Temasek Polytechnic
School of Informatics and IT

Diploma in Information Technology (IT)

Compiled Individual SDLC

Project Particulars

Tutor	Mr Mel
Class	P04
Project Title	Delonix Regia Hotel Management System

Project Team's Particulars

Matric Number	Student Name
1406932B	**Daniel Toh**
1400797E	**Edmund Yeo**
1407193J	**Dominic Ng**
1405387J	**Vivian Tan**

Table of Contents

Daniel Toh (1406932B)

What is Software Development Life Cycle? (Daniel)

Software Development Life Cycle is a process usually used in a software project within a software organization. The cycle consists of a detailed plan which describe the ways to develop, maintain, replace and alter or enhanced a specific software. This life cycle defines how will we be improving the quality of software and how will the software development process be.

What does SDLC mean?

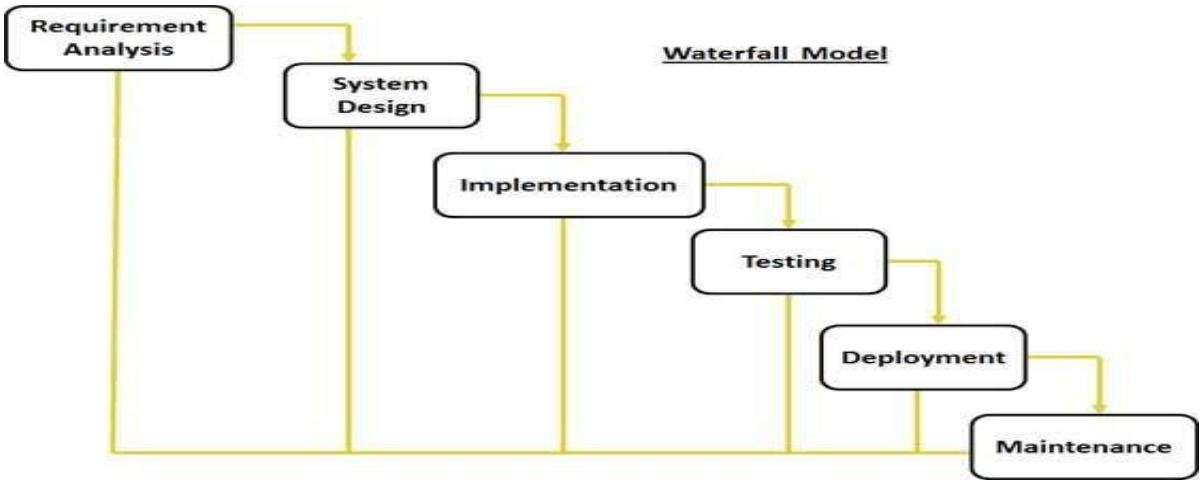
SDLC stands for Software Development Life Cycle or it can also be called the software development process.

3 Example of Development Software Model Life Cycle

1. Waterfall Model
2. Agile Model
3. Spiral Model

Waterfall Model

Waterfall model is the first process model to be introduced for the SDLC. In this approach, the whole process of developing the software is divided into separate phases in an easier term is the outcome of the first phase will act as the input for the second phase so it's a more sequentially way.

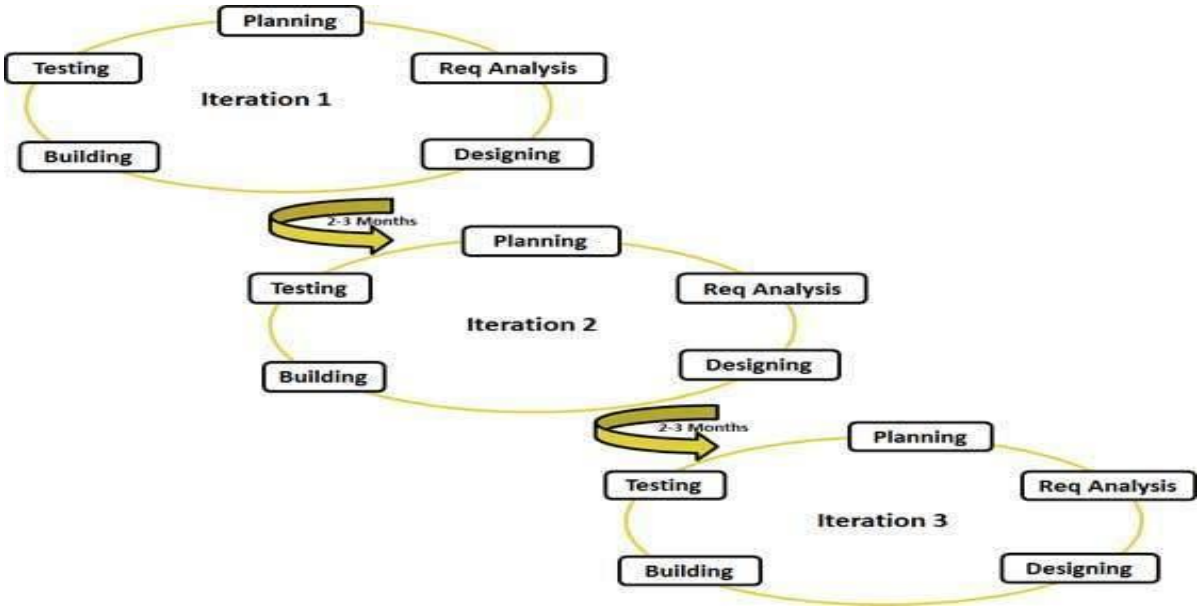


Reference: http://www.tutorialspoint.com/sdlc/sdlc_waterfall_model.htm

Pros	**Cons**
Simple and Easy to understand	No working software is produced until late cycle
Easy to arrange task	High amounts of risks
Well understood milestones	Not suitable for long or ongoing projects`

Agile Model

Agile model is one of the most popular used model in any software development project. Agile model is a combination of iterative and incremental process models which focus on adaptability and customer satisfaction by delivering a working software product.

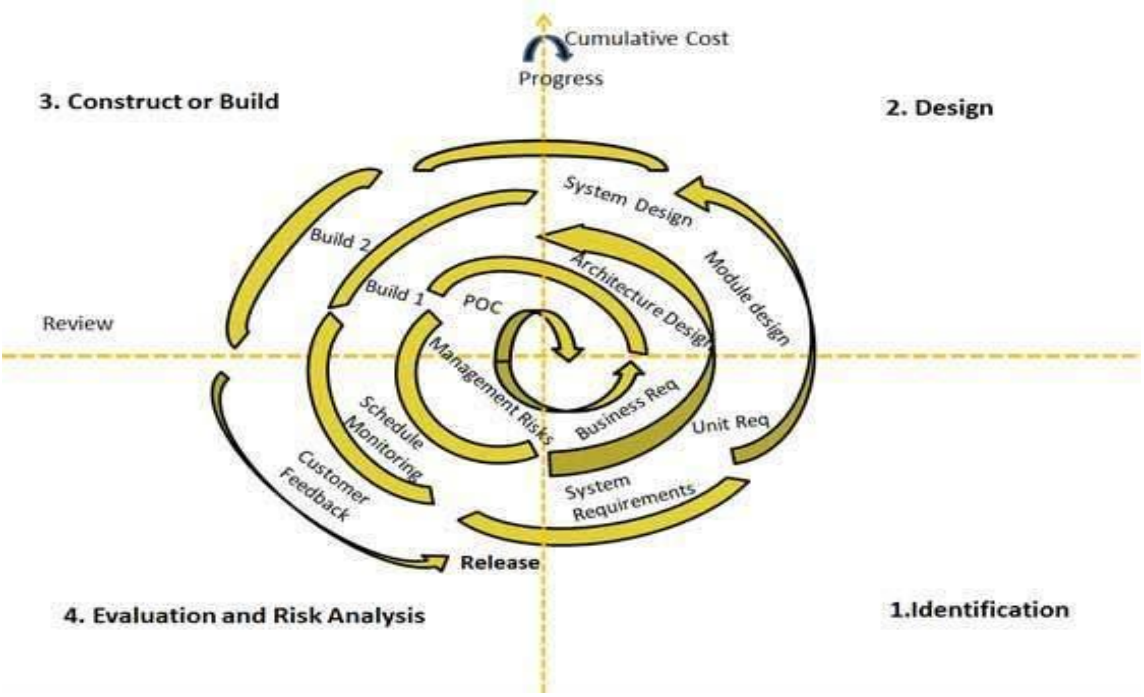


Reference: *http://www.tutorialspoint.com/sdlc/sdlc_agile_model.htm*

Pros	**Cons**
Promotes teamwork	More Risk of maintainability
Intense to work on the software	Depends heavily on customer interaction
Good model for changing environment	Lack of documentation

Spiral Model

Spiral model is the combination of iterative development process and sequential development model which is same as the waterfall model. This cycle allows for incremental releases of the product or refinement through each iteration. The spiral model possesses four phases which are Identification, Design, Construct or Build, Evaluation and Risk Analysis.



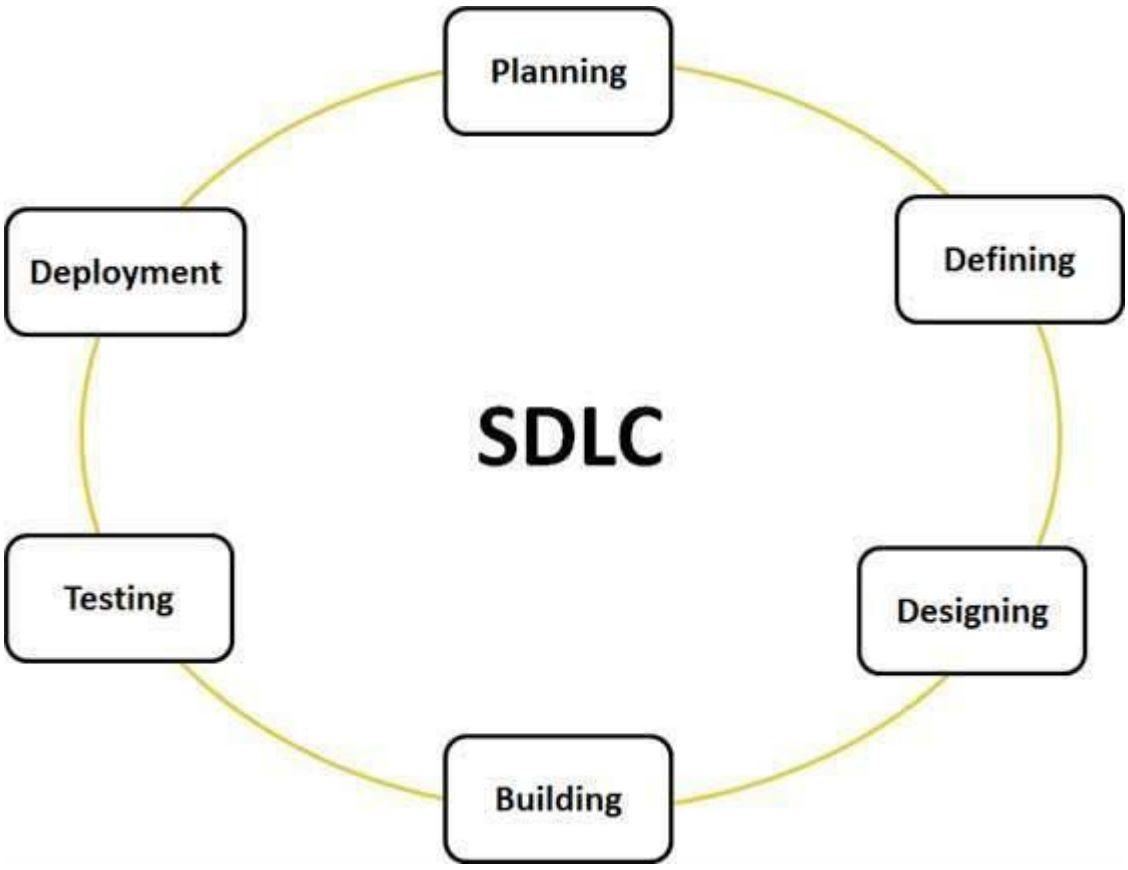
Pros	Cons
Development can be divided into parts	Management is more complex
Users gets to see the prototype of the product	Not suitable for small or low risk project
Requirement is captured accurately	No early definite date of the project completion

Edmund Yeo (1400797E)

What is Software Development Life Cycle? (Edmund)

Systems development life cycle (SDLC), also referred or equal to application development life-cycle, is a term used in system engineering, information systems and software engineering to describe a process for planning, creating, testing, and deploying an information system. The system development life-cycle concept applies to a range of hardware and software configurations, as a system can be composed of hardware only, software only, or a combination of both.

A system development life cycle is composed of a number of clearly defined and distinct work phases which are used system engineers and systems developers to plans for, design, build, test and deliver information systems that meet or exceed customer expectations, based on customer requirements, by delivering systems which move through each clearly defined phase, within scheduled time-frames and cost estimates. Computer systems are complex and often (especially with the recent rise of service-oriented architecture) link multiple traditional systems potentially supplied by different software vendors. To manage this level of complexity, a number of SDLC models or methodologies have been created, such as "waterfall"; "spiral"; "Agile software development"; "rapid prototyping"; "incremental"; and "synchronize and stabilize".



Stage 1: Planning and Requirement Analysis

Requirement analysis is one of the most important and the primary stage in Software Development Life Cycle (SDLC). It is performed normally by the senior members of the team with requirements from the customer, the sales department, market surveys and domain experts in the industry. This information is then used to plan the basic project approach and to conduct product feasibility study in the economical, operational, and technical areas.

Planning for the quality assurance requirements and identification of the risk associated with the project is also done in the planning stage. The outcome of the feasibility of the technical specs and study is to define the various technical approaches that can be followed to implement the project successfully within the minimum risks.

Stage 2: Defining Requirements

Once the first stage analysis is done, the next step is to clearly define and document the product requirements and get them approved from the customer or the market analysis. This is done through software requirement specification document which consist of all the product requirement to be designed and developed during the project life cycle.

Stage 3: Designing the product architecture

The SRS is the reference for the product architects to come out with the best architecture for the product to be developed. Based on the requirement specified in SRS, usually more than one design approach for the product architecture is proposed and documented in a DDS – design document specification.

This DDS is reviewed by all the important stakeholders and based on various parameters as risk assessment, product robustness, design modularity, budget and time constraints, the best design approach is selected for the product.

A design approach clearly defines all the design modules of the product along with its communication and data flow representation with the external and third party module (if any). The internal design of all the modules of the proposed architecture should be clearly defined with the minutest of the details in DDS.

Stage 4: Building or Developing the Product

In this stage: building or developing the product, the actual development starts and the product is built. The programming code is generated as per DDS during this stage. If the design is performed in a detailed and organized manner, code generation can be accomplished without much hassle.

Developers have to follow the coding guidelines defined by their organization and programming tools like compilers, interpreters, debuggers etc are used to generate the code. Different high level

programming languages such as C, C++, Pascal, Java, and PHP are used for coding. The programming language is chosen with respect to the type of software being developed.

Developers have to follow the coding guidelines defined by their organization and programming tools like compilers, interpreters, debuggers etc are used to generate the code. Different programming languages such as C, C++, Pascal, Java, and PHP are used for coding

Stage 5: Testing the Product

This stage is usually a subset of all the stages as in the modern SDLC models, the testing activities are mostly involved in all the stages of SDLC. However, this stage refers to the testing only stage of the product where products defects are reported, tracked, fixed and retested, until the product reaches the quality standards defined in the SRS.

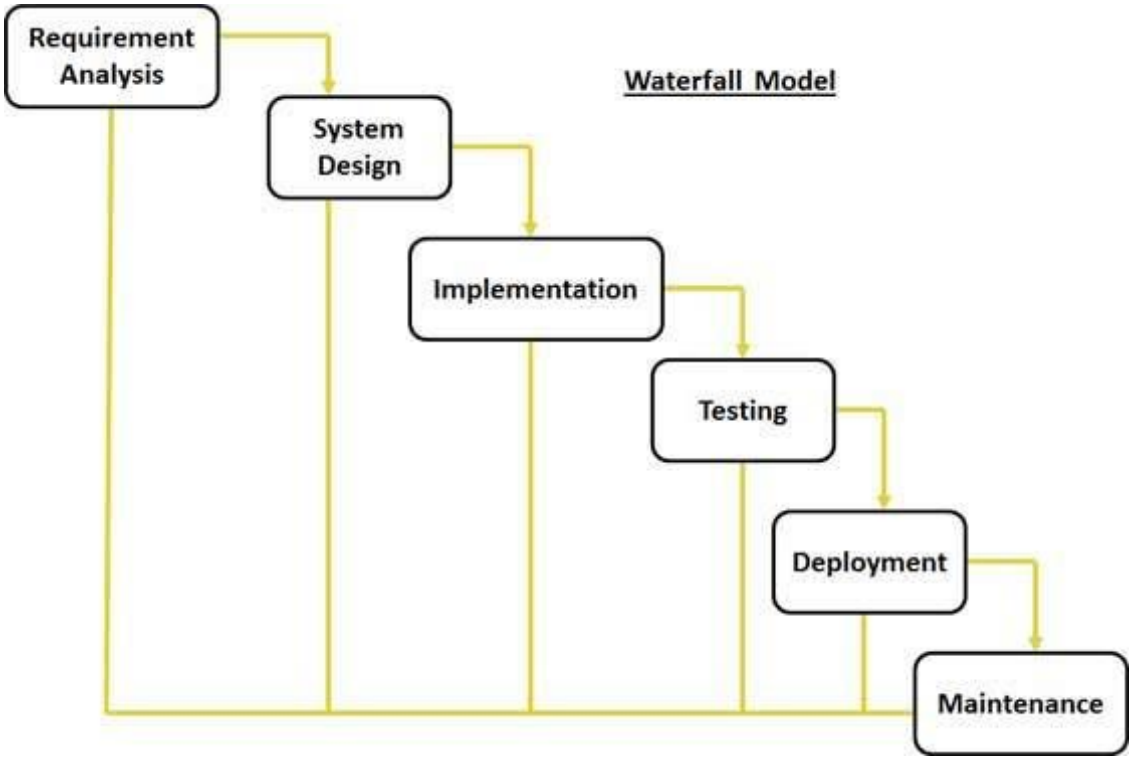
Stage 6: Deployment in the Market and Maintenance

Once the product is tested and ready to be deployed it is released formally in the appropriate market. Sometime product deployment happens in stages as per the organizations. business strategy. The product may first be released in a limited segment and tested in the real business environment (UAT- User acceptance testing).

Then based on the feedback, the product may be released as it is or with suggested enhancements in the targeting market segment. After the product is released in the market, its maintenance is done for the existing customer base.

Waterfall model

Waterfall model approach was the first SDLC Model to be used widely in software engineering to ensure success of the project. In this approach, the whole process of the software development is divided into separate phases. In the water model, typical dependent the outcome of the previous phase before starting the next phase.



Waterfall Model Application

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.
- Simple and straight forward requirements
- Ample resources with required expertise are available to support the product.
- Short project

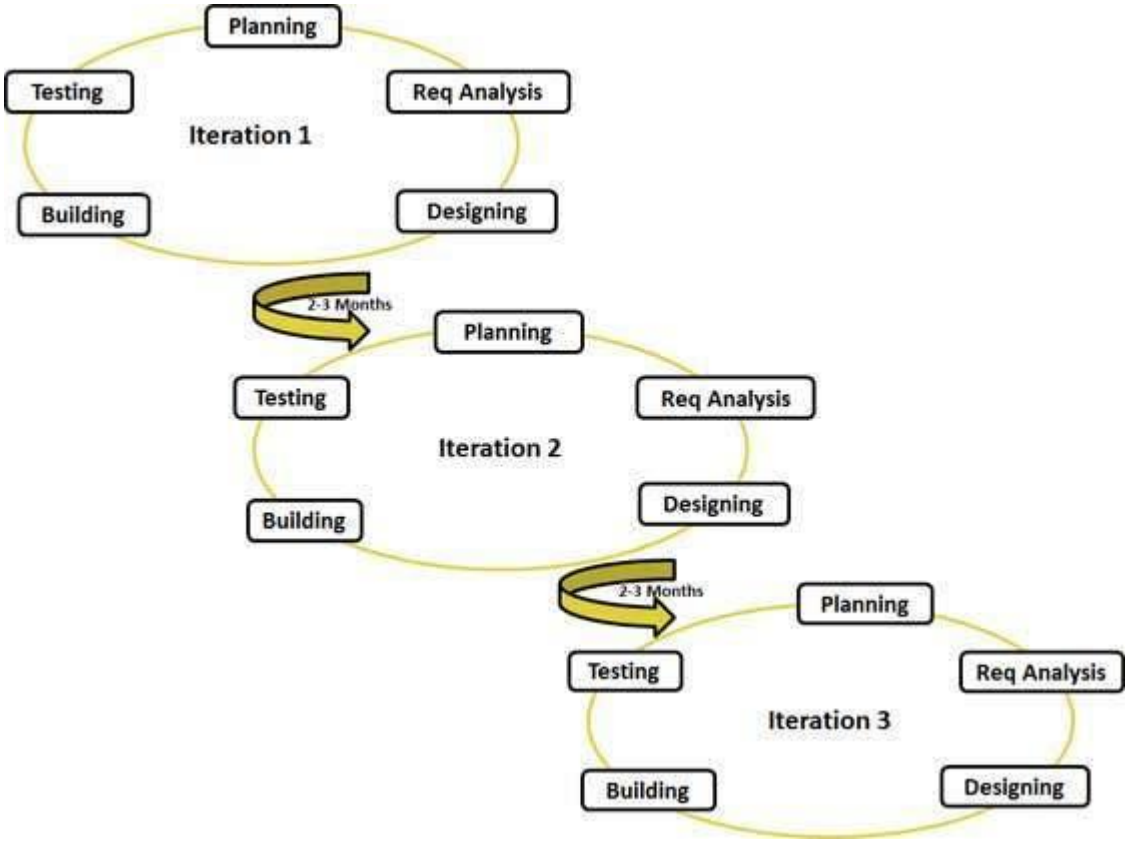
Pros and Cons

Pros	*Cons*
• 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.	• 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 bottleneck or challenges early.

Agile model

Agile model is a model in which handle each project requirements differently. 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.



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 customer to understand their requirement, 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.

Pros and Cons

Pros	***Cons***
• *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.* • *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*	• 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 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.

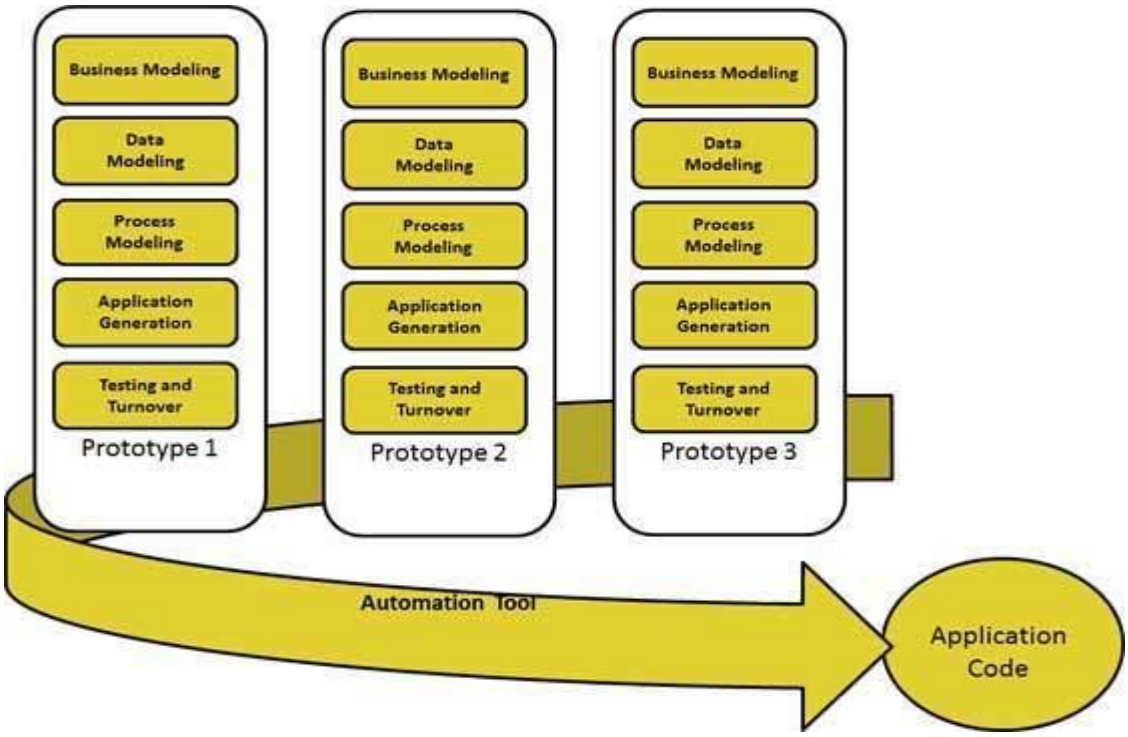
RAD model

Rapid application development (RAD) is a software development methodology that uses minimal planning in favor of rapid prototyping. A prototype is a working model that is functionally equivalent to a component of the product.

In RAD model the functional modules are developed in parallel as prototypes and are integrated to make the complete product for faster product delivery.

Since there is no detailed preplanning, it makes it easier to incorporate the changes within the development process. RAD projects follow iterative and incremental model and have small teams comprising of developers, domain experts, customer representatives and other IT resources working progressively on their component or prototype.

The most important aspect for this model to be successful is to make sure that the prototypes developed are reusable.



RAD Model Design

- **Business Modeling:** The business model for the product under development is designed in terms of flow of information and the distribution of information between various business channels. A complete business analysis is performed to find the vital information for business, how it can be obtained, how and when is the information processed and what are the factors driving successful flow of information.

- **Data Modeling:** The information gathered in the Business Modeling phase is reviewed and analyzed to form sets of data objects vital for the business. The attributes of all data sets is

identified and defined. The relation between these data objects are established and defined in detail in relevance to the business model.

- **Process Modeling:** The data object sets defined in the Data Modeling phase are converted to establish the business information flow needed to achieve specific business objectives as per the business model. The process model for any changes or enhancements to the data object sets is defined in this phase. Process descriptions for adding , deleting, retrieving or modifying a data object are given.

- **Application Generation:** The actual system is built and coding is done by using automation tools to convert process and data models into actual prototypes.

- **Testing and Turnover:** The overall testing time is reduced in RAD model as the prototypes are independently tested during every iteration. However the data flow and the interfaces between all the components need to be thoroughly tested with complete test coverage. Since most of the programming components have already been tested, it reduces the risk of any major issues.

RAD model Application

- RAD should be used only when a system can be modularized to be delivered in incremental manner.

- It should be used if there's high availability of designers for modelling.

- It should be used only if the budget permits use of automated code generating tools.

- RAD SDLC model should be chosen only if domain experts are available with relevant business knowledge.

- Should be used where the requirements change during the course of the project and working prototypes are to be presented to customer in small iterations of 2-3 months.

Pros and Cons

Pros	*Cons*
• Changing requirements can be accommodated. • Progress can be measured. • Iteration time can be short with use of powerful RAD tools. • Productivity with fewer people in short time. • Reduced development time. • Increases reusability of components • Quick initial reviews occur • Encourages customer feedback • Integration from very beginning solves a lot of integration issues.	• Dependency on technically strong team members for identifying business requirements. • Only system that can be modularized can be built using RAD. • Requires highly skilled developers/designers. • High dependency on modeling skills. • Inapplicable to cheaper projects as cost of modeling and automated code generation is very high. • Management complexity is more. • Suitable for systems that are component based and scalable. • Requires user involvement throughout the life cycle. • Suitable for project requiring shorter development times.

Dominic Ng (1407193J)

What is Software Development Life Cycle? (Dominic)

SDLC is a process that is followed for software project within a software organization. SDLC consists of detailed plan that is describing how to develop, replace and alter, maintain or enhance specific software. The life cycle methodology is to improve the quality of software and overall development process.

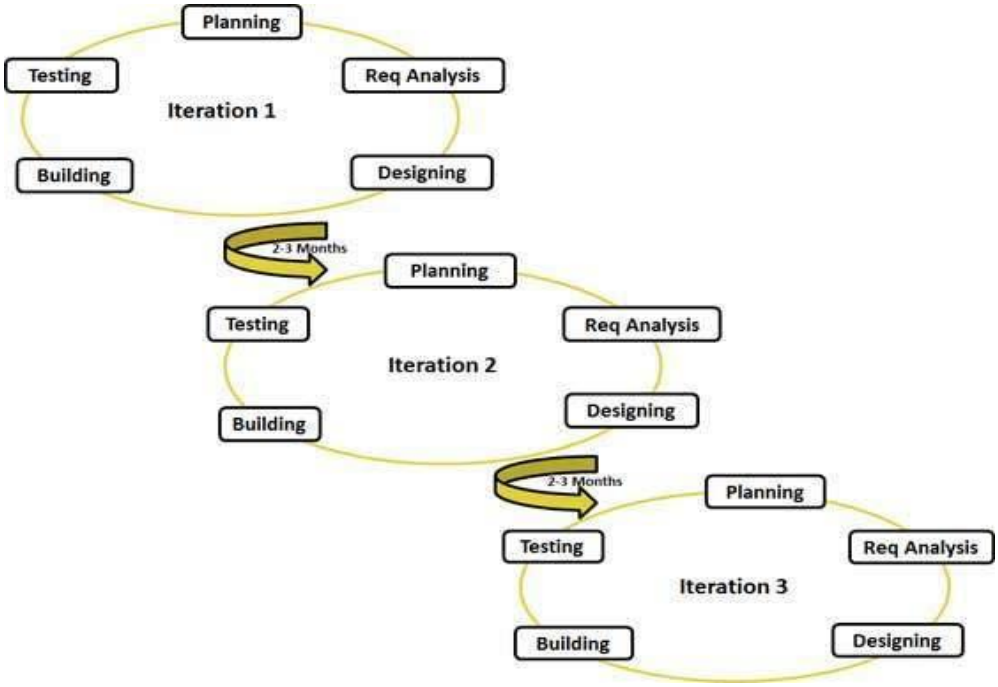
What does SDLC mean?

Systems Development Life Cycle (SDLC) it is also called as Software development process.

3 Example of Development Software Model Life Cycle

Agile Model

Agile had started early in software development and started to become popular with time due to its flexibility and adaptability, agile tasks are divided to time boxes to deliver specific features for a release and agile believes that every project needs to be handle differently as the existing methods need to be shape to the best suit to meet the project requirements.

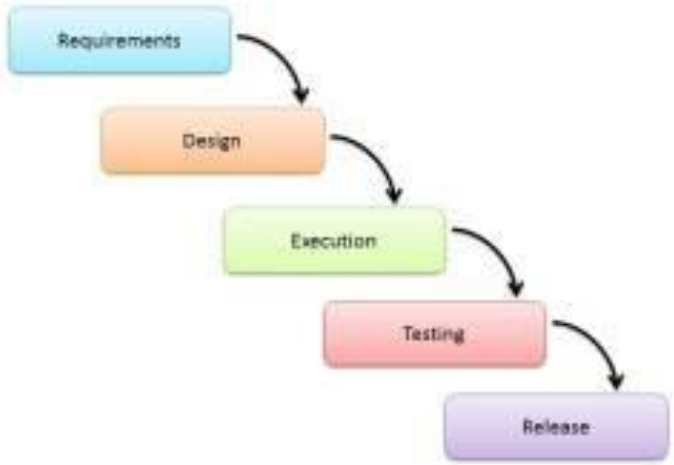


http://www.tutorialspoint.com/sdlc/sdlc_agile_model.htm

Pros	Cons
Easy to manage.	Not suitable for handling complex dependencies.
Gives flexibility to developers.	Strict delivery management dictates the scope, functionality to be delivered and adjustments to meet the deadlines.
Functionality can be developed rapidly and demonstrated.	More risk of sustainability, maintainability and extensibility.
Suitable for fixed or changing requirements.	Transfer of technology to new team members may be quite challenging due to lack of documentation.

Waterfall Model

Waterfall model is a sequential flow, which progress is soon as flowing steadily downwards through the phases of software implementation, means that any phase in development process can only begin if the previous phase is completed. Waterfall cannot go back to the previous phase to make any changes if it is require.

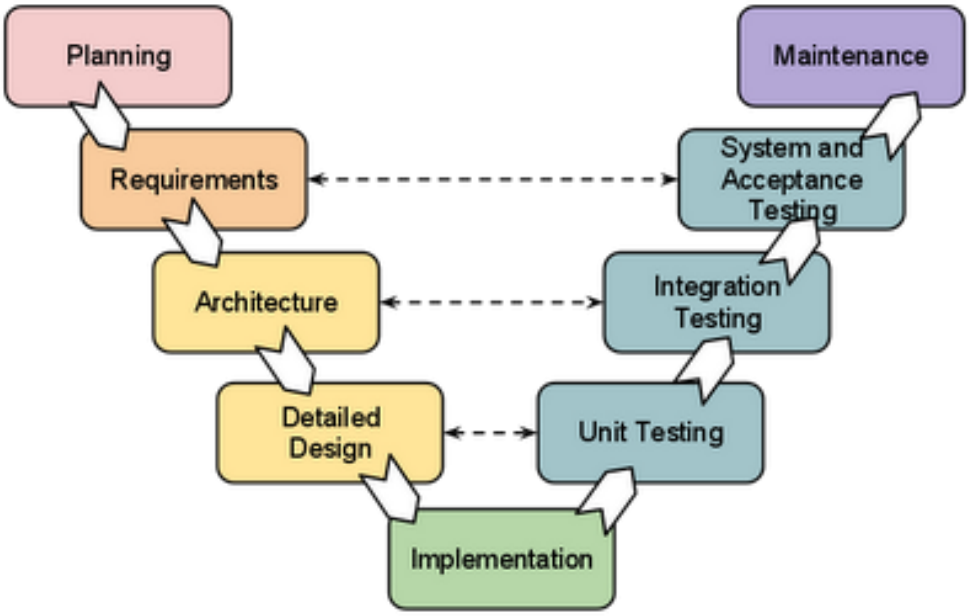


https://melsatar.wordpress.com/2012/03/15/software-development-life-cycle-models-and-methodologies/

Pros	Cons
Easy to explain to users.	Very difficult to go back to any stage after finished.
Each phase has specific deliverables.	Little flexibility and adjusting scope is difficult.
Verification at each stage ensures early detection of errors or misunderstanding.	Expensive.
Helps to plan and schedule project.	Require more time, in addition to detailed plans.

V- Shaped Model

V-Shaped Model is an extension for waterfall model, instead of moving down in a linear way, the v-shaped model process steps are bent upwards after the coding phase. The major difference between v-shaped and waterfall is the early test planning in v-shaped model.



https://melsatar.wordpress.com/2012/03/15/software-development-life-cycle-models-and-methodologies/

Pros	Cons
Simple and easy to use	Inflexible
Higher chance of success over waterfall model due to development test plans early on life cycle.	Model doesn't provide a clear path for problems found during testing phases.
Works well for requirement are easily understood. Verification and validation of product in early stages of product development.	Costly and required more time.
Each phase has specific deliverables.	Software is developed during the implementation phase therefore no early prototypes of software are produced.

Vivian Tan Li Ying (1405387J)

Software Development Life Cycle

Software Development Life Cycle (SDLC) is defined as a framework defining tasks performed at each step in the software development process. It is a structure followed by a development team within the software organization. It consists of a detailed plan describing how to develop, maintain and replace specific software. It aims to produce high quality software that meets or exceeds requirements and expectations needed within given time and budget.

Phases of SDLC

The life cycle of SDLC is also known as the phases in SDLC that clearly defines the sequence of stages in software engineering to develop a software. It is normally planned in 6 phases which will be explained below.

Phase 1: Planning and Requirement Analysis.

The first phase will be worked on by the software development team upon receiving request for producing a desired software product. The first phase is the foundation and most crucial phase of the life cycle as it is the fundamental stage where the project is built on.

The team will have discussions with inputs from customer, sales department, market surveys and various domain experts in the industry. The information collected from various stakeholders will then be used to plan the basic project structure and to conduct the feasibility study of the product in economical, operational, and technical aspects.

In other words, the information are significant and will be taken into consideration when developing the software in order to produce a product that fits the requirement best and appears as most appealing to users. This leads to the quality assurance requirements planning and the identification for the risks associated with the project which will also be done in the first phase.

As there are various technical approaches when developing a software, the result for technical feasibility study will be used to define which technical approach can be utilized with minimum risks. Doing the feasibility study allows the team to analyse whether the software can be made to fulfil all requirements stated by customer, or the possibility of the software having insignificance existence.

Phase 2: Defining Requirements

Upon completing the first phase, planning and requirement analysis, the next phase is to specifically define and document product requirements and get them approved by customer and market analysts.

Meeting the requirements specify by the customer is important in order to satisfy the customer. However, in order to maximize the full potential of the product in the market and among users, market analyst plays an important role in improving and altering requirements to not only satisfy customers but also maximizing the potential of the software.

This can be done through Software Requirement Specification document (SRS) which contains of all product requirements to be designed and developed during the life cycle.

Phase 3: Designing the Product Architecture

The SRS then serves as a reference for product architects to bring forth the best architect for the product to be developed. Commonly, based on the requirements stated in SRS, there will be more than one design approach for the proposed product architect and it will be documented in Design Document Specification (DDS).

The DDS will be scanned by various significant parties, such as developers, involved. They will decide the best design approach based on various factors such as assessment of risks, product robustness, design, and budget and time restrictions.

A design approach will clearly define all the architectural modules of the product together with its communication and data flow representation with any existing external party. The internal design of all modules of the proposed architect should be clearly defined with utmost details in DDS.

Phase 4: Developing the Software

This phase marks the definite start of the development stage where product is being built. The DDS plays an extremely crucial role in this phase as programming code is generated from DDS in this phase. If the design is completed in detailed and systematic manner, the code generation can be accomplished without much hassle. However, if the DDS lacks details and is done without a systematic structure, the hassle incurred will be a hurdle. Hence it was mentioned in Phase 3 that the design must be defined with utmost details.

Different coding languages can be used when developing such as C++, Java, and PHP. They will choose the suitable programming language in accordance to the type of software that will be developed in order to develop an efficient error-free software. Each company may have distinct coding guidelines and programming tools used to generate the code in which developers will have to follow with regards to.

Phase 5: Testing the Software

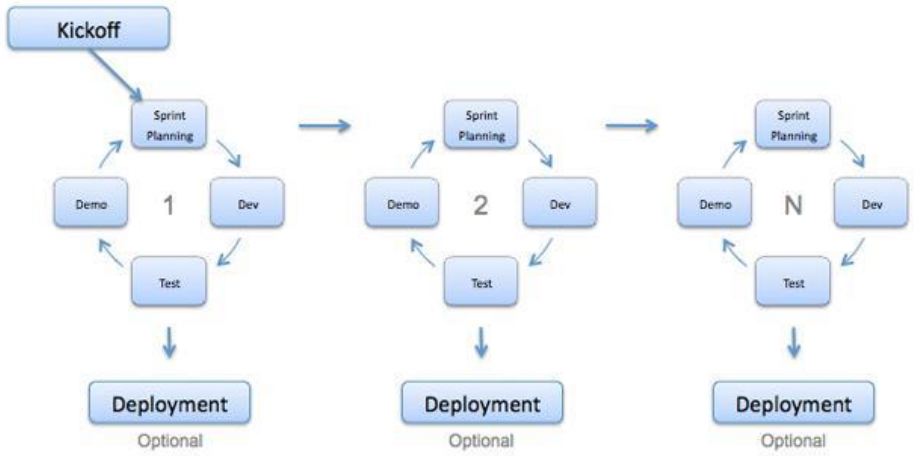
Despite testing being an important phase in almost all phases of the life cycle, this particular phase involves only testing of the software which product defects are reported, tracked, fixed and retested. The cycle of testing for software defects repeats until it reaches the highest quality standards stated in SRS.

Errors may ruin the software critically hence software testing is done at various levels of codes, for instance module testing, program testing, and product testing. Being aware of errors and methods of solution is the absolute importance to a reliable software. Also it prevents hassles in having to modifying the entire software by the ability to identify errors through testing the software throughout the developing stage.

Phase 6: Deployment in the Market and Maintenance

After the software is tested and ready to be deployed, it will be released in the fitting market. Occasionally, product will be deployed depending on the organization, they be release in a limited segment for a limited period of time for user acceptance testing. It is common as a business strategy to receive feedback from relative users in order to improve the software and raise user experience in general. With that, it concludes the software development life cycle.

Agile Software Development Model



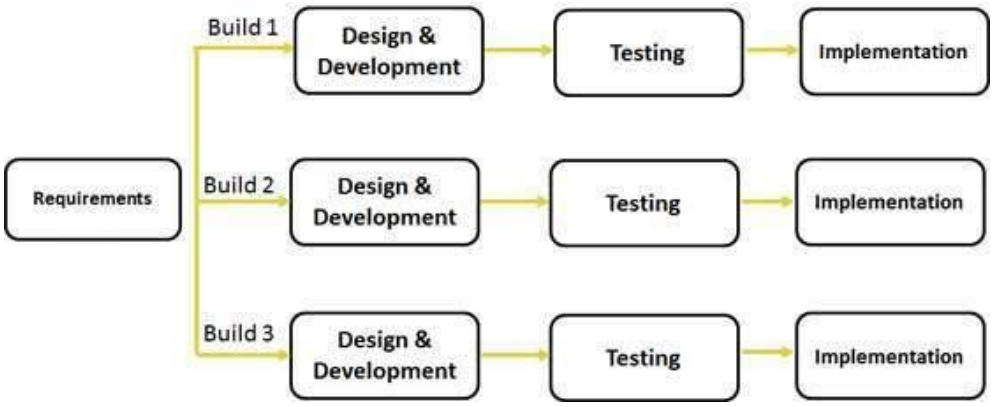
Agile model is a combination of the iterative and incremental process models. It focus on process adaptability and customer satisfaction by efficiently producing result of the working software product. The cycle of each phase is meticulously and thoroughly tested to ensure maintaining high software quality before deciding whether to deploy the software into the market or repeating the cycle.

The agile model is mostly used when new changes are needed to be implemented in the middle or later stages of development. New changes can be implemented at little cost due to the concept of the agile model. It lightens the stress of changes as each phase is not necessarily over-dependent on each other.

Both the developers and involved parties such as the market analysts or client have more sufficient time and options then having the software developed in a more rigid sequential way. The options provides them the ability to leave more significant and important decisions until more or better data are available. The project can continue to progress without the fear of reaching a sudden standstill. The sufficient time gives them allowance to analyse the current plan and discuss whether or not it is a good idea to proceed with deployment or should they repeat the cycle.

Pros	Cons
Customer satisfaction can be achieved by continuous and fast delivery of useful software.	Lack of emphasis on necessary designing and documentation.
Interactions are emphasized rather than process and tools. Customers, developers and testers will have to constantly interface with each other to ensure that there won't be communication breakdown. Communication is one important factor that contributes to building a satisfactory software.	Difficulty in estimating the effort required for the large software deliverables at the beginning of the cycle. Underestimating the manpower and effort needed to produce the results will cost wastage of time may result in unsatisfactory in customers.
Alteration and enhancements in requirement at the eleventh hour will not be affecting other phases as it will simply be added when the next cycle repeats.	In cases where customers are unsure of desired objective and requirements, the project can be off track easily.

Iterative Software Development Model



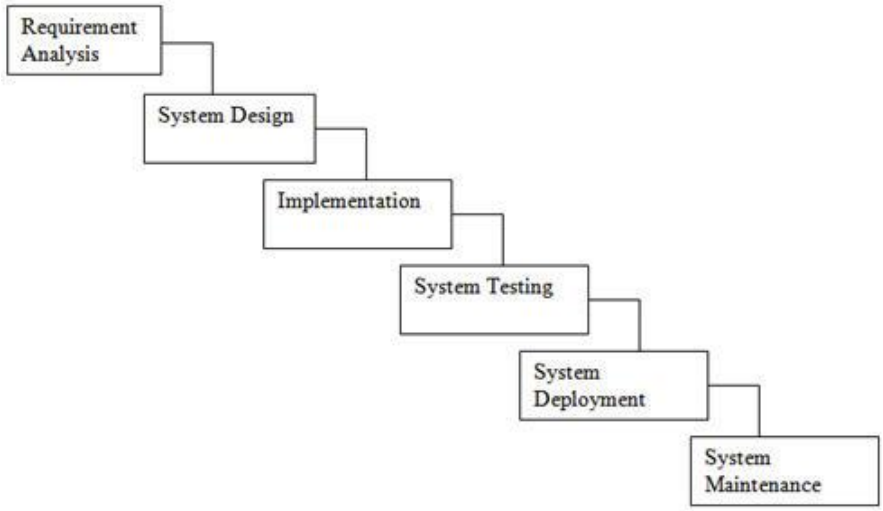
An iterative life cycle model does not attempt to start with a full specification of requirements. The development begins by specifying and implementing part of the software, which can then be reviewed in order to identify further requirements.

The process will be repeated, producing a new version of the software for each cycle of the model, as seen in the image above. There is built 1 where the design and development phase takes place, followed by testing of the software, and implementation. Then, a built 2 is produced which is the new version of the software.

The model is most commonly used when requirements of the complete system are clearly defined and understood. Major requirements must be defined in spite of the possibility for changes and enhancements over functionalities and requirements.

Pros	Cons
Ability to create high-level design of the application before the actual coding process begins. Developers can design and built a skeleton version of the software, and then enhance the design based of the skeleton version.	Each phase of an iteration is rigorous with no overlaps.
Product is built and improved stage by stage which allows early detection of defects at early stage. This avoids the downward complications of the defects.	High cost system architecture and design issues may arise because not all requirements are gathered up front for the entire lifecycle.
Less time spent on documenting and more time allocated to designing the software.	More management attention is required.
Risk analysis is better as at every stage, there is testing in every stage.	Management complexity is more and it is unsuitable for smaller projects.

Waterfall Software Development Model



The waterfall model is known as the linear-sequential life cycle model. It is simple to understand and use. Each phase must be completed before the next phase can begin and there is no overlapping in the phases.

Waterfall model was the first to be introduced in SDLC model and was widely used to ensure success of the project. In this model, the whole process of software development is divided into separate phases, the outcome of one phase acts as the input for next phase sequentially. All these phases are in a progress which is displayed and viewed as flowing steadily downwards seemingly like a waterfall which is why it is named as the waterfall model.

Phase 1: Requirement Analysis

All possible requirements of the requested software to be developed are recognized and analysed in this phase and documented in the requirement specification. It involves understanding the objective of the product.

Phase 2: System Design

The requirement specifications from first phase are re-analysed in this system in order to prepare the system design. System design helps in specifying hardware and system requirements and also helps in defining overall system architect. The team will analyse about the software and hardware needed to fulfil the project. From deciding the programming language to use for building the software to deciding the suitable database system are all decisions that have to be made in this phase.

Phase 3: Implementation

As mentioned, the output of the previous phase serves as an input for the next phase. In other words, with system design, the system is first developed in small programs called units. They are integrated in the next phase. Each unit will be developed and tested for its functionality which is known as Unit Testing.

Phase 4: Integration and Testing

The unit developed in the previous phase will be integrated into a system after unit testing. The entire system is then tested for any errors and failures after the integration progress.

Phase 5: Deployment of System

Once the functional and non-functional testing is completed, the software will then be released into the suitable market.

Phase 6: Maintenance

If there are any issues that occurs, this phase involves fixing that risen issues by releasing patches. It not only solve any issues but also improve and enhance the current version to ensure satisfactory of users.

Pros	Cons
Simple to implement as amount of resources needed if minimal.	In reality, projects rarely follows such a systematic sequential flow. Changes can cause confusion as the project proceeds.
Output of each phase is generated before doing the next phase, result in high visibility of the project.	As software and hardware decided for this model is fixed, any changes made will affect the next phase in sequence. Hence as technology improves at a fast speed, using same technology for a long time and not advisable. Hence with waterfall model, project cannot go on for a long period of time.
Easy management of project. Deadlines can be set for completion of each phase and evaluation can be done from time to time to check if project is going as planned.	The model is not recommended in situations when the client is IT-illiterate, as getting specifications from them will be a challenge. They might not know their requirements which is a challenge as each phases are dependent on each other. The team will need a detailed document on the specifications but client might not be able to provide them.