Journal of Management and Science

RESEARCH ARTICLE

Eleyon Publishers

DOI:10.26524.jms.11.3

11(1) (2021) 12-22

OPEN ACCESS

Overview of system development life cycle models

Kwadwo Kyeremeh[1]

Abstract

During the time half of the twentieth century, the utilization of Programmed computers has become huge. As an outcome, software programming has turned out to be increasingly differing and complex. Also, there are expanding requests on software programming – it must be less expensive, have more usefulness, be conveyed speedier, and be of higher quality than already. In the constantly changing environment and society of programming advancement, the procedures and strategies utilized when growing little projects are not adequate while developing extensive frameworks. As one response to this, distinctive improvement lifecycle models have been characterized. This paper portrays the three fundamental sorts of systems Development lifecycle models, from the successive models using incremental models to transformative models. The iterative advancement technique is additionally examined, and we additionally intricate the association of advancement lifecycle models to two rising fields in programming designing: programming design and part-based programming advancement.

Keywords: software,development, systems, lifecycle, programming, framework.

Author Affiliation: [1]Department of Accountancy, Sunyani Technical University, P.O Box 206, Sunyani-Bono Region, Ghana W/A.
Corresponding Author: Kwadwo Kyeremeh.Department of Accountancy, Sunyani Technical University, P.O Box 206, Sunyani-Bono Region, Ghana W/A.
Email: Kyeremehkwadwo@ymail.com
How to cite this article: Yashod.R, Overview Of System Development Life Cycle Models 11(1), 12-22. Retrieved from http://jms.eleyon.org/index.php/jms/article/view/455
Source of support: Nil
Conflict of interest: None.
Received: 4 Feburary 2021 **Revised:** 5 March 2021 **Accepted:** 7 March 2021

1.0 INTRODUCTION

During the time half of twentieth century, the utilizatin of Programmed computers has become huge. As an outcome, software programming has turned out to be increasingly differing and complex. Also, there are expanding requests on software programming – it must be less expensive, have more usefulness, be conveyed speedier, and be of higher quality than already. In the constantly changing environment and society of programming advancement, the procedures and strategies utilized when growing little projects are not adequate while developing extensive frameworks. As one response to this, distinctive improvement lifecycle models have been characterized. This paper portrays the three fundamental sorts of systems Development lifecycle models, from the successive models using incremental models to transformative models. The iterative advancement technique is additionally examined, and we additionally intricate the association of advancement lifecycle models to two rising fields in programming designing: programming design and part-based programming advancement.[1]

Tri.Soft is a manufacturing and installation of systems Software Development Company which normally operates in Ghana and some part of West Africa. The company has existed for the past 10 years. We offer the sale of software development equipment and also provide an upgrade of an existing systems software and also recommend the best systems development models to use based on the kind of services our clients provide. Our company has been contracted by FILER, a transport service company that transports goods from one geographical location to another. The company has an information system that handles its day-to-day activities. The company intends to do additional activities which will demand the upgrade of the existing systems. TRI.Soft. my company has been contracted to build upgrade systems that will handle the operations at FILER.

The systems to be built it will be a integrated with the other systems using APIs to access services from systems such as Google map.[2]

2.0 LITERATURE REVIEW

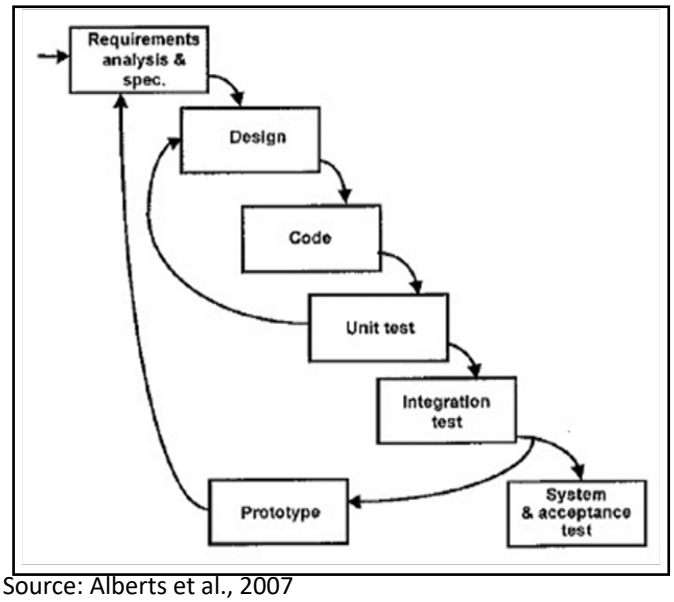
The literature review is the body of whole work and it talks about the models that are needed in the systems development life cycle. There are about eight models of systems development life cycle which includes Incremental model, Prototype model, Spiral model, Iterative model, Waterfall model, V- model, RAD model, and an Agile model but for this research, we are going to limit ourselves with only four of the models which will recommend for FILER to choose out of this four for the solution of their problems. The four selected models are the Waterfall model, V-model, Spiral model, and Iterative mode.[3]

2.1 Rapid Application Development (RAD)

As indicated by[4], The Rapid Application Development is a collection of methodologies that rose because of the inadequacies of waterfall development and its assortments. RAD unites excellent systems and PC gadgets to quicken the examination, diagram, and execution stages with a particular

ultimate objective to get some package of the structure framed quickly and under the control of the customers for assessment and information. CASE (PC upheld programming building) devices, JAD (joint application advancement) sessions, fourthperiod/visual software design jargon (e.g., Visual Basic. NET), and code generators may all accept a section in Rapid Application Development. Rapid Application Development can upgrade the speed and nature of system development, it might moreover introduce an issue in managing customer wants. As systems are made more quickly and customers get a prevalent appreciation of information advancement, customer yearnings may accumulation and system requirements may stretch out amid the wander (once in a while known as degree creep or highlight slither). Rapid Application Development may be driven in a gathering of ways. Iterative development breaks the general wander into a movement of adjustments that are made progressively. The most basic additionally, fundamental requirements are bundled into the important adjustment of the system. This shape is created quickly by a downsized waterfall get ready, and once executed, the customers can give imperative feedback to be united into the accompanying type of structure(5).

2.2 General Overview of "RAD MODEL"



Source: Alberts et al., 2007

In Rapid Application Development, the system model is a «serious» modification of the system and gives immaterial highlights. Taking after reaction and comments from the customers, the architects reanalyze, update, and re-realize a second model that changes needs and incorporates more parts. This cycle continues until the specialists, customers, and backing agree that the model gives enough convenience to be presented and used as a part of the affiliation. System prototyping quickly gives a structure to customers to survey furthermore, comforts customers that progress is being made. The approach is outstandingly useful right when customers encounter issues imparting requirements for the system. An impediment, in any case, is the default of mindful, exact analysis going before settling on plan and execution decisions.

System models may have some focal blueprint imperatives that are a prompt delayed consequence of a lacking cognizance of the structure›s certifiable requirements right on time in the wander.[6]

Taking everything into account, we have assessed and examine particular Rapid Application Development systems, methodologies, and related work. Using Rapid Application Development reasoning to make software design requires the slightest getting ready for snappy prototyping. It uses lesser time and makes quality program design however that quality writing computer software is sensible for little to medium size endeavors/ structures. Right when Rapid Application Development strategies are used for confounding or enormous errands, they can›t give required results, and the nature of the structure is dealt with. Thus, one can›t simply dial on the security. Rapid Application Development supports quick program design progression and makes without question transport on adjusted time.

As Rapid Application Development we do not have strict methods so a couple of sensible recommendations were found amid the written work review.[7]These recommendations are still unforeseen like conflicts to some extent and commitments among the associates, engineers have the vitality to take decision so their decision can be advancement driven, and builds facilitate oversee customers so deficient correspondence issue happens because specialists often use capable terms which can›t be adequately understood by a layman. Engineers expect the key part in completing Rapid Application Development strategies and their decision can›t be changed by different partners. Snappy transport is the essential point in Rapid Application Development so there may be an oversight or misstep during the change. In Rapid Application Development execution stage is fundamental and designers› change is consistent as demonstrated by customer›s essential. They assume that essentials are incredibly difficult to clear up freely so any system should be created with the planned exertion of clients. Customers appreciate the item progression gets ready. After examination of different methodologies and investigations related to Rapid Application Development, we gather that Rapid Application Development is uncommonly proper for only little to medium size errands. We can›t sign each one of the structures by using Rapid Application Development systems since they overwhelmingly depend on reusing the present parts; so it ends up being straightforward for software engineers to break the system and take the game plans. Up till now, every one of the methods of RAD like agile, scrum thus on are not free of the above-portrayed proposals.[6]

2.2.1 Incremental model

In the incremental model, the whole requirement is divided into various builds. Multiple development cycles take place here, making the life cycle a "multi-waterfall" cycle. Cycles are divided up into smaller, more easily managed modules. Each module passes through the requirements, design, implementation, and testing phases. A working version of the software is produced during the first module, so you have working software early on during the software development life cycle. Each subsequent release of the module adds function to the previous release. The process continues till the complete system is achieved. In the diagram below when we work incrementally, we are adding piece by piece but expect that each piece is fully finished. Thus, keep on adding the pieces until it's complete. As in the duplicate above a person has thought of the application. Then he started building it and in the first iteration, the first module of the application

or product is ready and can be demoed to the customers. Likewise in the second iteration, the other module is ready and integrated with the first module. Similarly, in the third iteration, the whole product is ready and integrated. Hence, the product got ready to step by step.[7]

2.2.2 Advantages of Incremental model

The incremental model has advantages which are as follows: Generates working software quickly and early during the software life cycle, this model is more flexible – less costly to change scope and requirements, It is easier to test and debug during a smaller iteration, In this model, customer can respond to each built, Lowers initial delivery cost, Easier to manage risk because risky pieces are identified and handled during it'd iteration.[8]

2.2.3 Disadvantages of the Incremental model

The incremental model has disadvantages which are as follows: Needs good planning and design, Needs a clear and complete definition of the whole system before it can be broken down and built incrementally, Total cost is higher thanwaterfall.

2.2.4 When to use the Incremental model:

Certain circumstances call for the use of Incremental model and they are as follows: This model can be used when the requirements of the complete system are clearly defined and understood, Major requirements must be defined; however, some details can evolve with time, there is a need to get a product to the market early, A new technology is being used, Resources with needed skill set are not available, there are some high-risk features and goals.[7]

2.3 Prototyping

The basic idea here is that a instead of the freezing the requirements before a design or coding can proceed, a throwaway prototype is built to understand the requirements. This prototype is developed based on the currently known requirements. By using this prototype, the client can get an "actual feel" of the system, since the interactions with the prototype can enable the client to better understand the

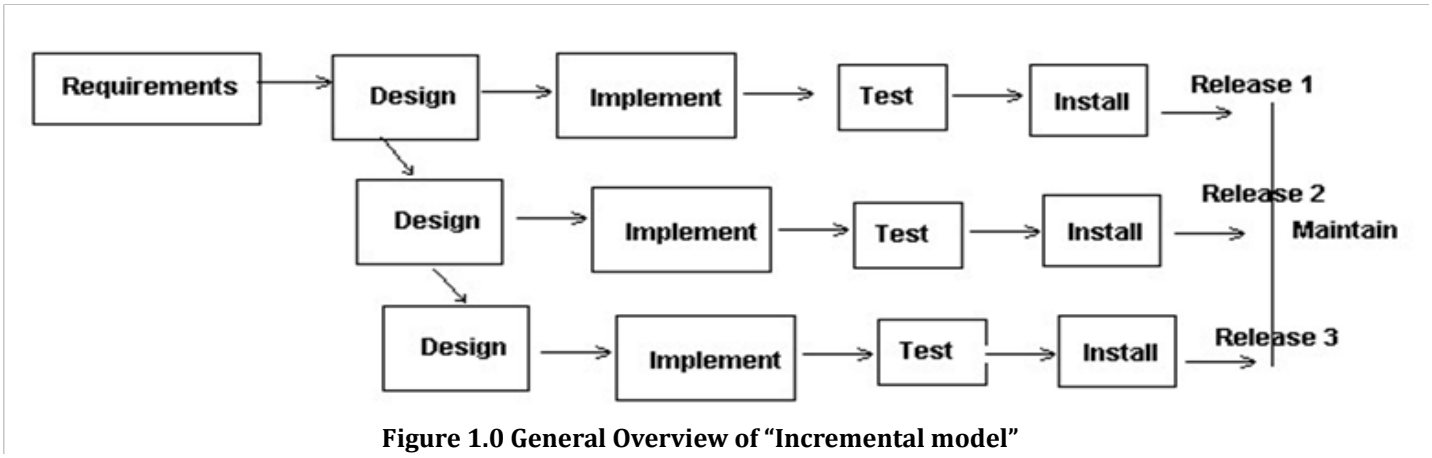


Figure 1.0 General Overview of "Incremental model"

Source: Lewallen, 2005

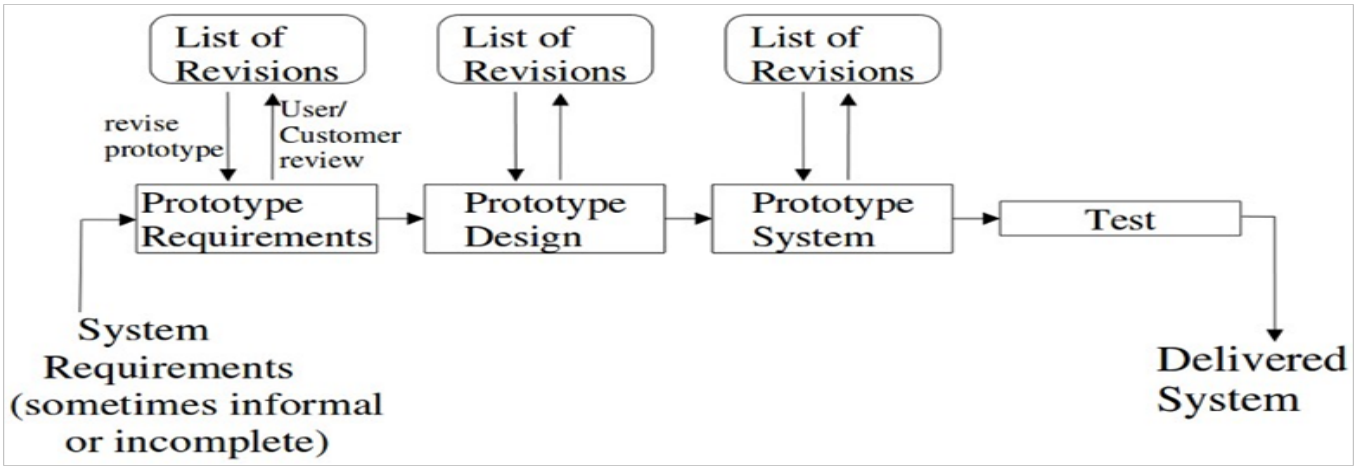


Figure 2:General Overview of "Prototype model"

Source: Lewallen, 2005

requirements of the desired system. Prototyping is an attractive idea for complicated and large systems for which there is no manual process or existing system to help to determine the requirements.The prototype is usually not complete systems and many of the details are not built in the prototype. The goal is to provide a system with overall functionality.[6]

2.3.1 Advantages of Prototype model

Prototype Models have several advantages which are as follows: Users are actively involved in the development, Since in this methodology a working model of the system is provided, the users get a better understanding of the system being developed, Errors can be detected much earlier, Quicker user feedback is available leading to better solutions, Missing functionality can be identified easily, Confusing or difficult functions can be identified Requirements validation, Quick implementation of, incomplete, but functional, application.[6]

2.3.2 Disadvantages of Prototype model

Prototype Models have several advantages which are as follows: Leads to implementing and then repairing way of building systems, Practically, this methodology may increase the complexity of the system as the scope of the system may expand beyond original plans, Incomplete application may cause application not to be used as the full system was designed Incomplete or inadequate problem analysis.

2.3.3 When to use Prototype model

Certain circumstances call for the use of a Prototype model and they are as follows: Prototype model should be used when the desired system needs to have a lot of interaction with the end-users, Typically, online systems, web interfaces have a very high amount of interaction with end-users, are best suited for the Prototype model. It might take a while for a system to be built that allows ease of use and needs minimal training for the end-user, Prototyping ensures that the end-users constantly work with the system and provide feedback which is incorporated in the prototype to result in a useable system. They are excellent for designing good human-computer interface systems.

2.4 Agile Development

Agile development is a gathering of programming-driven techniques that attention on streamlining the Systems Development Life Cycle (SDLC). A great part of the demonstrating and documentation overhead is dispensed with; rather, up close and personal correspondence is favored. A extend accentuates straightforward, iterative application advancement in which each emphasis is a finished programming venture, including arranging, prerequisites examination, outline, coding, testing, and documentation.[8]

As you can see in the diagram above, in a waterfall process the “design” and “implementation” stages exist before, and distinct from, the “verification” and “maintenance” stages. This split between software developers and software testers, positioning them as separate entities at different points along a production cycle, is one of the fundamental problems that Agile seeks to resolve. Agile Development is a social event of programming-driven procedures that consideration streamlining the SDLC.

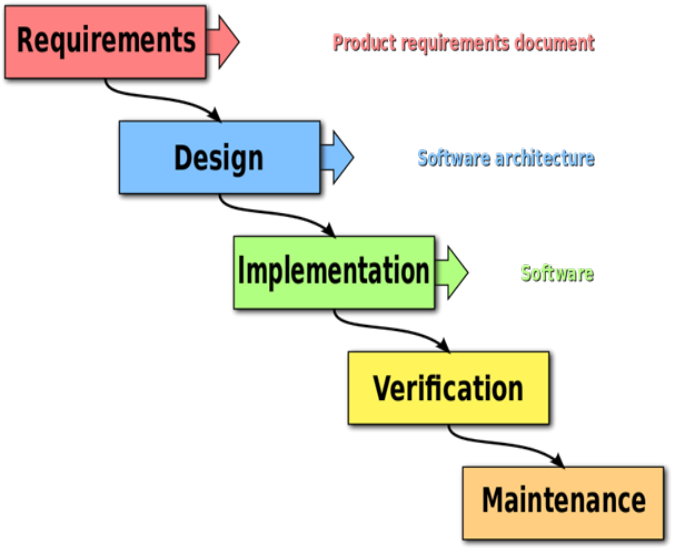


Figure 3: General Overview of “Agile model”

Source: Larman, 2003

An extraordinary part of the exhibiting and documentation overhead is shed; rather, very close correspondence is favored. An augment complements clear, iterative application progression in which every accentuation is a completed process of programming endeavors, including orchestrating, requirements examination, layout, coding, testing, and documentation.[7]

Agile Methods push efficiency and values of the over substantial-weight prepare overhead and ancient rarities. The Coordinated Pronouncement a succinct outline of Light-footed qualities was composed and marked in 2001 albeit Nimble techniques have existed since the mid-90s. Nimble techniques advance an iterative component for delivering programming, also, they promote increment the iterative way of the programming lifecycle by fixing plan - code- test circle too in any event once every day (if very little more habitually) instead of once per emphasis. Lithe visionary Kent Beck challenged the conventional cost of progress bend to prove by Boehm et al., 1987, more than a quarter-century. To finish this «compliment « cost of progress bend, Agile methods advance several designing practices that empower practical change. As opposed to centering a great deal of exertion on huge in advance plan investigation, little additions of practical code are delivered by business requirements. Nimble tasks dodge « upfront» necessities gathering for the reasons expressed previously: clients can›t adequately deliver all necessities in sufficiently high detail for usage to happen toward the start of a venture. Customers might not have any desire to settle on choices about the framework until they have more information. Coordinated qualities a high deceivability and client association. The visit showing and the arrival of programming regular in Spry approaches allows clients to «attempt programming» occasionally, what›s more, give criticism. Agile makes difference organizations deliver the «right item». An iterative approach permits clients to defer choices too. Choices can be postponed to some future emphasis when better data or innovation is accessible to advance the decision. For instance, we as of late deferred selecting a database bundle for an application because a portion of the craved highlights was not accessible at that time in the alternatives we needed to browse. We in this manner assembled the framework in a database autonomous way, also, (fortunately) a couple of weeks before the product dispatch another adaptation was discharged

by one of the database sellers that tackled our issue. One of the biggest favorable circumstances to IID is that work can start before the greater part of the necessities is known. Numerous associations are not completely staffed with business analysts turning out reams of necessities specs. A remarkable opposite, in our encounter frequently the bottleneck in the advancement handle has been the absence of accessibility of client space specialists for definite prerequisites investigation. This is particularly the case with little organizations where area specialists wear numerous caps and regularly can›t focus on a few months of straight necessities investigation. IID is preferably suited then to go up against chomp measured lumps of prerequisites that the client can without much of a stretch process. How do Agile tasks organize function? A think about by the Standish Bunch demonstrates that in the run-of-the-mill programming frameworks 45 percent of the components are never really utilized by clients and another 19% are just uncommon utilized. This is generally because the unused components were determined in some up-front arrangement before the proportion of their cost to esteem was viewed as or even comprehensible. Concentrating on high business esteem highlights first is accordingly a basic part of proficient Agile Development. Since we can alter course rapidly (every emphasis) also, the cost of progress is low, there is a significant opportunity for the client to re-look at business variables at the start of every cycle to select components for incorporation in the present emphasis as per business return on initial capital investment.[9]

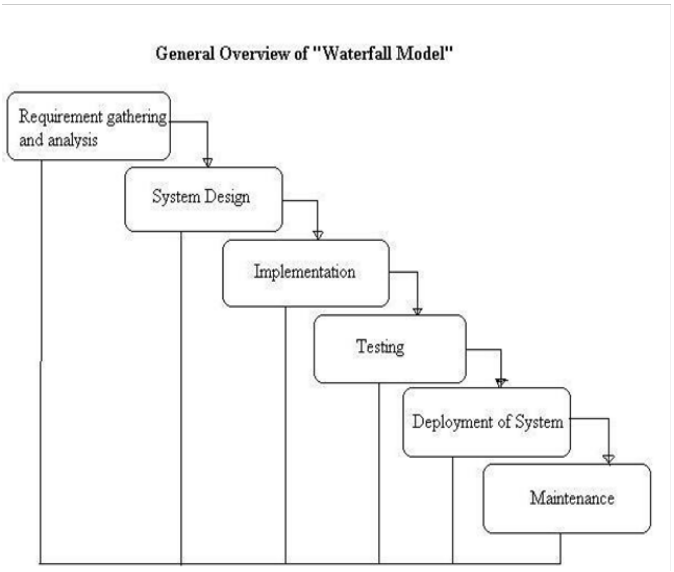
The advancement group must a convey specialized dangers to the client, yet at last, the client chooses what the improvement group fabricates.

A standout amongst the most normally made inquiries by those looking at Deft is, « how would you know when the product will be done if there›s no up-front arrange?» and the required take after up question, «in what manner would we be able to move for such a venture?» It sounds somewhat terrifying: how about we begin working in short iterative cycles that yield self-evident programming without really arranging everything in progress. In any case, we realize that we cannot get ready for everything ahead of time.

The Agile answer is to inspect extend advance exactly, as opposed to attempting to figure how things may get down to business from the earlier Spry procedures like Scrum and XP utilize an idea called speed which is the measure of evaluated exertion a group can finish in a time-boxed emphasis. Once a group has built up a speed, a Venture Burn down Outline can be used to assess the inevitable finish of an expected excess of work. In conclusion, I will say, the thought that Agile is a radical deviation from the since quite a while ago settled, reliable history of waterfall programming improvement is off base. Even thoughthe waterfall is regularly alluded to as «customary», programming building has had a short history in respect to other designing disciplines. Unlike connect building, programming improvement is not based on a great many years of experimentation and is along these lines in a quickly developing early stages as a designing order. Light-footed is just the most recent hypothesis that is generally supplanting the waterfall approach that itself will change furthermore, develop well into what›s to come.[9]

2.4 Waterfall Model

We have offered a several of our clients which had similar challenges to that of FILER with the approach of waterfall model to solve the systems development challenges however Waterfall Model was the first Process Model to be presented. It is additionally alluded to as a direct successive life cycle model. It is exceptionally easy to comprehend and utilize. In a waterfall model, every stage must be finished before the following stage can start. This sort of model is essentially utilized for the task which is little and there are no unverifiable necessities. Toward the end of every stage, a survey happens to figure out whether the task is on the right way and regardless of whether to proceed or dispose of the venture. In this model, the testing begins simply after the improvement is finished. In the waterfall model stages don›t cover.[7]



Source. Pilgrim, 2012.

2.4.1 Advantages and Disadvantages of waterfall model

The waterfall model has several advantages and also disadvantages as according to (10), that is going to be of good help when FILER accept that we apply the waterfall system development model in solving their challenges.

2.4.2 Advantages of Waterfall Model

This model is straightforward and straightforward and use. It is anything but difficult to oversee because of the unbending nature of the model – every stage has particular deliverables and an audit procedure and these model stages are prepared and finished each one in turn. Stages don›t cover. The waterfall model functions admirably for littler undertakings where necessities are exceptionally surely known.

2.4.3 Disadvantages of waterfall model

Once an application is in the testing stage, it is a exceptionally hard to do a reversal and change something that was not well-thoroughly considered in the idea stage. No working programming is created until late amid the life cycle. It has high measures of danger and instability which is not a decent model for complex and item situated tasks

2.4.3 Disadvantages of waterfall model

Once an application is in the testing stage, it is a exceptionally hard to do a reversal and change something that was not well-thoroughly considered in the idea stage. No working programming is created until late amid the life cycle. It has high measures of danger and instability which is not a decent model for complex and item situated tasks it is s poor model for long and continuous undertakings which is appropriate for the accomplishments where the fundamentals are at a moderate to high danger of evolving.

2.4.4 At the point when to utilize the waterfall model

In [7] it can be noted that there are circumstances at which the waterfall model must be utilized and below are some of the circumstances.This model is utilized just when the necessities are extremely outstanding, clear, and altered, Product definition is steady, Technology is caught on, there are no vague necessities, Ample assets with required mastery are accessible openly, the task is short.Less client enter activity is included amid the improvement of the item. Once the item is prepared then no one but it can be demoed to the end-user clients. Once the items are produced and if any disappointment happens then the expense of altering such issues is high since we have to upgrade wherever from the archive till the rationale of change is achieved.

2.5 V-Shaped model

The V-Shaped model means the Confirmation and Acceptance model. Much the same as the waterfall show, the Angular life cycle is a consecutive way of execution of procedures in a software upgrade. Every stage must be finished before the following stage starts. Testing of the item is arranged in parallel with a comparing period of advancement in the V-Shaped model.[7]

2.5.1 The different periods and advantages and disadvantages of the V-Shaped model

There are some periods in the V- model which differ but provide same results and also have some point of interest as stated by Alshamrani&Bahattab,

2.5.2 Different periods of V-Model

- Prerequisites like BRS and SRS start the life cycle show simply like the waterfall model. In any case, in this model, before advancement is begun, a framework test arrangement is made. The test arrangement concentrates on meeting the usefulness indicated in the necessities gathering.
- The abnormal state outline (HLD) stage concentrates on framework engineering and plan. It gives a review of arrangement, stage, framework, item, and administration/ process. An integration test arrangement is made in this stage also with a specific end goal to test the bits of the product framework›s capacity to cooperate.
- The low-level outline (LLD) stage is the place the genuine programming parts are planned. It characterizes the genuine rationale for every single part of the framework. Class chart with every one of the strategies and connection between classes goes under LLD. Part tests are made in this stage too.
- The usage stage is, once more, where all coding happens. When coding is finished, the way of execution proceeds up the right half of the V where the test arranges grew before are presently put to utilize.
- **Coding:** This is at the base of the Angular Shape model. Module configuration is changed over into code by engineers.

2.5.3 Advantages of V-model

There are some advantages of the V-model from which include, Simple and simple to utilize, Testing exercises like arranging, test outlining happens well before coding. This spares a great deal of time, thus a higher shot of progress over the waterfall model, Proactive imperfection following – that is deformities are found at an early stage, Avoids the descending stream of the imperfections. Works well for little undertakings where prerequisites are effortlessly caught on.

2.5.4 Disadvantages of V-model:

There are some disadvantages of the V-model which include, very unbending and minimum adaptable, Software is created a amid the usage stage, so no early models of the product are delivered, If any progressions happen halfway, then the test archives alongside necessity records must be upgraded.

2.5.5 At the point when to utilize the V-model

In (7) it can be noted that there are periods that are best for V-model to be utilized and this are some of the circumstances, The Angular model ought to be utilized for little to medium estimated ventures where necessities are plainly characterized and settled, The Angular model ought to be picked when sufficient specialized assets are accessible with required specialized mastery. High certainty of client is required for picking the Angular model methodology. Since, no models are created, there is a high hazard required in meeting client desires.

2.6 ITERATIVE MODEL

An iterative life cycle model is also part of a systems development life cycle model which does not endeavor to begin with a full particular of necessities. Rather, advancement starts by determining and executing simply part of the product, which can then be checked on so as to recognize further prerequisites. This procedure is then rehashed, delivering another variant of the product for every cycle of the model.[11]

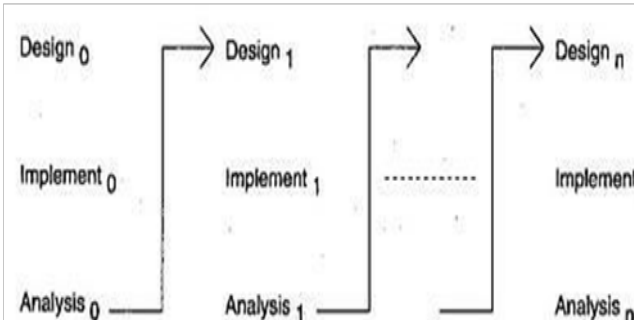


Figure 5: General Overview of "Iterative Model"
Source: Batory, Singhal, Dasari, Geraci, and Sirkin, 1994

2.6.1 The different periods and advantages and disadvantages of an Iterative model

As stated above using the diagram, it can be stated according to [8], there exist some advantages and disadvantages of Iterative models, and below will illustrate some of the advantages and disadvantages of the iterative model.

2.6.2 Advantages of Iterative model

As advantages of an iterative model, we can just make an abnormal state configuration of the application before we start to assemble the item and characterize the outline answer for the whole item. Later on, we can outline and construct a skeleton form of that, and afterward developed the configuration taking into account what had been manufactured. In the iterative model, we are building and enhancing the item regulated. Henceforth we can track the imperfections at early stages. This dodges the descending stream of the deformities.

In the iterative model, we can get dependable client input.At the point when displaying portrayals and outlines of the item to clients for their criticism, we are viably requesting that they envision how the item will function. Lastly in an iterative model less time is invested on reporting and more energy is given for planning.

2.6.3 Disadvantages of Iterative model

There are some disadvantages of the Iterative model which include are that, each period of a cycle is unbending without any covers on a costly framework engineering or plan issues may emerge because not all prerequisites are gotten together front for the whole lifecycle.

2.6.4 At the point when to utilize the iterative model

It can be noted that there are periods that are best for the Iterative model to be utilized and these are some of the circumstances,Requirements of the complete framework are characterized and caught on, When the task is enormous, Major necessities must be characterized; be that as it may, a few subtle elements can develop with time.[8]

2.7 Spiral model

The spiral model is similar to the incremental model, with more emphasis placed on risk analysis in the systems analysis procedure. The spiral model is like the incremental model, with more attention put on risk analysis. The spiral model has four stages: Engineering, Evaluation, Risk Analysis, and Planning. A system spread out more than once goes through these stages in cycles (called Spirals in this model). In the standard spiral, beginning in the arranging stage, requirements are accumulated and risk is measured. Each ensuing spiral expands on the pattern spiral.[9]

2.7.1 The spiral model phases.

InBoehm et al., 1987it is well from noted that the Spiral model has four phases according to the Diagram above and the phases are as follows:Planning, Risk Analysis, Engineering, Evaluation. A software project repeatedly passes through from these phases in iterations (called Spirals in this model). In the baseline spiral, starting in the planning phase, requirements are gathered and risk is assessed.

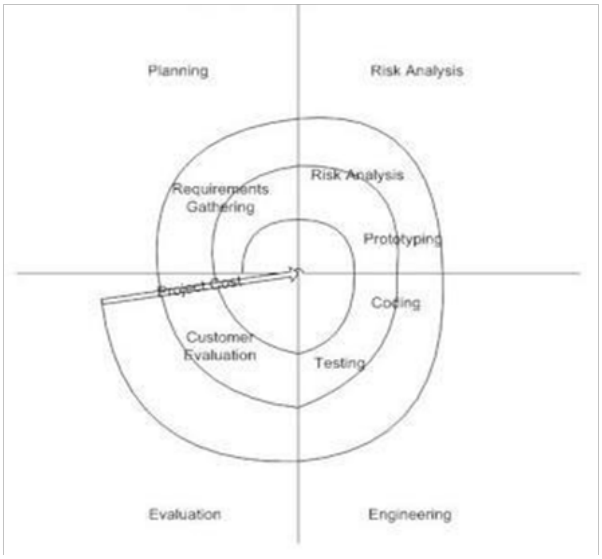


Figure 6: General Overview of "Spiral Model"

Source: Boehm, 2000

Each subsequent spiral builds on the baseline spiral.

- **Planning Phase**: Requirements are gathered during the planning phase. Requirements like 'BRS' that is 'Business Requirement Specifications' and 'SRS' that is 'System Requirement specifications.
- **Risk Analysis:** In the risk analysis phase, a process is undertaken to identify risk and alternate solutions. A prototype is produced at the end of the risk analysis phase. If any risk is found during the risk analysis, then alternate solutions are suggested and implemented.
- **Engineering Phase:** In this phase,the software is developed, along with testing at the end of the phase. Hence in this phase, the development and testing are done.
- **Evaluation phase**: This phase allows the customer to evaluate the output of the project to date before the project continues to the next spiral.

2.7.2 Advantages and Disadvantages of a Spiral model and when to use this model

As stated above from using the diagram, it can be stated according to [9], there exist some advantages and disadvantages of Spiral models, and below will illustrate some of the advantages and disadvantages of the Spiral Model.

2.7.2.1 Advantages of Spiral model

There are some advantages of a Spiral model which are, High amount of risk analysis hence, avoidance of Risk is enhanced, good for large and mission-critical projects, Strong approval and documentation control, Additional Functionality can be added at a later date, Software is produced early in the software life cycle.

2.7.2.2 Disadvantages of the Spiral model

There are some disadvantages of a Spiral model which are as follows, can be a costly model to use, Risk analysis requires highly specific expertise, Project's success is highly dependent on the risk analysis phase, does not work well for smaller projects.

2.7.3 When to use the Spiral model

It can be noted that there are periods that are best for the Spiral model to be utilized and these are some of the circumstances. When costs and risk evaluation is important, for medium to high-risk projects, Long-term project commitment unwise because of potential changes to economic priorities, Users are unsure of their needs, Requirements are complex, new product line, Significant changes are expected.[12]

2.7.4 Comparison of Different SDLC Models

As there are different models of programming advancement life cycle, each has its points of interest and drawbacks relying on which we need to choose, which model we ought to pick.For example, if the prerequisites are known beforehand and surely knew and we need full control over the venture at record-breaking, then we can utilize the waterfall model.The winding model is useful for vast and mission basic ventures where a high measure of danger investigation is required like dispatching of the satellite. Iterative life cycle models don›t endeavor to begin with full detail of prerequisites. Rather, improvement starts by indicating and executing simply part of the product which can then be inspected with a specific end goal to recognize further prerequisites. This procedure is then rehashed, creating another form of the product for every cycle of the model.[11] Angular Model has a higher shot of progress over the waterfall model because of the improvement of test arrangements amid the life cycle. It functions admirably for little tasks where necessities are effectively caught on.

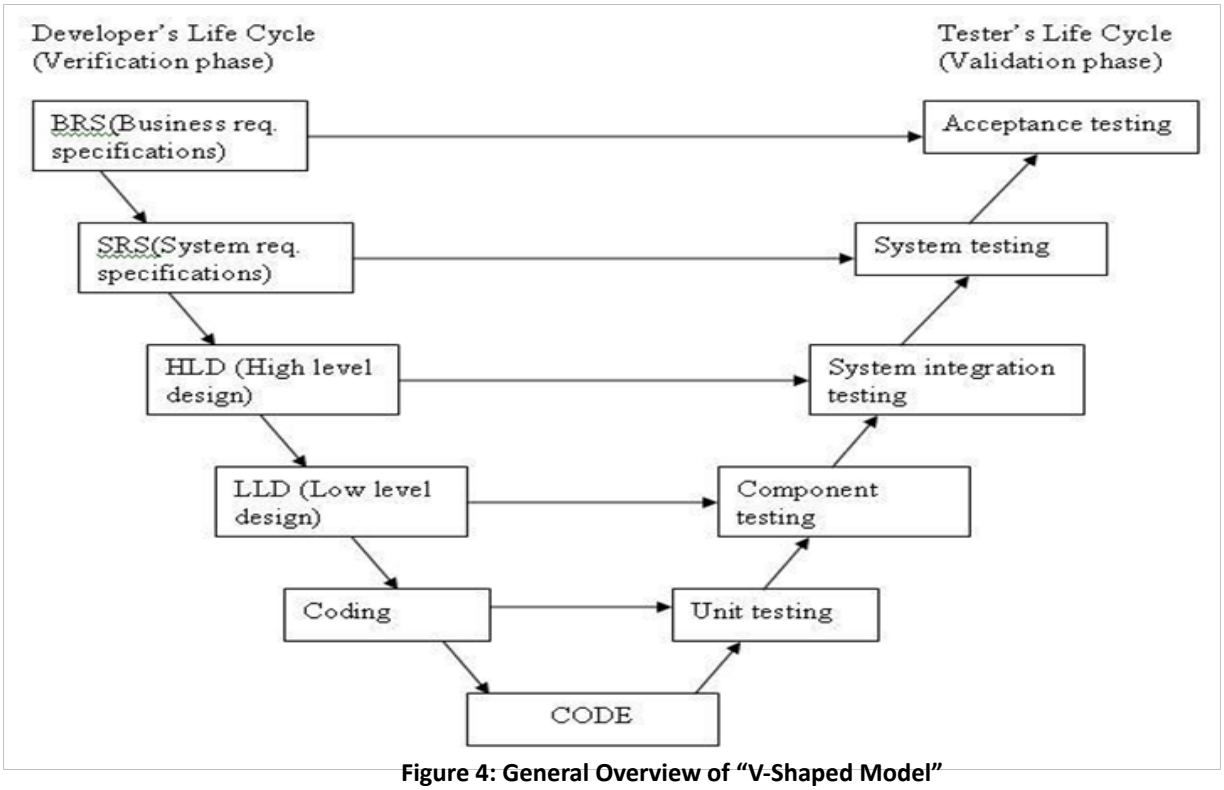


Figure 4: General Overview of "V-Shaped Model"

Source: Ali Munassar and Govardhan, 2010

Table 1: Comparison by Disadvantages

S.No	Waterfall Model	Iteration Model	Spiral Model	V Shape Model
1	Adjusting scope during the life cycle can kill a project	Milestones are more ambiguous than the waterfall	Can be a costly method to use	Too rigid like the waterfall model
2	It does not produce any working software until late during the life cycle	Activities performed in parallel are subject to miscommunication and mistake assumptions	Risk analysis required highly specific expertise	Little flexibility and adjusting scope is difficult and expensive
3	High amount o f risk and uncertainty	Unforeseen interdependencies can create problems	The project's success is highly dependent on the risk analysis phase	Software is developed during the implementation phase, so no early prototypes of the software are produced
4	Poor model for complex and object - oriented projects. A poor model where requirements are at a moderate to high risk of changing	Changes are possible as it is an iterative model	Does not work well for smaller projects	The model does not provide a clear path for problems found during testing faces.

Source: Rastogi, 2015

Table 2: Comparison by Advantages

S.No	Waterfall Model	Iteration Model	Spiral Model	V Shape Model
1	Simple and easy to use	More flexible thanthe basic waterfall model	The high amount of risk analysis	Simple and easy to use
2	Phases are processed and completed one at a time	Implementation of easy areas does not need to wait for the hard ones.	Software is produced early in the software life cycle	Higher chances of success over the waterfall model due to the development of test plans early on during the life cycle
3	Easy to manage due to the rigidity of the model because each phase has specific deliverables and a review process.	If there is personnel continuity between the phases, the documentation can be substantially reduced.	Good for large and mission-critical projects	Each phase has specific deliverables
4	Works well for smaller projects where requirements are very well understood	Works well for smaller and moderate size projects	Works well for projects where risk analysis contains higher priority	Works well for small projects where requirements are easily understood.

Source: Rastogi, 2015

2.0 Conclusion

There are numerous Systems Development Life Cycle (SDLC) models according toRastogi, 2015,, for example, Waterfall, RAD, Prototype, Iterative, Incremental, V-molded, Agile, Spiral and so on utilized as a part of different associations relying on the conditions winning there. All these diverse Systems Development models have their particular points of interest and hindrances. In the Software Industry, half and half of every one of these systems are utilized i.e., with some alteration. In this paper, we have thought about the distinctive programming improvement life cycle models on the premise of specific elements like-Requirement particulars, Risk association, User contribution, Cost, and so on the premise of these elements for a specific programming venture one can choose which of these product advancement life cycle models ought to be decided for that specific undertaking. Selecting the right life cycle model is critical in a product industry as the product must be conveyed inside the time due date and ought to likewise have the sought quality. This study will make the way toward selecting the SDLC model easily henceforth will turn out to be extremely compelling for FILER to choose the best out of the four selected system models namely, Waterfall, Spiral, V-Shaped and Iterative Models for possible integration into their existing systems for a possible upgrade. Based on this study we can also conclude that finding and fixing the software defects of FILER in respective phases is less expensive than finding and fixing them in the testing phase. Our survey results have shown that in our local software industry Prototyping, Joint Requirement Development, and training are the techniques mostly used for defect prevention. Other techniques which are being used are Case Tools and QFD. If we relate the defect prevention techniques and their impact on the requirements attributes then according to our survey, Correctness, Clarity, Completeness, and unambiguous are the attributes which are being improved because of defect prevention techniques. Organizations are unable to provide the percentage gain. The major reason is the lack of data to calculate this percentage gain to provide an upgrade and possible integration.[13]

4.0 REFERENCE

1. S. Khan, A.B. Dulloo, M. Verma. Systematic Review of Requirement Elicitation Techniques. 4(2) (2014) 133–138.
2. R.R. Lecture, Methodologies A. Software Development Methodologies, (2015).
3. I.H. Sarker, F. Faruque, U. Hossen, A. Rahman, A survey of software development process models in software engineering. International Journal of Software Engineering and its Applications. 9(11) (2015) 55-70.
4. W. Scacchi, Process Models in Software Engineering, 24(1) (2001) 1–24.
5. P.A. Laplante, Requirements engineering for software and systems. Auerbach Publications, (2017).
6. E. Camel, S. Becker, A Process Model for Packaged Software Development. IEEE TRANSACTIONS ON ENGINEERING MANAGEMENT. 42 (1995) 50-61.
7. A. Alshamrani, A. Bahattab, A Comparison Between Three SDLC Models Waterfall Model, Spiral Model, and Incremental/Iterative Model [Internet]. Available from: www.IJCSI.org, 12(1) (2015) 106-111.
8. Y. Ghanam, F. Maurer, An Iterative Model for Agile Product Line Engineering, (2008).
9. B.W. Boehm, Defense TRW, Group S. A Spiral Model of Software Development and Enhancement. 1 (1987) 61-72.
10. D.M. Rodvold, A Software Development Process Model for Artificial Neural Networks in Critical Applications, (1999).
11. V. Rastogi, Software Development Life Cycle Models-Comparison , Consequences. 6(1) (2015) 72–168.
12. B.J.J. Voigt, Dynamic System Development Method. (2004).