SDM (Software development Model)

The **software development models** are the various processes or methodologies that are being selected for the development of the project depending on the project’s aims and goals. There are many development life cycle models that have been developed in order to achieve different required objectives. The models specify the various stages of the process and the order in which they are carried out.

The selection of model has very high impact on the testing that is carried out. It will define the what, where and when of our planned testing, influence regression testing and largely determines which test techniques to use.

1. [**Waterfall model**](http://tryqa.com/what-is-waterfall-model-advantages-disadvantages-and-when-to-use-it/)
2. [**V model**](http://tryqa.com/what-is-v-model-advantages-disadvantages-and-when-to-use-it/)
3. [**Incremental model**](http://tryqa.com/what-is-incremental-model-advantages-disadvantages-and-when-to-use-it/)
4. [**RAD model**](http://tryqa.com/what-is-rad-model-advantages-disadvantages-and-when-to-use-it/)
5. [**Agile model**](http://tryqa.com/what-is-agile-model-advantages-disadvantages-and-when-to-use-it/)
6. [**Iterative model**](http://tryqa.com/what-is-iterative-model-advantages-disadvantages-and-when-to-use-it/)

Different companies based on the software application or product, they select the type of development model whichever suits to their application. But these days in market the ‘**Agile Methodology**‘ is the most used model.

What is Software Development Life Cycle?

Software Development Life Cycle (SDLC) is the set of processes used to develop and deliver high quality software.

SDLC 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.

Its beneficial to follow SDLC when you need to combine technical and non-technical activities to deliver high [**quality software**](http://tryqa.com/what-is-software-quality/).

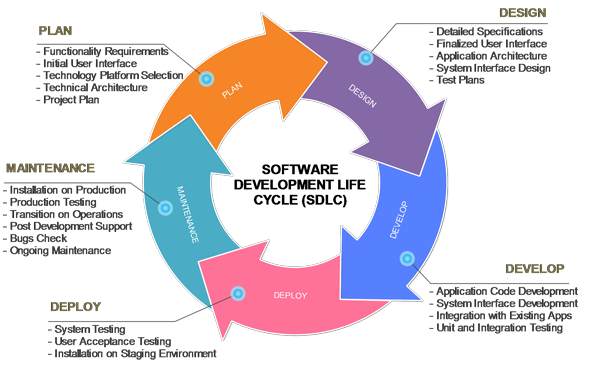
## Why is Software Development Life Cycle necessary?

Software Development Life Cycle is needed in any of the project for the below 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
* 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

These steps are (very) roughly the same from one methodology to another. They tend to occur in this order, though they can also be mixed together, such that several steps occur in parallel.



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**](http://tryqa.com/what-are-the-software-development-models/)  and the project can choose any among them based on the strategies, schedules, complexity and many more business factors.

How to select the right SDLC

Selecting the right SDLC is a process in itself that the organization can implement internally or consult for. There are some steps to get the right selection.

STEP 1: Learn the about SDLC Models

SDLCs are the same in their usage. In order to select the [right SDLC](https://melsatar.blog/2017/05/05/the-best-sdlc-model/), you should have enough experience and be familiar with the SDLCs that will be chosen and understand them correctly.

STEP 2: Assess the needs of Stakeholders

We must study the business domain, stakeholders concerns and requirements, business priorities, our technical capability and ability, and technology constraints to be able to choose the right SDLC against their selection criteria.

STEP 3: Define the criteria

Some of the selection criteria or arguments that you may use to select an SDLC are:

* Is the SDLC suitable for the size of our team and their skills?
* Is the SDLC suitable for the selected technology we use for implementing the solution?
* Is the SDLC suitable for client and stakeholders concerns and priorities?
* Is the SDLC suitable for the geographical situation (distributed team)?
* Is the SDLC suitable for the size and complexity of our software?
* Is the SDLC suitable for the type of projects we do?
* Is the SDLC suitable for our software engineering capability?
* Is the SDLC suitable for the project risk and quality insurance?

What are the criteria?

Here are my recommended criteria, It will be good to share any new criteria you see that it will be valid



STEP 4: Decide

When you define the criteria and the arguments you need to discuss with the team, you will need to have [a decision matrix](https://melsatar.blog/2017/09/23/trade-off-analysis-technique-make-the-decision-easier/) and give each criterion a defined weight and score for each option. After analyzing the results, you should document this decision in the project artifacts and share it with the related stakeholders.

STEP 5: Optimize

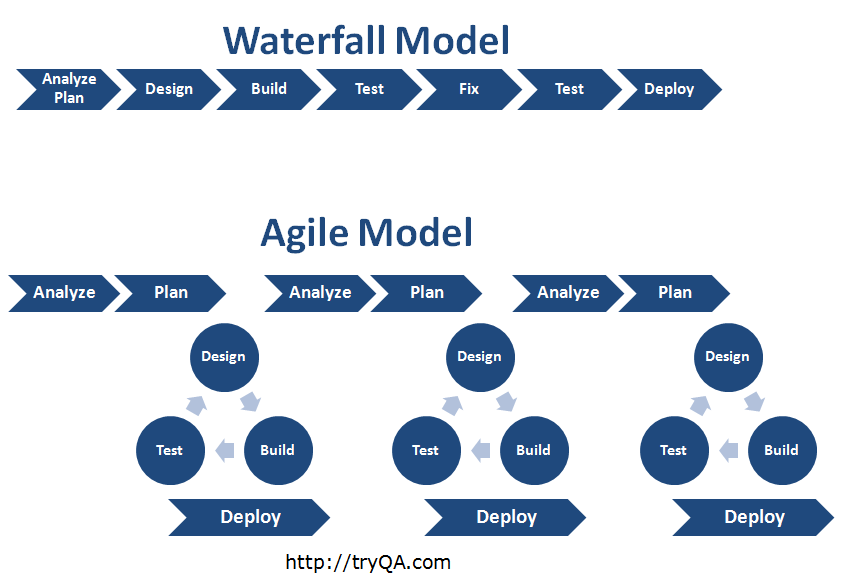
You can always optimize the SDLC during the project execution, you may notice upcoming changes do not fit with the selected SDLC, it is okay to align and cope with the changes. You can even make your own SDLC model which optimum for your organization or the type of projects you are involved in.

References

Software Development Life Cycle Models and Methologies. (2012, 3). Retrieved from melsatar.blog:  [Software Development Life Cycle Models and Methodologies](http://melsatar.blog/2012/03/15/software-development-life-cycle-models-and-methodologies/)

**Agile Methodology**

Agile is the methodology where the requirements, development and testing stages are all ongoing processes and the system analysts, developers, testers and customers all work together as a single team.



Here the releases are shorter, usually 6 – 8 weeks and each release has the timeframe called “Sprints” which is of 2 weeks (typically). Each sprint produces a work product.

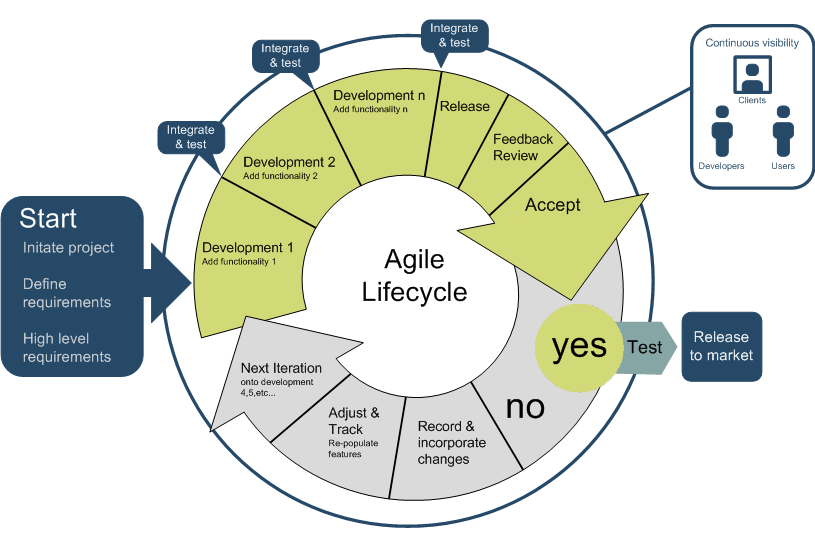
The collected requirements (user stories) are prioritized for the release and they are again prioritized for sprints. All the resources in the team work towards the identified user stories for the sprint.

When the developers are coding the user stories, testers identify acceptance test cases, system test cases, regression test cases for the user stories.

When the build is deployed, testers execute the identified test cases and reports defects and developers focus on fixing the defects and makes sure to close all the defects within the sprint itself. Once one sprint ends the next spring begins.

This goes on in all the other sprints which keeps including new features and enhancements to the existing features.

Customer interaction is very high in this methodology and the feedback and suggestions are collected at regular intervals to enhance the quality of the software.



**Principles of Agile:**

* Individuals and interactions are given more priority than processes.
* Working software is considered than heavy documentation of the software
* Customer collaboration is very extensive and the continuous feedback, suggestions and discussions helps in improving the software to meet user needs
* Respond to change in requirements dynamically. High priority requirements if coming in later stages, then the medium or low-level priority items are moved right to accommodate the high priority ones.

**Agile Methodology is best suitable when:**

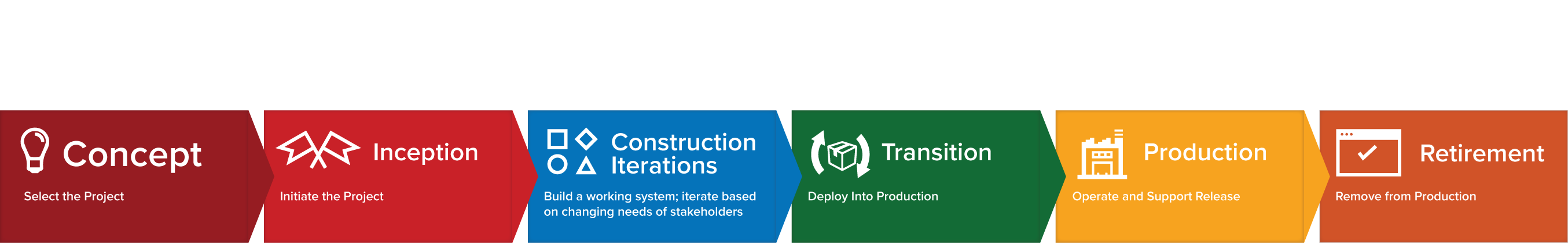
* Dynamic change in user requirements
* Early returns of Investment
* Low budget for changing requirements

### **THE AGILE PROCESS FLOW**

1. **Concept** - Projects are envisioned and prioritized
2. **Inception** - Team members are identified, funding is put in place, and initial environments and requirements are discussed
3. **Iteration/Construction** - The development team works to deliver working software based on iteration requirements and feedback
4. **Release** - QA (Quality Assurance) testing, internal and external training, documentation development, and final release of the iteration into production
5. **Production** - Ongoing support of the software
6. **Retirement** - End-of-life activities, including customer notification and migration

This view presents the full Agile lifecycle model within the enterprise. In any enterprise there may be projects operating simultaneously, multiple sprints/iterations being logged on different product lines, and a variety of customers, both external and internal, with a range of business needs.

**Agile Software Development Lifecycle**



### THE AGILE ITERATION WORKFLOW

The Agile software development lifecycle is dominated by the iterative process. Each iteration results in the next piece of the software development puzzle - working software and supporting elements, such as documentation, available for use by customers - until the final product is complete. Each iteration is usually two to four weeks in length and has a fixed completion time. Due to its time-bound nature, the iteration process is methodical and the scope of each iteration is only as broad as the allotted time allows.

Multiple iterations will take place during the Agile software development lifecycle and each follows its own workflow. During an iteration, it is important that the customers and business stakeholders provide feedback to ensure that the features meet their needs.

A typical iteration process flow can be visualized as follows:

* **Requirements** - Define the requirements for the iteration based on the product backlog, sprint backlog, customer and stakeholder feedback
* **Development** - Design and develop software based on defined requirements
* **Testing** - QA (Quality Assurance) testing, internal and external training, documentation development
* **Delivery** - Integrate and deliver the working iteration into production
* **Feedback** - Accept customer and stakeholder feedback and work it into the requirements of the next iteration



For the duration of the project, while additional features may be fed into the product backlog, the rest of the process is a matter of repeating the steps over and over until all of the items in the product backlog have been fulfilled. As a result, the process flow is more of a loop and not a linear process.

### **MAKING THE AGILE PROCESS WORK FOR us**

As with any methodology, there are advantages and disadvantages. The Agile method is more suitable in situations where customers and project stakeholders are available to provide input, functional portions of software are needed quickly, flexibility is desired to accommodate changing requirements, and the team is co-located and able to effectively collaborate. As with any change, integrating Agile processes into your business can be overwhelming. Below are four activities that will help support the adoption of Agile workflow:

* **Daily Meetings** - Host consistent or daily stand-up meetings to maintain open communication, hold workers accountable, and keep each iteration moving forward
* **Live Demonstrations** - Deliver live demonstrations of each iteration’s final product to show progress
* **Share Feedback** - Receive feedback from stakeholders and customers and share it with the entire team before the next iteration begins
* **Remain Agile** - Make changes to your process based on feedback to ensure each iteration improves the last

**Advantages -Pros- of Agile Model**

* In Agile methodology the delivery of software is unremitting.
* The customers are satisfied because after every Sprint working feature of the software is delivered to them.
* Customers can have a look of the working feature which fulfilled their expectations.
* If the customers has any feedback or any change in the feature then it can be accommodated in the current release of the product.
* In Agile methodology the daily interactions are required between the business people and the developers.
* In this methodology attention is paid to the good design of the product.
* Changes in the requirements are accepted even in the later stages of the development.
* People oriented, more focus is on enhancing skill sets
* Minimum documentation saves time
* Supports dynamically changing requirements
* Helps in focusing on common goal
* High customer interaction improvises the quality continuously
* Highly flexible and realistic approach
* Early partial working software
* Less resources and equally distributed tasks
* Minimized risks

## Disadvantages -Cons- of Agile Model

* In Agile methodology the documentation is less.
* Sometimes in Agile methodology the requirement is not very clear hence it’s difficult to predict the expected result.
* In few of the projects at the starting of the software development life cycle it’s difficult to estimate the actual effort required.
* The projects following the Agile methodology may have to face some unknown risks which can affect the development of the project.
* Cannot finalize on the cost
* Client-oriented team
* Does not address dependencies
* Lack of detailed documentation causes challenging situation for the new members in the team