**Temasek Polytechnic**

**School of Informatics and IT**

**Diploma in Information Technology (IT)**

**Software Development Life Cycle**

**Project Particulars**

|  |  |
| --- | --- |
| **Tutor** | Mdm. Ho Li Chin |
| **Class** | P01 |

**Project Team’s Particulars**

|  |  |
| --- | --- |
| **Matric Number** | **Student Name** |
| 1401529A | Chen Yee Yin (Andrea) |
| 1405307H | Celine Neo Shi Ling |
| 1406183B | Rachael Ong Jing Hui |
| 1402047C | Serena See Lee Ping |

Contents

[*Celine* 3](#_Toc450339137)

[*Rachael* 5](#_Toc450339138)

[*Andrea* 8](#_Toc450339139)

[*Serena* 14](#_Toc450339140)

[*References* 23](#_Toc450339141)

# *Celine*

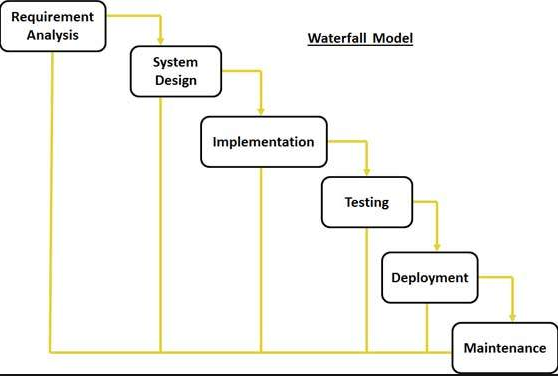
**SDLC Definition**

SDLC, also known as the Software development process, is a framework that defines tasks performed at every step during the software development process. Within a software organization, for software projects, the process SDLC is followed. It describes using a detailed plan, how to develop, maintain, replace and alter/enhance specific software(s). This life cycle defines a set of rules to improve the quality of the software and the overall development process.

**3 Models**

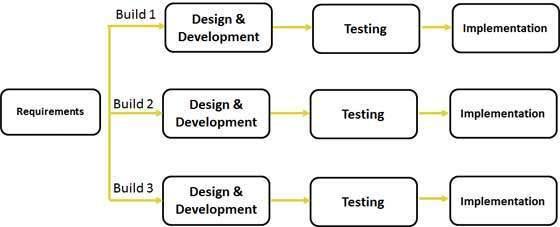
1. Waterfall Model Design

The first SDLC Model used widely in Software Engineering to ensure the success of a project. In this approach, the process of software development is divided into separate phases and typically, the outcome of each phase acts as the input for the next sequential phase.



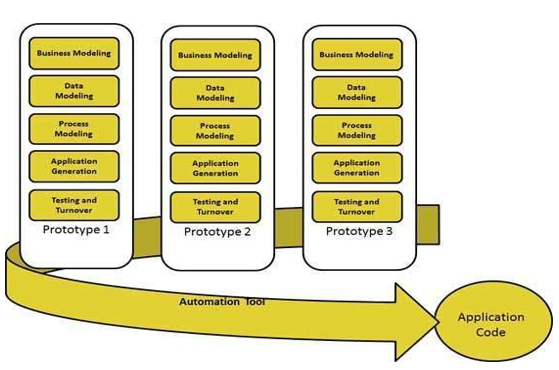
1. Iterative Model

This method basically develops systems through repeated cycles and in smaller portions at a time. Starting with the subset of the software requirements, the Iterative process begins and iteratively enhances the evolving versions until the full system is implemented. At every iteration, design modifications are made and new functional capabilities are added.



1. RAD Model Application

Rapid Application Development (RAD) is a method that uses little planning, favouring rapid prototyping which are working models that are functionally equivalent to a component of the product. Functional modules are developed parallel to each other and later integrated, saving time. With no detailed preplanning, it is thus easier to incorporate the changes within development processes. RAD project follow the iterative and incremental model and have smaller teams consisting of developers, domain experts, customer representatives and other IT resources working progressively on their component(s) and/or prototype(s). For this model to be successful, it is important to make sure the developed prototypes may be reused.



# *Rachael*

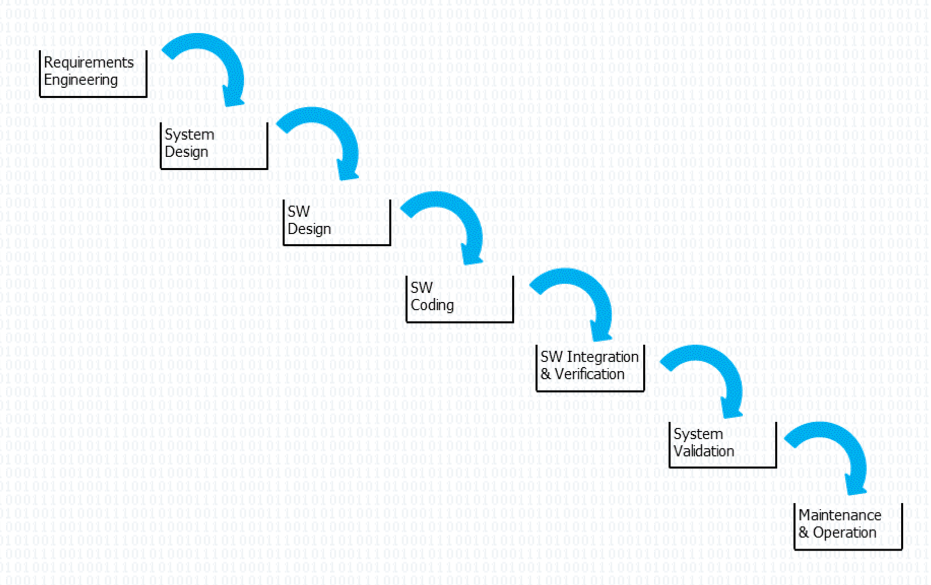
**SDLC Definition**

Software Development Lifecycle is a process used to track the different process during a project. First, it includes evaluating the current system to look for any weakness. Next would be to define the new system requirements and also to address the weakness in the system to look at how to improve it. After that, it would be to design the system where, the physical construction, hardware, operating systems, communications, programming, and security issues are laid out. The new system would then be developed, where new components and programs are to be installed. Users would then have to undergo training; testing and final adjustments would have to be made at this step. The system would then be put to use, this can be done in two ways, where the system would be phrased in according to application or location, gradually replacing the old system; or to shut down the old system and implement the new system all at once. Once the system is up and running, there should be regular maintenance of the system and users should be kept up to date on latest modification and procedures.

**3 Models**

1. Waterfall Model

The waterfall model is sequential following a top down method, earlier phrases deals with requirements then moving on to design followed by coding, implementation phrases and/or testing, following production and maintenance of the product. Each phrase of the waterfall model has a certain set of documents, documents at the front normally serves as inputs for later phrases. Waterfall model requires one phrase to be completed before the next phrase can be continued.

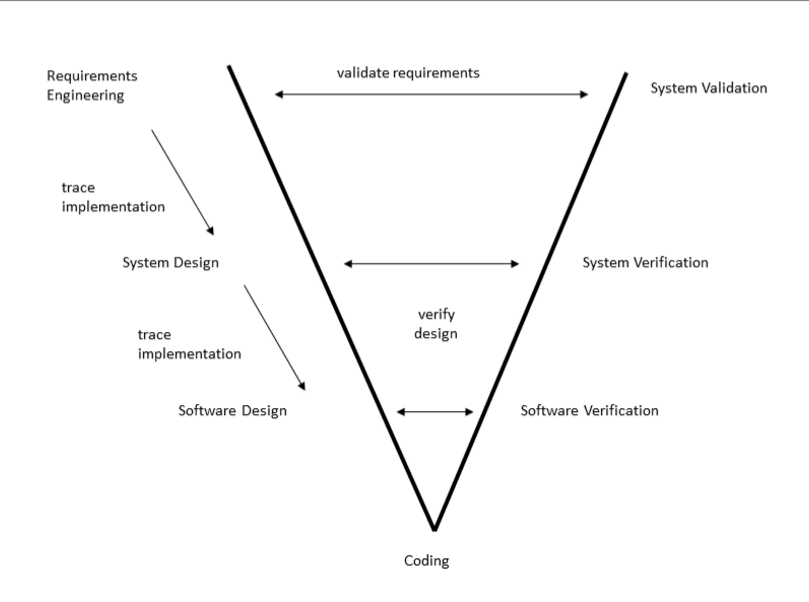


|  |  |
| --- | --- |
| Pro | Cons |
| 1. Different phrases are clearly separated from each other, allowing for better planning & control. 2. Simple & easy to understand. 3. Exact estimation of cost & required effort is possible. | 1. Reality phrases cannot be strictly separated. Iteration loops between subsequent phrases are usually needed. 2. Unable to work in parallel mode due to strict sequential manner. 3. When there is a change in requirements, have to re-do the entire waterfall model. |

1. V Model

The V model is similar to the waterfall model, following a linear process, but bends upwards at the coding phrase to form a V shape, the reason for the line to bend upwards is that the testing phrase would correlate to the design phrase, which make it possible to verify each of the design step individually.

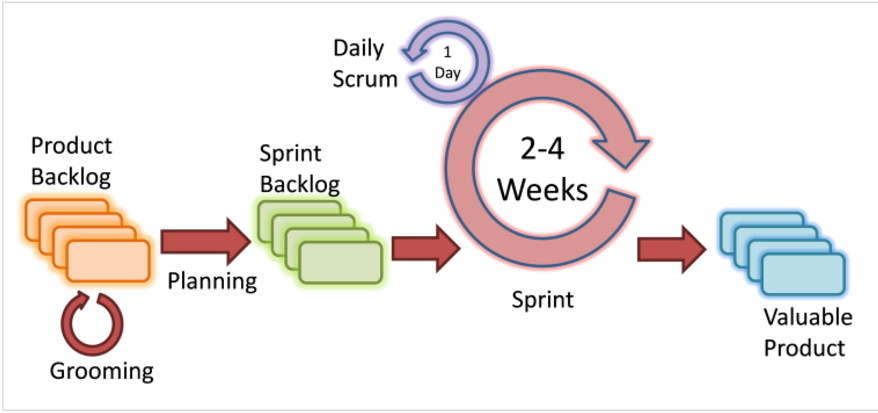
|  |  |
| --- | --- |
| Pro | Cons |
| 1. Phrases are separated from one another, which allows simple planning and control 2. Exact estimation of effort and cost is possible 3. Defined correlation between test and design activities, which has clear definition steps of design and test phrase in their work product | 1. Very rigid and least flexible 2. Reality phrases cannot be strictly separated. Iteration loops between subsequent phrases are usually needed. 3. Unable to work in parallel mode due to strict sequential manner. |



1. Extreme Programming (Agile Development)

Extreme Programming allows for iteration and incremental development, which allows changes in requirements and solutions through the project. The method allows for customer to give quick feedback as they would deliver the system to customers as early as possible to implement changes as suggested.

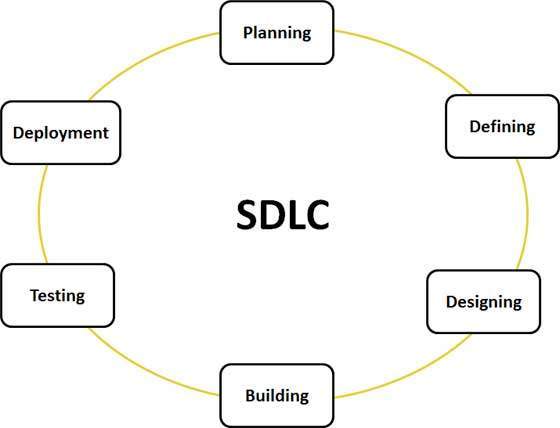
|  |  |
| --- | --- |
| Pro | Cons |
| 1. Decrease the time required to avail some features. 2. Face to face communication with customer allows for valuable feedback, so project changes are never assumed | 1. Scalability issues 2. Requires skilled software developers 3. Documents are done at a later stage |



# *Andrea*

**SDLC Definition**

In order to produce software of quality to meet or go clients’ expectations, software organizations adopt the Software Development Life Cycle to ensure success for the software project on hand. This cycle involves a plan which consists of comprehensive details. Examples of such details are the development, maintenance replacement and enhancement/alteration of the specific software. In summary, the Software Development Life Cycle establishes a methodology dedicated to the software’s quality improvement and its overall development process.



For SDLC, various models are defined and designed to guide the processes of software development. Referred to as *Software Development Process Models*, each individual model encompasses its own unique series of steps. The following are three of such examples:

**3 Models**

1. Big Bang Model

For this SDLC model, it is more commonly followed by small development teams tasked with small projects. The reason why this is so; this model’s focus is the assembly of possible resources in software development and coding, with very minimal planning. In other words, there is no formal development followed. Requirements are comprehended and implemented as they come.

As simple and easy as the Big Bang Theory sounds, it calls for very high risks and also, sporadic changes or misinterpretation in requirements. This can result in undesirable outcomes, such as complete reversal or scraping of projects.

|  |  |
| --- | --- |
| Pro | Cons |
| 1. Simple 2. Minimum to zero planning required 3. Easy management 4. Only few resources needed 5. Offers flexibility to developers 6. Serves as a good learning aid for newbies | 1. High levels of risks and uncertainty 2. Flawed model for complicated and object-oriented products 3. Poor model for long and ongoing projects 4. Misinterpretation of requirements can result in huge expenditure |

1. Waterfall Model

Referred to as a linear-sequential life cycle model, the Waterfall Model was the very first process model to be introduced, and also the earliest SDLC approach used for software development. Since the waterfall model is termed as a linear-sequential life cycle model, it means that each individual phase of the life cycle must be completed before the next can happen. This way, it is ensured that the phases do not overlap.

The sequential phases are as followed:

1. Requirement Gathering and Analysis

At the first stage, a requirement specification document is produced where all the possible requirements of the system are developed and then listed.

1. System Design

Taking the requirement specification document from the previous phase, it is carefully studied during this stage. The system design will be prepared accordingly to the requirements stated in the document. During this phase, the hardware and system requirements are better specified and the overall architecture of the system is defined as well.

1. Implementation

With the complete system design, the next step would be to move on to implementation. The system is first developed in small programs termed units, which will be part of the integration in the following phase. After which, every individual unit will undergo development and ‘Unit Testing’, where units will be tested for their functionalities.

1. Integration and Testing

After the units have undergone ‘Unit Testing’, they will be officially integrated into a system. Following the integration would be another round of testing, where the system is tested for potential faults or failures.

1. Deployment of System

When the system passes the functional and non-functional testing, it will then be deployed in the customer environment or released into the market.

1. Maintenance

In cases where issues arise in the client environment, issue patches will be released to combat them. Newer and better versions will also be formulated and released to ensure its relevance in the market in the long run.



|  |  |
| --- | --- |
| Pro | Cons |
| 1. Simple and easy to understand and use 2. Fixed and stable nature of model allows easy management 3. Completion of phases are systematic 4. Ideal for minor projects where requirements are well interpreted 5. Stages and milestones are clearly defined 6. Easy arrangement of tasks 7. Good documentation of process and results | 1. Operating software is produced only at the later part of the life cycle 2. High levels of risks and uncertainty for projects which involve high chances of change in requirements 3. Flawed model for complicated and object-oriented products 4. Poor model for long and ongoing projects 5. Challenging to measure progress from stage to stage 6. Scope adjustment during life cycle may result in project termination 7. Integration is done in the style of “Big Bang”, meaning that the identification of technological or business challenges cannot be done in the early part. |

1. Spiral Model

This model inherits the systematic and controlled aspects of the waterfall model, both aspects evident from its illustrated iterative development process and sequential linear development. What sets the Waterfall and Spiral model apart is that the latter has a very clear emphasis on risk analysis. With that, a product can be released and refined incrementally, through each iteration around the spiral.

In total, there are four phases of the Spiral Model. Each spiral represents an iteration, and a software project would have to pass through the four phases repeatedly before completion.

The following are the phases of the Spiral Model:

1. Identification

Starting with the baseline spiral, business requirements are collected and as the product passes through subsequent spirals, the sub-system and system requirements are identified. System requirements are understood through consistent communication between the system analyst and the client. The final result of the spiral would be the deployment of the product in an identified market.

1. Design

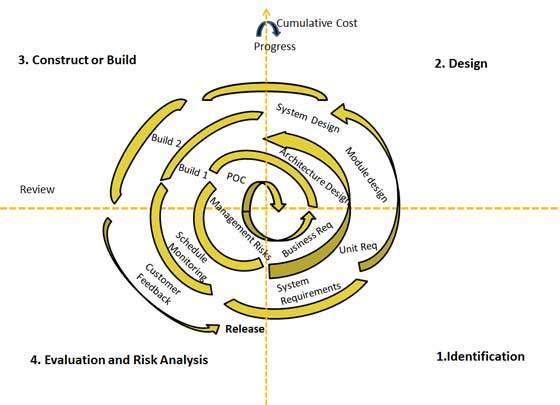
With the focus still on the baseline of the spiral, this stage would begin with a conceptual design. Various types of designs are produced in the subsequent spirals, and they are architectural design, logical design of modules, physical product design and the final design.

1. Construction

Here, the designs of the actual software product of every spiral come to life. At the baseline spiral, a Proof of Concept is produced at this stage to

1. Evaluation/Risk Analysis

Technical feasibility and management risks are identified, estimated and monitored at the risk analysis stage. Some examples of management risks are schedule slippage and cost overrun. After the build is tested, at the end of the initial iteration, the client evaluates the software and gives feedback.



|  |  |
| --- | --- |
| Pros | Cons |
| 1. Easy integration of changes in requirements 2. Allows extensive use of prototypes 3. Requirements can be captured more accurately 4. Better risk management means tasks are carried out according to their level of risk (tasks of higher risk levels are developed earlier) | 1. Management is more complex 2. Prediction of project ending may not be known in advance. 3. Poor model for long and ongoing projects 4. Complex process 5. Spiral may continue indefinitely |

# *Serena*

**SDLC Definition**

Software Development Life Cycle (SDLC) is a conceptual model used in project management that describes the stages involved in an information system development project, from an initial feasibility study to the maintenance of the completed application. It is also sometimes referred to as System Development Life Cycle. IT analysts uses this process in order to develop or redesign high quality software systems which meets both the customer and real world requirements, taking into consideration all associated aspects of pros and cons.

**3 Models**

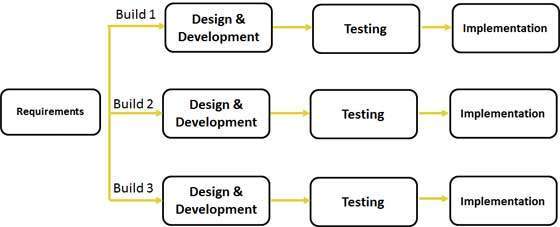
1. Iterative Model

This model starts with an implementation of a small set of the software requirements and iteratively enhances the evolving versions until the complete system is implemented and deployed. An iterative life cycle model does not begin the whole process with the full specification of requirements. Development starts with a simple implementation of a subset of the software, then it is reviewed to strengthen the existing software and also carry on with further requirements of the software. Each iteration would mean more design modification and new functional capabilities would be included. During the development, more than one iteration of the process may be in progress at the same time. This is due to its incremental nature and may be described as an “evolutionary acquisition” approach.

During each iteration, the development module goes through the requirements, design, testing and implementation phrases all over again. Each subsequent release of the module adds new functions to the previous release. It will continue till the whole system is completed and ready as per requirement. Iterative model will only succeed with vigorous validation of requirements, verification and testing of each version of software against the requirements within each cycle of the model. As the software evolve gradually after many successive iterations, the tests have to be repeated and extended to verify each version of the software.

This model is often used in the following scenarios:

* Requirements of the complete system are clearly defined and understood
* There is a time to the market constraint
* Resources with needed skill set are not available and are planned to be on contract basis for specific iterations.



|  |  |
| --- | --- |
| Pro | Cons |
| 1. Some working functionality can be developed quickly and early in the life cycle 2. Parallel development can be planned 3. Progress can be measured 4. Less costly to change scope or requirements 5. Testing and debugging each iteration is easy 6. Risk analysis is better 7. Software is produced early which facilities customer evaluation and feedbacks | 1. More resources may be required 2. System architecture or design issues may arise as not all requirements are gathered in the beginning of the entire life cycle 3. Management complexity is higher 4. End of project may not be known which may pose as a risk 5. Progress is highly dependent on the risk analysis phase |

1. V Model

The V model is the execution of processes that happens in a sequential manner in a V shape. It is also known as Verification and Validation model. It is the extension of the waterfall model and is based on association of a testing phase for each corresponding development stage. For every single phase, there will be a testing done. It follows a very strict order where the next phrase will only start after the completion of the previous phase. Under the V model, the corresponding testing phase of the development phase is planned in parallel. There are verification phases on one side and Validation on the other side. The coding phase joins the two sides of the model.

1. *Verification Phase:*
   1. Business Requirement Analysis

This is the first phase is the cycle where the product requirements are understood from the customer’s perspective. The acceptance test design planning is also done at this stage as business requirements can be used as an input for acceptance testing.

* 1. System Design

The complete system is designed at this phase. It includes understanding under development. The system test plan is developed based on the system design.

* 1. Architectural Design

Architectural specifications are understood and designed in this phase. More than one technical approach will be proposed and based on the technical and financial feasibility, the final decision is taken. System design is broken down into modules that takes up different functionality. This is also referred as High Level Design (HLD).

The data transfer and communication between internal modules and other systems will be planned. With the information, integration tests can be designed and documented during this stage.

* 1. Module Design

This is the phase where detailed internal design is specified. It is also referred to as Low Level Design (LLD). It is essential that the design is compatible with the other modules in the system architecture and other external systems. Unit tests is crucial in any development process as it helps to detect utmost faults and errors in the early stages of development.

1. *Coding Phase:*

The actual coding of the modules designed earlier in the design phase is done during the coding phase. The suitable programming language is decided on based on the system and architecture requirements. The codes are constructed based on the coding guidelines and standards. It will be reviewed over and over again and amended till it is satisfactory before the final build is checked into its repository.

1. *Validation Phase:*
   1. Unit Testing

Unit tests designed in the module design phase are executed on the code during this validation phase. Unit testing is the testing at code level and helped to eliminate bugs at an early stage.

* 1. Integration Testing

Integration testing is associated with the architectural design phase. Integration tests are performed to test the coexistence and communication of the internal modules within the system.

* 1. System Testing

This testing is directly associated with the system design phase. System tests check the entire system functionality and the communication of the system under developments with external systems. Most software and hardware compatibility issues can be uncovered during the test execution.

* 1. Acceptance Testing

Acceptance testing is associated with the business requirement analysis phase and involves testing the product in user environment. Acceptance tests uncover the compatibility issues with the other systems available in the user environment. It also discovers the non-functional issues such as load and performance defects in the actual user environment.

This model is often used in the following scenarios:

* Product definition is stable
* No ambiguous or undefined requirements
* Technology is not dynamic and is well understood by the project team



|  |  |
| --- | --- |
| Pro | Cons |
| 1. A highly disciplined model and phases are completed one at a time 2. Simple and easy to understand and use 3. Easy to manage due to the rigidity of the model. Each phase has specific deliverables and a review process. | 1. High risk and uncertainty 2. Not a good model for complex and object oriented projects. 3. Once the software reaches testing stage, it is very hard to go back and change a functionality 4. No working software is produced until late during the life cycle |

1. Agile Model

Agile model is a combination of iterative and incremental process models with focus on process adaptability and customer satisfaction by rapid delivery of working software products. It breaks down the product into small incremental builds. These builds are provided in iterations. Each iterations involves cross functional teams working simultaneously on areas like planning, requirements analysis, design, coding, unit testing and acceptance testing. Each iteration will typically last about one to three weeks. Agile method believes in catering to suit each project’s requirements best, thus the tasks are divided into small time frames to deliver specific features for a release. Each build is incremental in terms of features. The final build will then hold all the features require by the customer.

There are many different kind of agile methods out in the market. For example:

* Scrum
* Crystal Clear
* Extreme Programming
* Adaptive Software Development (ASD)
* Feature Driven Development (FDD)
* Dynamic Systems Development Method (DSDM)

But now, they are collectively known as agile methodologies after the Agile Manifesto was published in 2001. The Agile Manifesto is a formal proclamation of the 4 key values and 12 principles to guide an iterative and people-centric approach to software development.

*4 values of Agile Manifesto:*

* Individual & Interactions

Self-organization and motivation is really important, as are interactions like co-location and pair programming.

* Working Software

Workable demo software is considered the best means of communication with the customer to understand their requirements, instead on basing just on documentation.

* Customer Collaboration

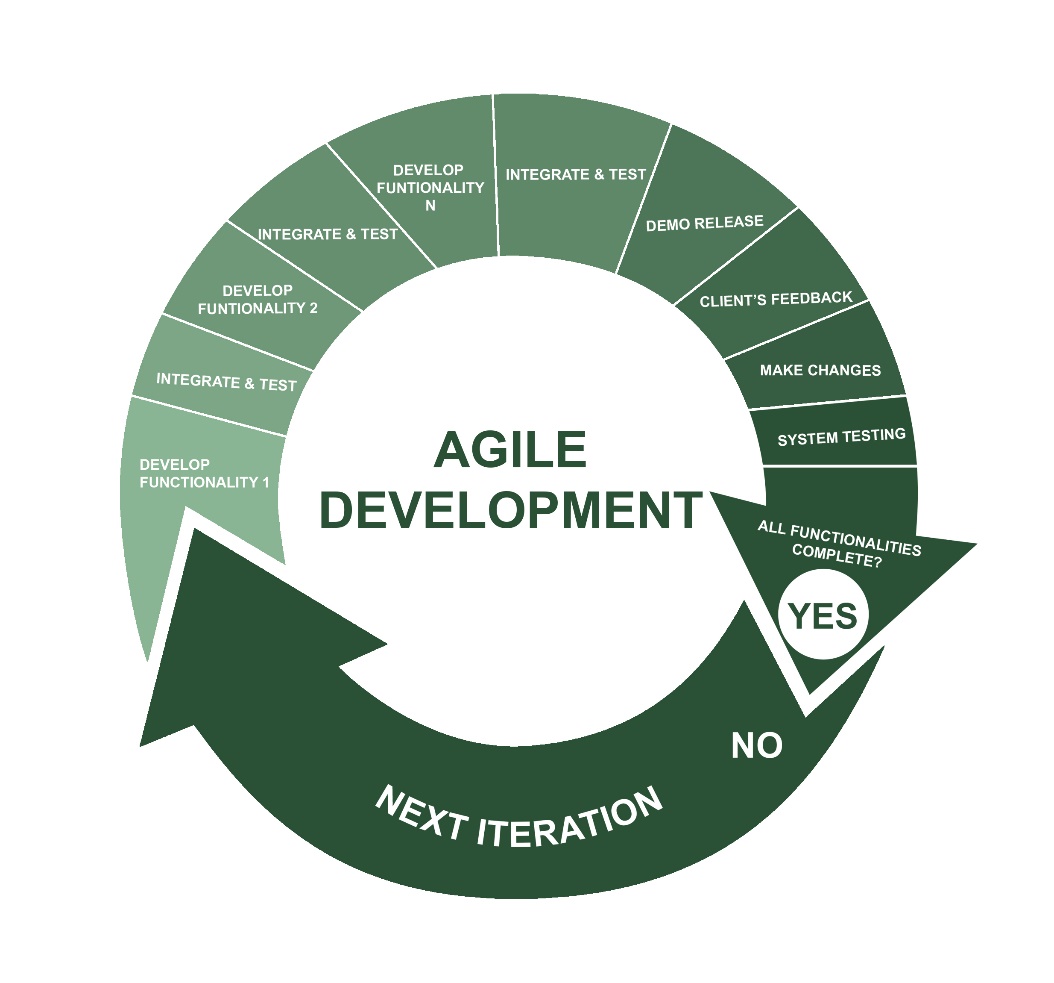
It is not possible to gather all the requirements at the start of the project due to various reasons, thus it is important for continuous customer interactions to get the proper product requirements.

* Responding to Change

Agile is focused on quick responses to changes and continuous development.

*12 principles of Agile Manifesto:*

1. Highest priority is to satisfy customers through early and continuous delivery of valuable software.
2. Welcome changing requirements, even in late development. Agile processes harness change for the customer’s competitive advantage
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face to face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity – the art of maximizing the amount of work not done – is essential.
11. The best architectures, requirements and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjust its behaviour accordingly.



*Agile vs Traditional SDLC Models*

|  |  |
| --- | --- |
| Agile | Traditional |
| 1. Uses adaptive approach 2. No detailed planning & only has clarity on future tasks in respect of what is to be developed 3. There is feature driven development & the team adapts to the changing product requirements dynamically 4. Tested frequently, through the release iterations | 1. Uses predictive approach 2. Detailed planning & complete forecast of exact tasks & features to be delivered during the life cycle 3. Predictive methods entirely depend on requirement analysis and planning done in the beginning 4. Any changes to be incorporated o through a strict change control management and prioritization |

|  |  |
| --- | --- |
| Pro | Cons |
| 1. A realistic approach to software development 2. Promotes teamwork and cross training 3. Resource requirements are minimum 4. Suitable for fixed or changing requirements 5. Little or no planning required 6. Easy to manage 7. Give flexibility to developers | 1. Not suitable for handling complex dependencies 2. More risk of sustainability, maintainability and extensibility 3. Depends heavily on customer interaction, so if customer is not clear, team can be driven in the wrong direction 4. There is very high individual dependency, since there is minimum documentation generated |

# *References*

Celine:

|  |  |
| --- | --- |
| Waterfall Model: | <http://www.tutorialspoint.com/sdlc/sdlc_waterfall_model.htm> |
| Iterative Model: | <http://www.tutorialspoint.com/sdlc/sdlc_iterative_model.htm> |
| RAD Model: | <http://www.tutorialspoint.com/sdlc/sdlc_rad_model.htm> |

Rachael:

|  |  |
| --- | --- |
| Waterfall Model: | <http://www.the-software-experts.com/e_dta-sw-process-model-waterfall.php> |
| V Model: | <http://www.the-software-experts.com/e_dta-sw-process-model-V.php> |
| Extreme Programming (Agile Development) | <https://melsatar.wordpress.com/2012/03/15/software-development-life-cycle-models-and-methodologies/> |

Andrea:

|  |  |
| --- | --- |
| Big Bang: | <http://www.tutorialspoint.com/sdlc/sdlc_bigbang_model.htm> |
| Waterfall Model: | <http://www.tutorialspoint.com/sdlc/sdlc_waterfall_model.htm> |
| Spiral Model: | <http://www.tutorialspoint.com/sdlc/sdlc_spiral_model.htm> |

Serena:

|  |  |
| --- | --- |
| Definition: | <http://searchsoftwarequality.techtarget.com/definition/systems-development-life-cycle>  https://www.sdlc.ws/what-is-sdlc/ |
| Iterative Model: | http://www.tutorialspoint.com/sdlc/sdlc\_iterative\_model.htm |
| V Model: | <http://www.tutorialspoint.com/sdlc/sdlc_v_model.htm> |
| Agile Model: | <http://www.tutorialspoint.com/sdlc/sdlc_agile_model.htm>  <https://www.agilealliance.org/agile101/12-principles-behind-the-agile-manifesto/> |