understand if the proposed plan conform the legal and ethical requirements.

2. Economic Feasibility
Involves a cost benefits analysis to identify how well, or how poorly, a project will be concluded.

3. Technical Feasibility
Process of validating the technical resources and capabilities to convert the ideas into working systems.

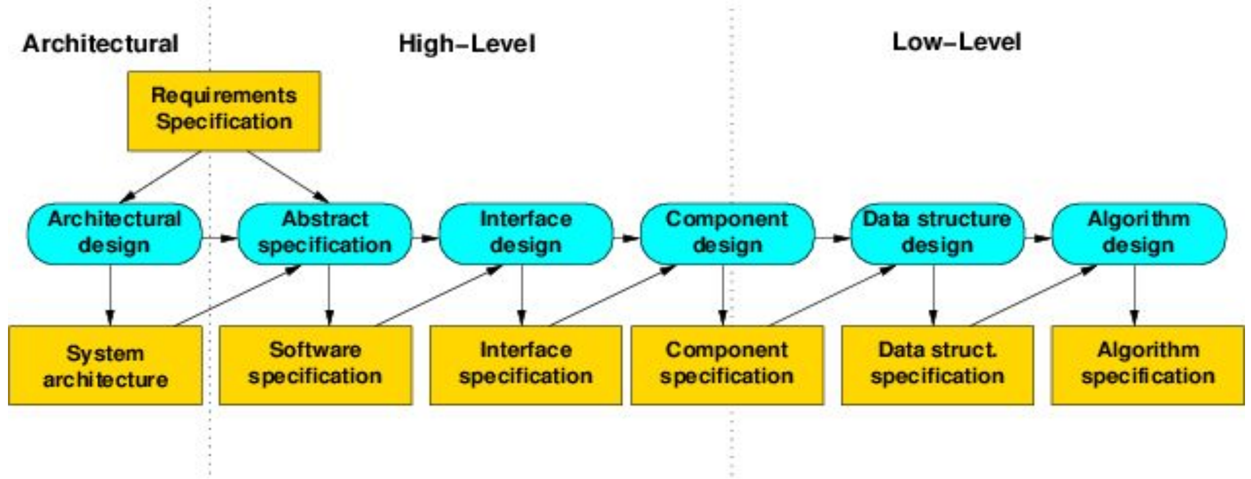
4. Operational Feasibility
Performed to understand well a proposed system solves the problems.

5 Scheduling Feasibility
Measure of how reasonable the project duration is.

- Once the requirement analysis phase is completed the next step is to define and document software needs.
- This process conducted with the help of 'Software Requirement Specification' document also known as 'SRS' document.
- It includes everything which should be designed and developed during the project life cycle.

Phase 3: Design

- In this third phase, the system and software design documents are prepared as per the requirement specification document. This helps define overall system architecture.
- This design phase serves as input for the next phase of the model.



TIA-Academy

Phase 3: Design: High-Level Design (HLD)

- Brief description and name of each module
- An outline about the functionality of every module
- Interface relationship and dependencies between modules
- Database tables identified along with their key elements
- Complete architecture diagrams along with technology details

TIA-Academy

Phase 3: Design: Low-Level Design(LLD)

- Functional logic of the modules
- Database tables, which include type and size
- Complete detail of the interface
- Addresses all types of dependency issues
- Listing of error messages
- Complete input and outputs for every module

TIA-Academy

Phase 4: Coding

- Once the system design phase is over, the next phase is **coding**. In this phase, **developers start build the entire system by writing code** using the chosen programming language.
- In the coding phase, **tasks are divided into units or modules and assigned to the various developers**. It is the longest phase of the Software Development Life Cycle process.
- In this phase, Developer needs to **follow certain predefined coding guidelines**. They also need to use programming tools like compiler, interpreters, debugger to generate and implement the code.

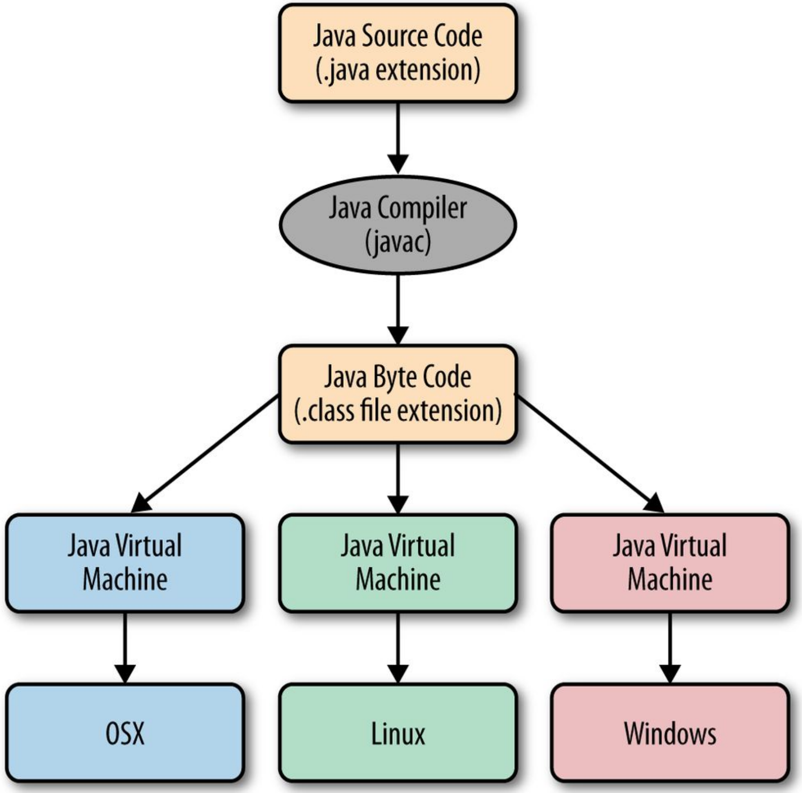
TIA-Academy

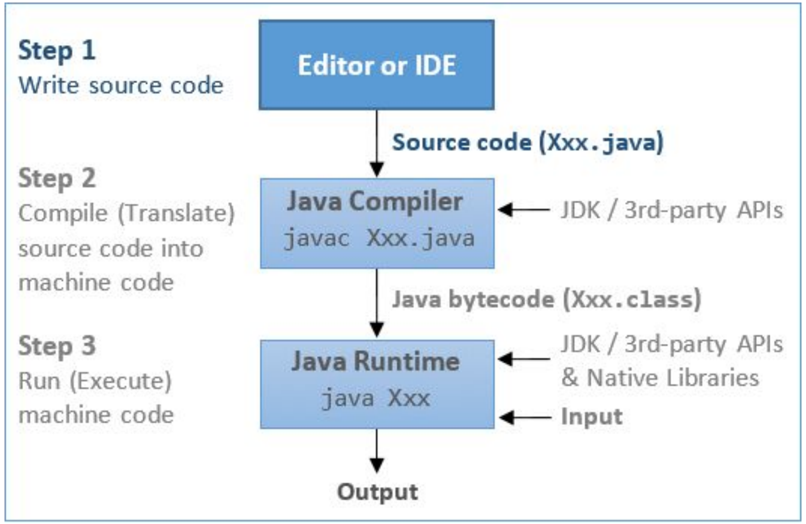
Phase 4: Coding

Rich API'S
API
Cloud Computing
Cloud Computing
Security Blocks
Java
Open Source
Robust
Memory Management
Cross Platform
Database Driven

TIA-Academy

Phase 4: Coding





TIA-Academy

Phase 5: Testing

- Once the software is complete, and it is deployed in the testing environment. **The testing team starts testing the functionality of the entire system**. This is done to verify that the entire application works according to the customer requirement.

Phase 5: Testing

- During this phase, QA and testing team **may find some bugs/defects which they communicate to developers**. The development team fixes the bug and send back to QA for a re-test. **This process continues until the software is bug-free, stable, and working according to the business needs of that system.**

Phase 6: Installation/Deployment

- Once the software testing phase is over and no bugs or errors left in the system then the final deployment process starts.

- Based on the feedback given by the project manager, **the final software is released and checked for deployment issues if any.**

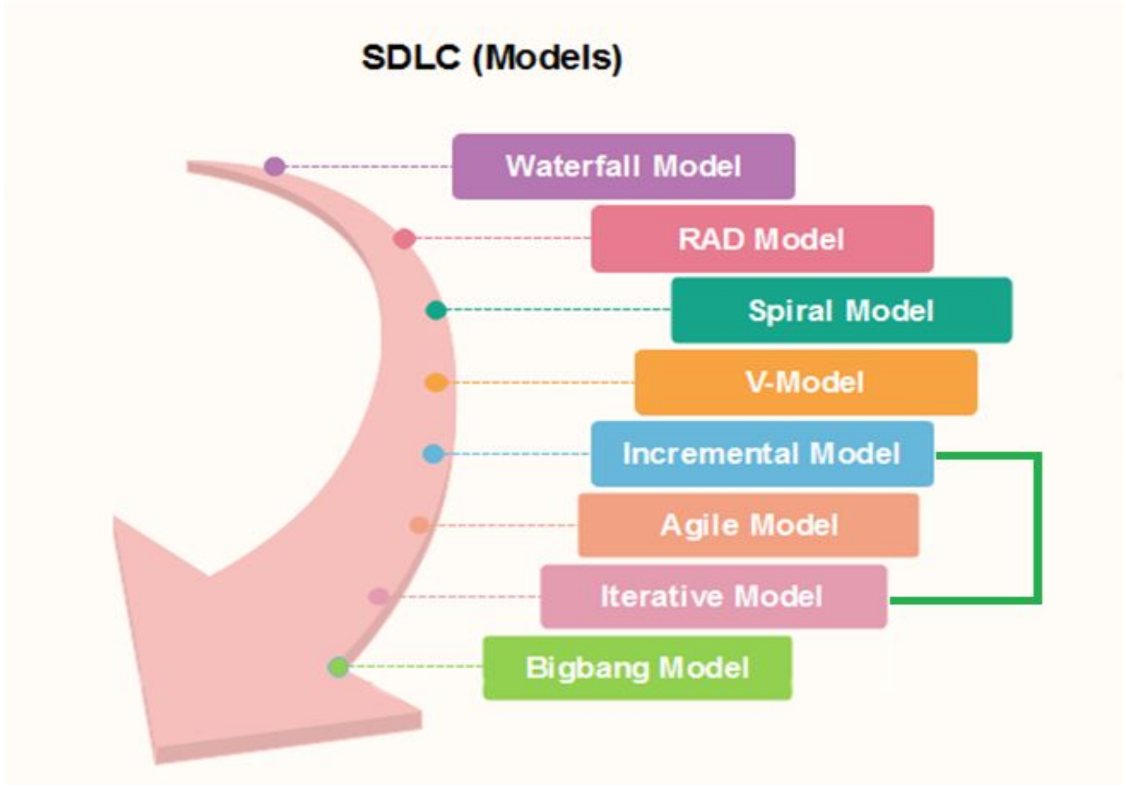
TIA-Academy

Phase 7: Maintenance

- Once the system is deployed, and customers start using the developed system, following 3 activities occur
  - Bug fixing - bugs are reported because of some scenarios which are not tested at all
  - Upgrade - Upgrading the application to the newer versions of the Software
  - Enhancement - Adding some new features into the existing software
- The main focus of this SDLC phase is to ensure that needs continue to be met and that the system continues to perform as per the specification mentioned in the first phase.

TIA-Academy

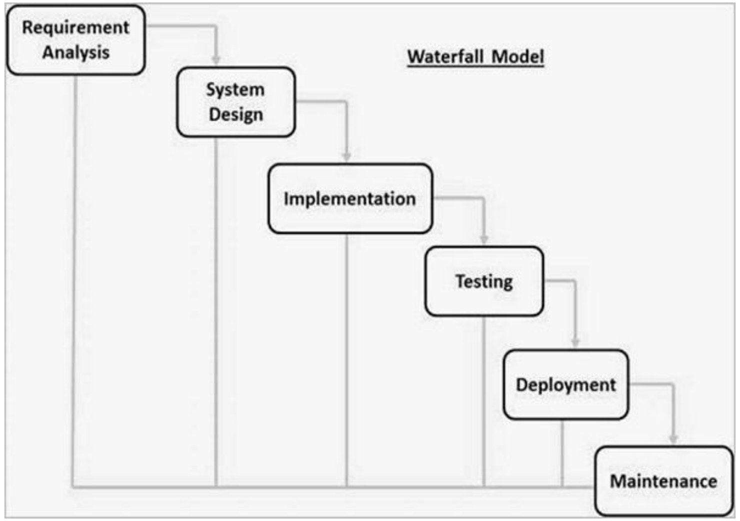
Popular SDLC models



TIA-Academy

Waterfall Model

- Waterfall approach was first SDLC Model to be used widely in Software Engineering to ensure success of the project. **In "The Waterfall" approach, the whole process of software development is divided into separate phases.**
- In this Waterfall model, typically, **the outcome of one phase acts as the input for the next phase sequentially.**
- The following illustration is a representation of the different phases of the Waterfall Model.



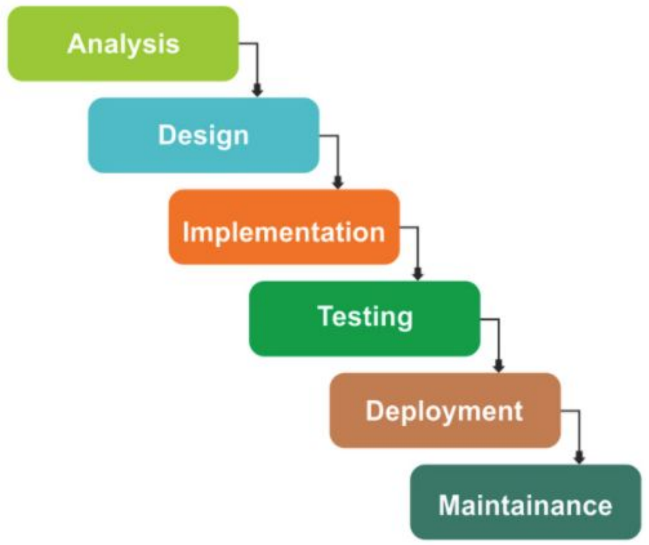
Waterfall Model

The sequential phases in Waterfall model are –

1. **Requirement Gathering and analysis** – All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.
2. **System Design** – The requirement specifications from first phase are studied in this phase and the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.
3. **Implementation** – With inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.
4. **Integration and Testing** – All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.
5. **Deployment of system** – Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.
6. **Maintenance** – There are some issues which come up in the client environment. To fix those issues, patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer

Waterfall Model

- All these phases are cascaded to each other in which progress is seen as flowing steadily **downwards (like a waterfall) through the phases.**
- The next phase is started only after the defined set of goals are achieved for previous phase and it is signed off, so the name "Waterfall Model". In this model, **phases do not overlap.**



Waterfall Model

Every software developed is different and requires a suitable SDLC approach to be followed based on the internal and external factors. Some situations where the use of **Waterfall model is most appropriate are** –

- Requirements are very well documented, clear and fixed.
- Product definition is stable.
- Technology is understood and is not dynamic.
- There are no ambiguous requirements.
- Ample resources with required expertise are available to support the product.
- The project is short.

Waterfall Model

Advantages

- **The advantages of waterfall development are that it allows for departmentalization and control**. A schedule can be set with deadlines for each stage of development and a product can proceed through the development process model phases one by one.
- Development moves from concept, through design, implementation, testing, installation, troubleshooting, and ends up at operation and maintenance. **Each phase of development proceeds in strict order.**

Waterfall Model

Some of the major advantages of the Waterfall Model are as follows –

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

Waterfall Model

Disadvantages

- The disadvantage of waterfall development is that it **does not allow much reflection or revision.**
- **Once an application is in the testing stage, it is very difficult to go back** and change something that was not well-documented or thought upon in the concept stage.

TIA-Academy

Waterfall Model

The major disadvantages of the Waterfall Model are as follows −

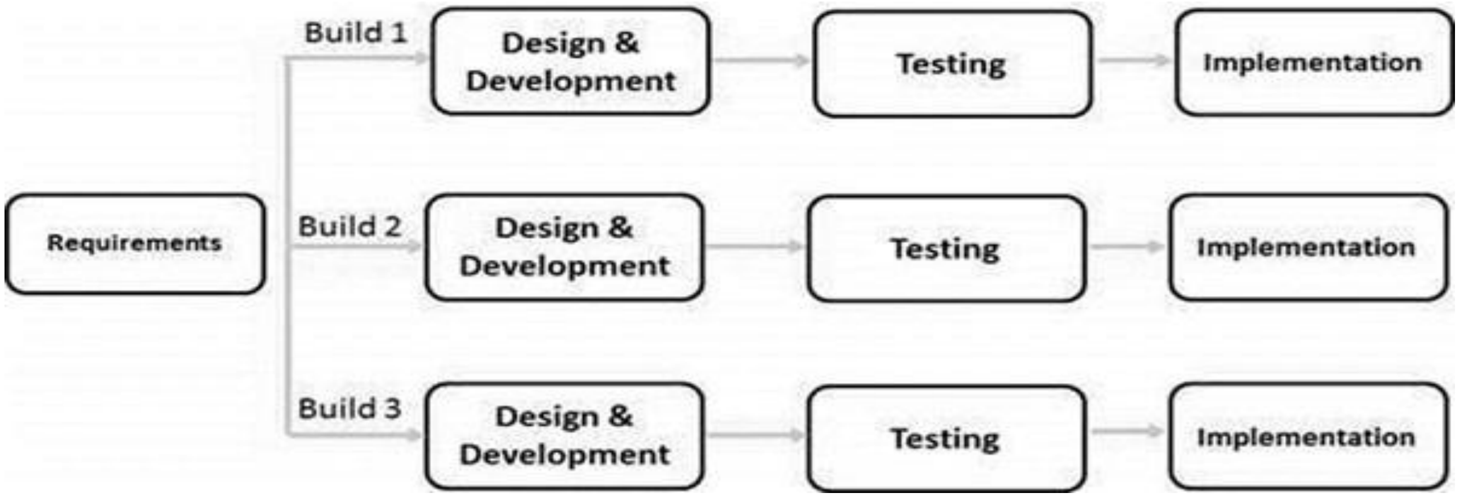
- No working software is produced until late during the life cycle.
- High amounts of risk and uncertainty.
- Not a good model for complex and object-oriented projects.
- Poor model for long and ongoing projects.
- Not suitable for the projects where requirements are at a moderate to high risk of changing. So, risk and uncertainty is high with this process model.
- It is difficult to measure progress within stages.
- Cannot accommodate changing requirements.
- Adjusting scope during the life cycle can end a project.
- Integration is done as a "big-bang. at the very end, which doesn't allow identifying any technological or business

Iterative & Incremental Model

- Iterative process **starts with a simple implementation of a subset of the software requirements and iteratively enhances the evolving versions** until the full system is implemented.
- At each iteration, design modifications are made and new functional capabilities are added. The basic idea behind this method is to **develop a system through repeated cycles (iterative) and in smaller portions at a time (incremental).**

Iterative & Incremental Model

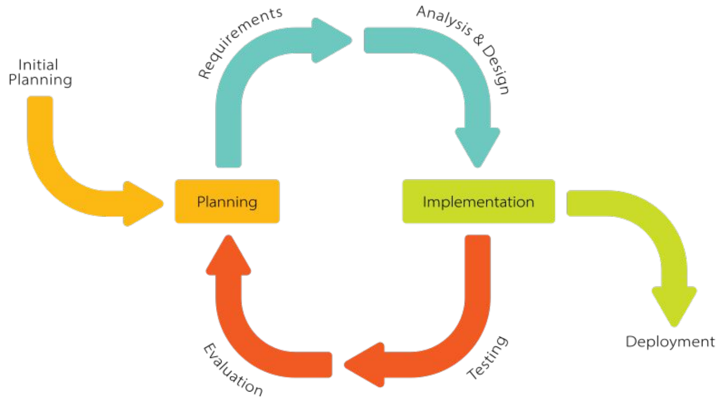
- The following illustration is a representation of the Iterative and Incremental model –



TIA-Academy

Iterative & Incremental Model

- Iterative and Incremental development is a **combination of both iterative design or iterative method and incremental build model for development.**
- "During software development, more than one iteration of the software development cycle may be **in progress at the same time**." This process may be described as an **"evolutionary acquisition" or "incremental build" approach."**



TIA-Academy

Iterative & Incremental Model

- In this incremental model, the whole requirement is divided into various builds. **During each iteration, the development module goes through the requirements, design, implementation and testing phases.** Each subsequent release of the module adds function to the previous release. **The process continues till the complete system is ready as per the requirement.**
- The key to a successful use of an iterative software development lifecycle **is rigorous validation of requirements, and verification & testing of each version of the software against those requirements within each cycle of the model.** As the software evolves through successive cycles, tests must be repeated and

TIA-Academy

Iterative & Incremental Model

Like other SDLC models, Iterative and incremental development has some specific applications in the software industry. This model is most often used in the following scenarios −

- Requirements of the complete system are clearly defined and understood.
- Major requirements must be defined; however, some functionalities or requested enhancements may evolve with time.
- There is a time to the market constraint.
- A new technology is being used and is being learnt by the development team while working on the project.
- Resources with needed skill sets are not available and are planned to be used on contract basis for specific iterations.
- There are some high-risk features and goals which may change in the future.

TIA-Academy

Iterative & Incremental Model

- The **advantage of this model is that there is a working model of the system at a very early stage of development, which makes it easier to find functional or design flaws.** Finding issues at an early stage of development enables to take corrective measures in a limited budget.
- **The disadvantage with this SDLC model is that it is applicable only to large and bulky software development projects.** This is because it is hard to break a small software system into further small serviceable increments/modules.

TIA-Academy

Iterative & Incremental Model

Advantages

- Some working functionality can be developed quickly and early in the life cycle.
- Results are obtained early and periodically.
- Parallel development can be planned.
- Progress can be measured.
- Less costly to change the scope/requirements.
- Testing and debugging during smaller iteration is easy.
- Risks are identified and resolved during iteration; and each iteration is an easily managed milestone.
- Easier to manage risk - High risk part is done first.
- With every increment, operational product is delivered.
- Issues, challenges and risks identified from each increment can be utilized/applied to the next increment.
- Risk analysis is better.
- It supports changing requirements.
- Initial Operating time is less.

TIA-Academy

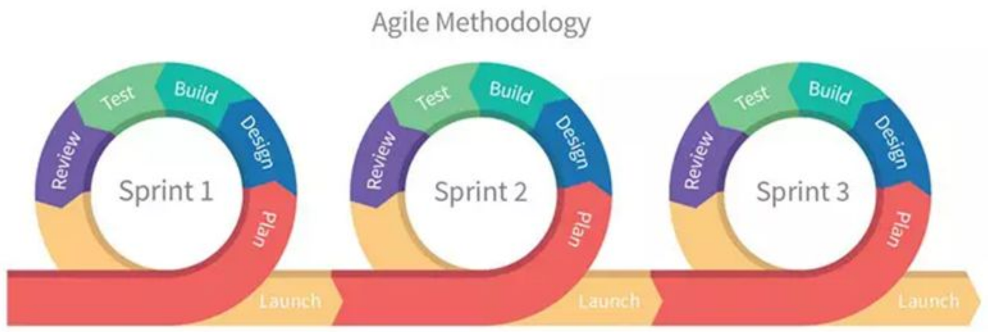
Iterative & Incremental Model

Disadvantages

- More resources may be required.
- Although cost of change is lesser, but it is not very suitable for changing requirements.
- More management attention is required.
- System architecture or design issues may arise because not all requirements are gathered in the beginning of the entire life cycle.
- Defining increments may require definition of the complete system.
- Not suitable for smaller projects.
- Management complexity is more.
- End of project may not be known which is a risk.
- Highly skilled resources are required for risk analysis.
- Projects progress is highly dependent upon the risk analysis phase.

Agile Model

- Agile SDLC model is a **evolution of iterative and incremental process models with focus on process adaptability and customer satisfaction by rapid delivery of working software product.**
- Agile Methods **break the product into small incremental builds**. These builds are provided in iterations. Each iteration typically lasts from about one to three weeks.



Agile Model

Every iteration involves cross functional teams working simultaneously on various areas like –

- Planning
- Requirements Analysis
- Design
- Coding
- Unit Testing and
- Acceptance Testing.

At the end of the iteration, a working product is displayed to the customer and important stakeholders.

Agile Model

- Agile model believes **that every project needs to be handled differently** and the existing methods need to be tailored to best suit the project requirements. In Agile, the tasks are divided to time boxes (small time frames) to deliver specific features for a release.
- Iterative approach is taken and working software build is delivered after each iteration. Each build is incremental in terms of features; **the final build holds all the features required by the customer.**

TIA-Academy

Agile Model

- The Agile thought process had started early in the software development and started becoming popular with time due to its flexibility and adaptability.
- Following are the Agile Manifesto principles –
  - **Individuals and interactions** – In Agile development, self-organization and motivation are important, as are interactions like co-location and pair programming.
  - **Working software** – Demo working software is considered the best means of communication with the customers to understand their requirements, instead of just depending on documentation.
  - **Customer collaboration** – As the requirements cannot be gathered completely in the beginning of the project due to various factors, continuous customer interaction is very important to get proper product requirements.
  - **Responding to change** – Agile Development is focused on quick responses to change and continuous development.

TIA-Academy

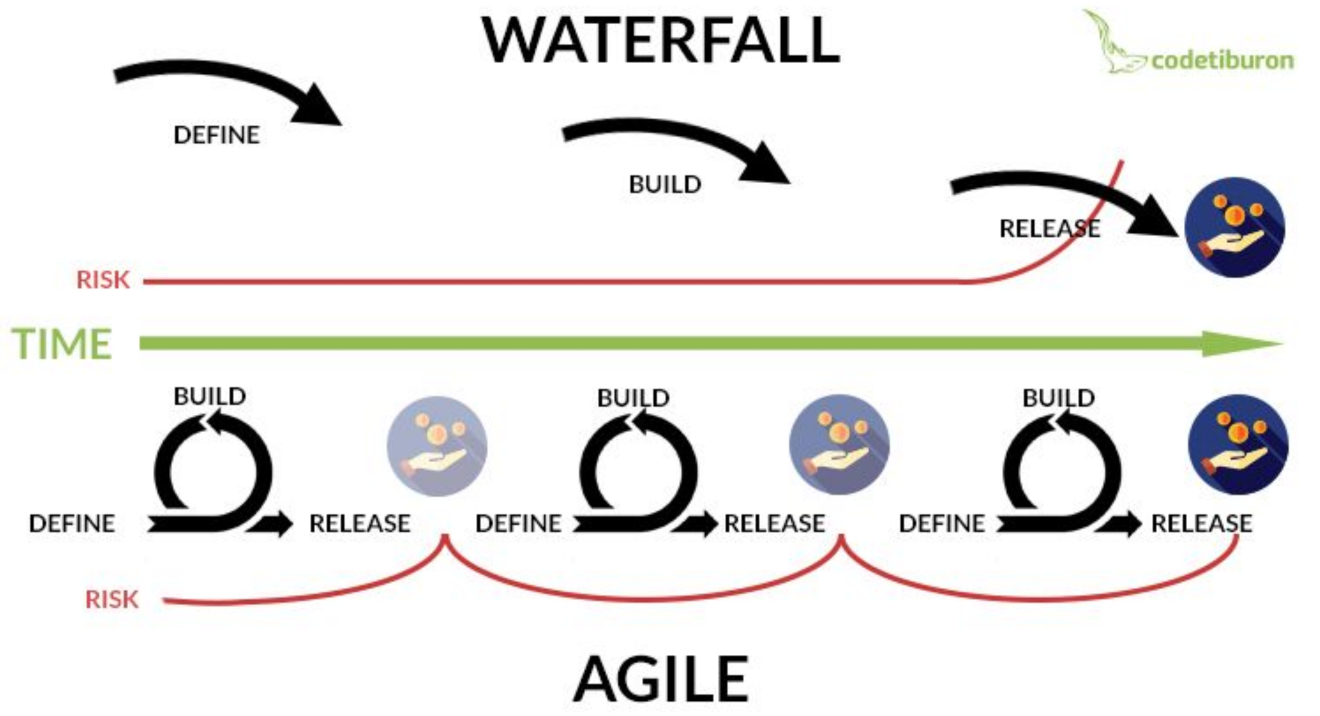
Agile Model

Agile VS Traditional SDLC Models

- **Agile** is based on the **adaptive software development methods**, whereas the **traditional SDLC models** like the waterfall model is based on a **predictive approach.**
- Predictive teams in the traditional SDLC models usually work with detailed planning and have a complete forecast of the exact tasks and features to be delivered in the next few months or during the product life cycle.
- Predictive methods entirely depend on the **requirement analysis and planning** done in the beginning of cycle. Any changes to be incorporated go through a strict change control management and prioritization.

TIA-Academy

Waterfall Vs Agile



Agile Model

Agile VS Traditional SDLC Models

- Agile uses an **adaptive approach** where there is no detailed planning and there is clarity on future tasks only in respect of what features need to be developed. There is feature driven development and the team adapts to the changing product requirements dynamically. The product is tested very frequently, through the release iterations, minimizing the risk of any major failures in future.
- **Customer Interaction** is the backbone of this Agile methodology, and **open communication with minimum documentation are the typical features of Agile development environment.** The agile teams work in close collaboration with each other and are most often **located in the same geographical location.**

Agile Model

- Is a very realistic approach to software development.
- Promotes teamwork and cross training.
- Functionality can be developed rapidly and demonstrated.
- Resource requirements are minimum.
- Suitable for fixed or changing requirements
- Delivers early partial working solutions.

Advantages

- Good model for environments that change steadily.
- Minimal rules, documentation easily employed.
- Enables concurrent development and delivery within an overall planned context.
- Little or no planning required.
- Easy to manage.
- Gives flexibility to developers.

TIA-Academy

Agile Model

Disadvantages

- Not suitable for handling complex dependencies.
- More risk of sustainability, maintainability and extensibility.
- An overall plan, an agile leader and agile PM practice is a must without which it will not work.
- Strict delivery management dictates the scope, functionality to be delivered, and adjustments to meet the deadlines.
- Depends heavily on customer interaction, so if customer is not clear, team can be driven in the wrong direction.
- There is a very high individual dependency, since there is minimum documentation generated.
- Transfer of technology to new team members may be quite challenging due to lack of documentation.