# **Software Development Life Cycle Models and Methodologies**

## Introduction

Software development life cycle (**SDLC**) is a series of [phases](http://melsatar.blog/2017/06/13/what-do-you-need-to-know-about-the-eight-software-development-phases/) that provide a common understanding of the software building process. How the software will be realized and developed from the business understanding and requirements elicitation phase to convert these business ideas and requirements into functions and features until its usage and operation to achieve the business needs. The good software engineer should have enough knowledge on how to choose the SDLC model based on the project context and the business requirements.

Therefore, it may be required to choose the right SDLC model according to the specific concerns and requirements of the project to ensure its success.

You can think of SDLC models as tools that you can use to better deliver your software project. Therefore, knowing and understanding each model and when to use it, the advantages and disadvantages of each one are important to know which one is suitable for the project context.



## Types of Software developing life cycles (SDLC)

* [Waterfall Model](http://melsatar.blog/2018/02/16/the-waterfall-model-a-different-perspective/)
* [V-Shaped Model](https://melsatar.blog/2018/08/27/the-validation-and-verification-model-the-v-model/)
* [Evolutionary Prototyping](http://en.wikipedia.org/wiki/Software_prototyping) Model
* [Spiral](http://en.wikipedia.org/wiki/Spiral_model) Method ([SDM](http://en.wikipedia.org/wiki/Software_development_methodology))
* [Iterative and Incremental](http://en.wikipedia.org/wiki/Iterative_and_incremental_development) Method
* [Agile development](http://en.wikipedia.org/wiki/Agile_software_development)

### Waterfall Model

#### **Description**

The [Waterfall Model](http://melsatar.blog/2018/02/16/the-waterfall-model-a-different-perspective/) is a linear sequential flow. In which progress is seen as flowing steadily downwards (like a waterfall) through the phases of software implementation. This means that any phase in the development process begins only if the previous phase is complete. The waterfall approach does not define the process to go back to the previous phase to handle changes in requirement. The waterfall approach is the earliest approach and most widely known that was used for software development.



#### **The usage**

Projects which not focus on changing the requirements, for example, projects initiated from a request for proposals ([RFPs](http://en.wikipedia.org/wiki/Request_for_proposal)), the customer has a very clear documented requirements

|  |  |
| --- | --- |
| Advantages | Disadvantages |
| * Easy to explain to the users.
* Structures approach.
* Stages and activities are well defined.
* Helps to plan and schedule the project.
* Verification at each stage ensures early detection of errors/misunderstanding.
* Each phase has specific deliverables.
 | * Assumes that the requirements of a system can be frozen.
* Very difficult to go back to any stage after it finished.
* A little flexibility and adjusting scope is difficult and expensive.
* Costly and required more time, in addition to the detailed plan.
 |

#### **Advantages and Disadvantages**

### [V-Shaped Model](https://melsatar.blog/2018/08/27/the-validation-and-verification-model-the-v-model/)

#### **Description**

It is an extension of the waterfall model, Instead of moving down in a linear way, the process steps are bent upwards after the implementation and coding phase, to form the typical V shape. The major difference between the V-shaped model and waterfall model is the early test planning in the V-shaped model.



#### **The usage**

* Software requirements clearly defined and known
* Software development technologies and tools are well-known

|  |  |
| --- | --- |
| Advantages | Disadvantages |
| * Simple and easy to use
* Each phase has specific deliverables.
* Higher chance of success over the waterfall model due to the development of test plans early on during the life cycle.
* Works well for where requirements are easily understood.
* Verification and validation of the product in the early stages of product development.
 | * Very inflexible, like the waterfall model.
* Adjusting scope is difficult and expensive.
* The software is developed during the implementation phase, so no early prototypes of the software are produced.
* The model doesn’t provide a clear path for problems found during testing phases.
* Costly and required more time, in addition to a detailed plan
 |

#### **Advantages and Disadvantages**

### Prototyping Model

#### **Description**

It refers to the activity of creating prototypes of software applications, for example, incomplete versions of the software program being developed. It is an activity that can occur in software development and It used to visualize some component of the software to limit the gap of misunderstanding the customer requirements by the development team. This also will reduce the iterations may occur in the waterfall approach and hard to be implemented due to the inflexibility of the waterfall approach. So, when the final prototype is developed, the requirement is considered to be frozen.

It has some types, such as:

* Throwaway prototyping: Prototypes that are eventually discarded rather than becoming a part of the finally delivered software



* Evolutionary prototyping: prototypes that evolve into the final system through an iterative incorporation of user feedback.
* Incremental prototyping: The final product is built as separate prototypes. In the end, the separate prototypes are merged in an overall design.



* Extreme prototyping: used in web applications mainly. Basically, it breaks down web development into three phases, each one based on the preceding one. The first phase is a static prototype that consists mainly of HTML pages. In the second phase, the screens are programmed and fully functional using a simulated services layer. In the third phase, the services are implemented

#### **The usage**

* This process can be used with any software developing life cycle model. While this shall be chosen when you are developing a system has user interactions. So, if the system does not have user interactions, such as a system does some calculations shall not have prototypes.

|  |  |
| --- | --- |
| Advantages | Disadvantages |
| * Reduced time and costs, but this can be a disadvantage if the developer loses time in developing the prototypes.
* Improved and increased user involvement.
 | * Insufficient analysis. User confusion of prototype and finished system.
* Developer misunderstanding of user objectives.
* Excessive development time of the prototype.
* It is costly to implement the prototypes
 |

#### **Advantages and Disadvantages**

### Spiral Model (SDM)

#### **Description**

It is combining elements of both design and prototyping-in-stages, in an effort to combine advantages of top-down and bottom-up concepts. This model of development combines the features of the prototyping model and the waterfall model. The spiral model is favored for large, expensive, and complicated projects. This model uses many of the same phases as the waterfall model, in essentially the same order, separated by planning, risk assessment, and the building of prototypes and simulations.



#### **The usage**

It is used in the large applications and systems which built-in small phases or segments.

|  |  |
| --- | --- |
| Advantages | Disadvantages |
| * Estimates (i.e. budget, schedule, etc.) become more realistic as work progressed because important issues are discovered earlier.
* Early involvement of developers.
* Manages risks and develops the system into phases.
 | * High cost and time to reach the final product.
* Needs special skills to evaluate the risks and assumptions.
* Highly customized limiting re-usability
 |

#### **Advantages and Disadvantages**

### Iterative and Incremental Model

#### **Description**

It is developed to overcome the weaknesses of the waterfall model. It starts with an initial planning and ends with deployment with the cyclic interactions in between. The basic idea behind this method is to develop a system through repeated cycles (iterative) and in smaller portions at a time (incremental), allowing software developers to take advantage of what was learned during the development of earlier parts or versions of the system. It can consist of mini waterfalls or mini V-Shaped model



#### **The usage**

It is used in shrink-wrap application and large system which built-in small phases or segments. Also, can be used in a system has separated components, for example, ERP system. Which we can start with the budget module as a first iteration and then we can start with the inventory module and so forth.



#### **Advantages and Disadvantages**

|  |  |
| --- | --- |
| Advantages | Disadvantages |
| * Produces business value early in the development lifecycle.
* Better use of scarce resources through proper increment definition.
* Can accommodate some change requests between increments.
* More focused on customer value than the linear approaches.
* We can detect project issues and changes earlier.
 | * Requires heavy documentation.
* Follows a defined set of processes.
* Defines increments based on function and feature dependencies.
* Requires more customer involvement than the linear approaches.
* Partitioning the functions and features might be problematic.
* Integration between the iterations can be an issue if it is not considered during the development and project planning.
 |

### Agile Model

#### **Description**

It is based on iterative and incremental development, where requirements and solutions evolve through collaboration between cross-functional teams.

Scrum Agile Model

#### **The usage**

It can be used with any type of the project, but it needs more engagement from the customer and to be interactive. Also, we can use it when the customer needs to have some functional requirement ready in less than three weeks and the requirements are not clear enough. This will enable more valuable and workable piece for software early which also increase the customer satisfaction.

#### **Advantages and Disadvantages**

|  |  |
| --- | --- |
| Advantages | Disadvantages |
| * Decrease the time required to avail some system features.
* Face to face communication and continuous inputs from customer representative leaves no space for guesswork.
* The end result is the high-quality software in the least possible time duration and satisfied customer.
 | * Scalability.
* The ability and collaboration of the customer to express user needs.
* Documentation is done at later stages.
* Reduce the usability of components.
* Needs special skills for the team.
 |

**Selecting a Software Development** Life Cycle (SDLC) methodology is a challenging task for many organizations and software engineers. What tends to make it challenging is the fact that few organizations know what are the criteria to use in selecting a methodology to add value to the organization. Fewer still understand that a methodology might apply to more than one Life Cycle Model. Before considering a framework for selecting a given SDLC methodology, we need to define the different types and illustrate the advantages and disadvantages of those models (please see [the Software Development Life Cycle Models and Methodologies](https://melsatar.blog/2012/03/15/software-development-life-cycle-models-and-methodologies/)).

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.

As described in [the software development life cycle models](https://melsatar.blog/2012/03/15/software-development-life-cycle-models-and-methodologies/) article, models are similar to the tools that important to know each tool usage to know which context it can fit into.

Imagine the image below by Jacob Lawrence, if the carpenter did not know the tools he will use, what will be the results? Did you visualize the disaster?

### 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

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Factors** | **Waterfall** | **V-Shaped** | **Evolutionary Prototyping** | **Spiral** | **Iterative and Incremental** | **Agile** |
| **Unclear User Requirement** | Poor | Poor | Good | Excellent | Good | Excellent |
| **Unfamiliar Technology** | Poor | Poor | Excellent | Excellent | Good | Poor |
| **Complex System** | Good | Good | Excellent | Excellent | Good | Poor |
| **Reliable system** | Good | Good | Poor | Excellent | Good | Good |
| **Short Time Schedule** | Poor | Poor | Good | Poor | Excellent | Excellent |
| **Strong Project Management** | Excellent | Excellent | Excellent | Excellent | Excellent | Excellent |
| **Cost limitation** | Poor | Poor | Poor | Poor | Excellent | Excellent |
| **Visibility of Stakeholders** | Good | Good | Excellent | Excellent | Good | Excellent |
| **Skills limitation** | Good | Good | Poor | Poor | Good | Poor |
| **Documentation** | Excellent | Excellent | Good | Good | Excellent | Poor |
| **Component reusability** | Excellent | Excellent | Poor | Poor | Excellent | Poor |

### 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.