International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

The Waterfall Model-Software Engineering

[1]Dr. Rais Abdul Hamid Khan, [2]Patil Raj Gopal, [3]Patil Bhavesh Yogesh, [4]Patil Vaishnavi Nitin, [5]Khairnar Shekhar Sahebrao, [6]Parkhe Nilesh Sharad, [7]Patil Vivek Santosh

[1]Professor, SOCSE, Sandip University, Nashik
[2,3,4,5,6] BTech Scholar's, School of Computer Science and Engineering Sandip University, Nashik, Maharastra, INDIA
[1]rais.khan/@-sandipuniversity.edu.in, [2]rp789407@gmail.com

ABSTRACT

Software development projects have long been plan driven processes, but the coming of age of Agile Methodologies has given rise to a more adaptive approach to software/system development. the main objective of this research is to represent some modifications to the model, which correct most of the problems, connected with the waterfall model. Commonly accepted problems are for example to survive with change and that defects all too often are detected too late in the software development process.

Keywords- software development process , software development model , software development , SDLC (Software Development Life Cycle).

I. Introduction

This paper talks about waterfall method of software development. a software development methodology is known as SDLC short for Software Development Life Cycle and is majorly used in several engineering and industrial fields such as systems engineering, software engineering, mechanical engineering, computer science, computational sciences, and applied engineering . This paper proposes a simulation model to simulate and mimic the Waterfall SDLC development process from the analysis to the maintenance phase using the Simphony. NET computer simulation tool. The model simulates the different stakeholders involved in the Waterfall model which are essential throughout the whole development process.

The waterfall model comprises of five consecutive phases such as business analysis, design, implementation, testing and maintenance. This method was a success and many development firms and industrial manufactures have adopted this as their prime development framework. Moreover, besides this many people were hired to run different departments and the departments were its phases. It was very optimal to find out number of resources which should be assigned to complete the specific phases so, this created much fuss among project managers and the directors.

II. The Waterfall Model: Introduction, History of software engineering

A. *Software engineering: a history*

The history of software engineering begins around the 1960s. Writing software has evolved into a profession concerned with how best to maximize the quality of software and of how to create it. Quality can refer to how maintainable software is, to its stability, speed, usability, testability, readability, size, cost, security, and number of flaws or "bugs", as well as to less measurable qualities like elegance, conciseness, and customer satisfaction, among many other attributes. How best to create high quality software is a separate and controversial problem covering software design principles, so-called "best practices" for writing code, as well as broader management issues such as optimal team size, process, how best to deliver software on time and as quickly as possible, work-place "culture", hiring practices, and so forth. All this falls under the broad rubric of software engineering. In the 1960s and 1970s, the field of software engineering began to take shape.

Researchers and practitioners began to develop formal methods for software design and development, such as structured programming and the use of flowcharts to represent algorithms. In 1968, a conference on software engineering was held, and the term "software engineering" was officially coined.

In the following decades, software engineering continued to evolve and mature. The introduction of object-oriented programming in the 1980s led to a shift in how software was designed and developed. The 1990s saw the emergence of the Agile software development methodologies, which emphasized flexibility and responsiveness to change. Today, software engineering is a well-established field with its own set of best practices, methodologies, and tools. It is considered as a discipline of engineering and follows the same principles like any other engineering field.

In summary, software engineering has evolved from an art to a discipline with its own set of best practices, methodologies, and tools. The field has grown and matured over time, with new technologies and approaches being developed to improve the design and development of software.

B. The Waterfall Model

Winston Royce introduced the Waterfall Model in 1970.This model has five phases: Requirements analysis and specification, design, implementation, and unit testing, integration and system testing, and operation and maintenance. The steps always follow in this order and do not overlap. The developer must complete every phase before the next phase begins. This model is named "**Waterfall Model**", because its diagrammatic representation resembles a cascade of waterfalls.

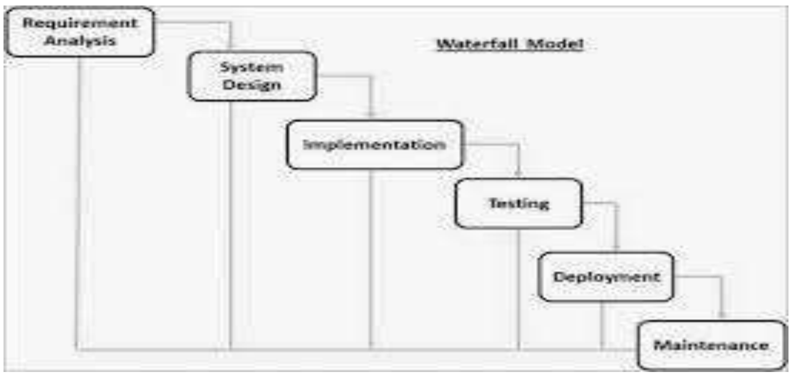
1. **Requirements analysis and specification phase:** The aim of this phase is to understand the exact requirements of the customer and to document them properly. Both the customer and the software developer work together so as to document all the functions, performance, and interfacing requirement of the software. It describes the "what" of the system to be produced and not "how."In this phase, a large document called **Software Requirement Specification (SRS)** document is created which contained a detailed description of what the system will do in the common language.
2. **Design Phase:** This phase aims to transform the requirements gathered in the SRS into a suitable form which permits further coding in a programming language. It defines the overall software architecture together with high level and detailed design. All this work is documented as a Software Design Document (SDD).
3. **Implementation and unit testing:** During this phase, design is implemented. If the SDD is complete, the implementation or coding phase proceeds smoothly, because all the information needed by software developers is contained in the SDD.

   During testing, the code is thoroughly examined and modified. Small modules are tested in isolation initially. After that these modules are tested by writing some overhead code to check the interaction between these modules and the flow of intermediate output.
4. **Integration and System Testing:** This phase is highly crucial as the quality of the end product is determined by the effectiveness of the testing carried out. The better output will lead to satisfied customers, lower maintenance costs, and accurate results. Unit testing determines the efficiency of individual modules. However, in this phase, the modules are tested for their interactions with each other and with the system.
5. **Operation and maintenance phase:** Maintenance is the task performed by every user once the software has been delivered to the customer, installed, and operational

When to use SDLC Waterfall Model?

Some Circumstances where the use of the Waterfall model is most suited are:

- When the requirements are constant and not changed regularly.
- A project is short
- The situation is calm
- Where the tools and technology used is consistent and is not changing
- When resources are well prepared and are available to



Requirement Analysis: Business requirements are gathered in this phase. This phase is the main focus of the project managers and stake

holders. Meetings with managers, stake holders and users are held in order to determine the requirements like; Who is going to use the system? How will they use the system? What data should be input into the system? What data should be output by the system? These are general questions that get answered during a requirements gathering phase. After requirement gathering these requirements are analyzed for their validity and the possibility of incorporating the requirements in the system to be

Analysis phase: Analysis phase is also known as software requirement specifications (SRS) which is a complete description of the behaviour of the software which is going to be developed. This phase wants business analyst to define both functional and non-functional requirements.

Functional requirements include requirements such as purpose, scope, perspective, functions, software attributes, user characteristics and database requirements. On the other hand, non- functional requirements include constraints, limitations, requirements on design and operation of the software. It has properties like the reliability, scalability, testability, performance, and quality standards, etc.

Design phase: This phase includes the process of planning and problem solving for software solution. It means that the software developers and the designers are going to define the plan for a solution, and it includes algorithm design, software architecture design, logical diagram scheme, data structure definition, etc. The phase is about designing the software which includes furthermore efforts to design the software.

Implementation phase: It refers to the understanding of business requirements and designing requirements into a solid execution program, database, website through programming and deployment. This is where the real code is written and compiled into operational application, from where the database and text files were created. In a nutshell, it means conversion of the process phase into production phase.

Testing phase: This phase is also known as verification and validation which includes a process for checking that the software expectations meet the original performance and specifications and it completes its intended purpose. Verification refers to the process where the process of evaluation of software is done to determine whether the product at the given phase satisfies the conditions which were there in the start. Validation, on the other hand refers to the process of evaluating he software during and at the end of development process to find that the software satisfies the specified requirements. In this phase the bugs and system glitches are found, and they are corrected, redefined accordingly.

Maintenance phase: this phase includes the process of modifying a software solution after delivery and deployment to refine the output, correct the errors and improve performance and quality. This can also include adaption of software to its environment, accommodating new user requirements and increasing its reliability, etc.

Applications

The waterfall model is widely and extensively used in the software industry some given below point explains the usage of the Waterfall model.

- The requirement is very well clear and fixed
- Product Definition is stable and well defined
- No ambiguity in the requirements
- Due to the rigidity of the model projects are short

Advantages of Waterfall models

- Simple and Easy to understand.
- Clearly and defined stages
- Easy to arrange tasks.
- Process and results are well documented.
- Works well when the project is small.
- Easy to manage due to the rigidity of the model.

Dis-Advantages of Waterfall models.

- The high number of risks and uncertainty
- Poor model for long and ongoing projects
- Not suitable for life-critical systems because of the rigidity
- Difficult to manage projects progress within stages.
- Not suitable for the project that they are following an Object-oriented approach.
- Cannot accommodate the requirement change process.

III. The Waterfall Model at the Company

The waterfall model used at the company runs through the phases requirements engineering, design & implementation, testing, release, and maintenance.

Between all phases the documents have to pass a quality check, this approach is referred to as a stage-gate model . An overview of the process is shown in Figure 1.

We explain the different phases and provide a selection of checklist-items to show what type of quality checks are made in order to decide whether the software artifact developed in a specific development phase can be passed on to the adjacent phase. The Waterfall Model in Large-Scale Development 389 Main Product Line Requirements Engineering Testing Release Maintenance Design &Implementation

Quality Door(Checklist)Main Development Project Quality Door(Checklist)Quality Door(Checklist) Quality Door(Checklist)

Requirements Engineering: In this phase, the needs of the customers are identified and documented on a high abstraction level. Thereafter, the requirements

are refined so that they can be used as input to the design and implementation phase. The requirements (on high as well as low abstraction level) are stored ina requirements repository. From this repository, the requirements to be implement dare selected from the repository. The number of requirements selected depends on the available resources for the project. As new products are not built from the scratch, parts from the old product (see main product line in Figure 1)are used as input to the requirements phase as well. At the quality gate (among others) it is checked whether all requirements are understood, agreed upon, and documented. Furthermore, it is checked whether the relevant stakeholders are identified and whether the solution would support the business strategy.

Design and Implementation: In the design phase the architecture of the system is created and documented. Thereafter, the actual development of the system takes place. The developers also conduct basic unit testing before handing the developed code over to the test phase. The quality gate checklist (among others) verifies whether the architecture has been evaluated, whether there are deviations from the requirements compared to the previous quality gate decision, and whether there is a deviation from planned time-line, effort, or product scope.

Testing: In this phase the system integration is tested regarding quality and functional aspects. In order to make a decision whether the the system can be deployed, measures of performance (e.g, throughput) are collected in the test laboratory. As the company provides complete solutions (including hardware and software) the tests have to be conducted on a variety of hardware and software configurations as those differ between customers. The outcome of the phase is reviewed according to a checklist to see whether the system has been verified and whether there are deviations from previous quality gate decisions in terms of quality and time, whether plans for hand-over of the product to the customer are defined according to company guidelines, and whether the outcome of the project meets the customers' requirements.

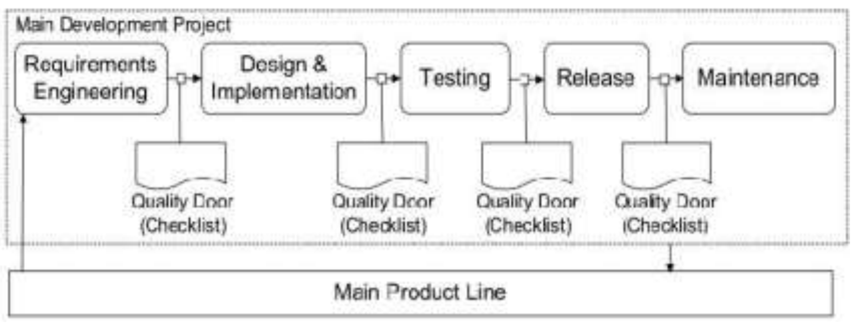


Fig. 1. Waterfall Development at the Company have to be programmed. Build-instructions can be used to enable and disable features of the main product line to tailor the system to specific customer needs.

At the quality gate (among others) it is checked whether the outcome meets the customers' requirements, whether the customer has accepted the outcome, and whether the final outcome was presented in time and fulfilled its quality requirements. A post-mortem analysis has to be performed as well.

Maintenance: After the product has been released to the customer it has to be

maintained. That is, if customers discover problems in the product they report them to the company and get support in solving them. If the problems are due to faults in the product, packages for updating the system are delivered to the customers.

IV. AGILE METHODOLOGIES AND THE WATERFALL MODEL :

describes the traditional waterfall life cycle as a linear, phased approach to software development, while he sees an Agile approach which seems to be diametrically opposed to this traditional development life cycle. The following sections will shed some light on deciding how projects should be developed, Agile or Waterfall.

A. Archetypal projects

According to Unhelkar [17], projects that benefit the most from an Agile approach are "greenfield" development projects. A greenfield project being a project that aims to develop a system for a totally new environment.

Furthermore, Agile approaches to software development are most conducive in these projects wherein coding is at the center of activities [17]. Typically, a small project comprising five programmers and lasting for a period of about 6 months would be ideally placed to make extensive use

of Agile principles and practices, [17]. An experimental or pilot project in the small category is also likely to benefit with Agility. Such a project can be used to experiment with Agile itself, or may be a part of a new, major development [17]. According to

Petersen et al. [12] the Waterfall Model is predictable and pays attention to planning the architecture and structure of the software system in detail which is especially important when dealing with large systems.

B. Agile and plan-driven project characteristics

How about other project types, which project characteristics will make a difference when it comes to choosing the right development methodology?

1. Primary goals: Boehm [3] states a major set of objectives for the more traditional plan-driven methods, has been predictability, repeatability, and optimization. This while Agility focuses on rapid value and responding to change.
2. Size: Plan-driven methods scale better to large projects as opposed to Agile projects. A bureaucratic, plan-driven organization that requires an average of a person-month just to get a project authorized and started won't be very efficient on small projects though.
3. Customer relations: Agile methods work best when customers operate in dedicated mode with the development team, and when their tacit knowledge is sufficient for the full span of the application .This method risks tacit knowledge shortfalls, which the plan-driven methods reduce via documentation and the use of architecture review boards and independent expert project reviews to compensate for on-site customer shortfalls .
4. Planning and control: According to Unhelkar formal project management plays an important role in successful completion of a software project. Project management requires careful planning, estimation, coordination, tracking, and control. Aspects formally covered in The Waterfall Model . Agility places more value on the planning process than on the resulting documentation .
5. Communications: Agility advocates face-to-face communication, the Waterfall Model requires explicit documented knowledge.
6. Requirements: Formal and solution-independent, up- front requirements engineering is not directly used by pure Agile practitioners [17]. The heavyweight, formal Waterfall Model will encounter problems keeping up with rapidly changing requirements. On the other hand, if the architecture anticipates and accommodates requirements changes, plan-driven methods can keep even million-line applications within budget and schedule.
7. Development: Agility values working software over comprehensive documentation, and emphasizes simplicity, maximizing the amount of work not done [6]. This while the Waterfall Model relies heavily upon software architecture, as it is part of the development sequence, see.
8. Test: According to Beznosov and Kruchten [2], conventional assurance methods involve design and architectural principles, dynamic testing, static analysis and internal and third-party reviews, evaluation, and vulnerability testing. These methods are much more adapted to Waterfall development which are well documented and focused on architecture [2]. Agile Methodologies, on the other hand, facilitate internal design and code review, and motivate developers to adopt coding standards , but lack the focus on architecture or documentation.
9. Customers: Agility requires dedicated, co-located, knowledgeable customers . The Waterfall Model requires, adequately skilled knowledgeable customers .
10. Developers: Developers in an Agile project are to be agile knowledgeable, collocated, and collaborative and should be amicable, talented, skillful and communicative . Agile approaches emphasize cross-functional teams of developers, testers, subject matter experts, and architects . Waterfall developers are to be plan-oriented, adequately skilled with access to external knowledge

Conclusion

From the making of this practical work information system, it can be concluded that an application with a simple appearance has been made with the aim of facilitating practical work management so that with this application the process of doing practical work becomes more effective and efficient. In addition, the system has been tested with 13 tests using black cox testing and has valid results. The advice that can be given for the development of this system is that the system is still not integrated with the academic system, so it can be developed again by integrating it with the campus academic system.

REFERENCES

1. www.Google.com
2. www.chatgpt.com

3. New Idea In Waterfall Model For Real Time Software Development, International Journal of Engineering Research & Technology (IJERT), Vol. 2 Issue 4, April – 2013.

4. Waterfall Model Used in Software Development Reference: Software Requirements Engineering Waterfall Model, E3S Web of Conferences 328, 0 (2021),ICST 2021.

5. A Simulation Model for the Waterfall Software Development Life Cycle, International Journal of Engineering & Technology (iJET), ISSN: 2049-3444, Vol. 2, No. 5, 2012.

6. Waterfall Process Operations in the Fast-paced World: Project Management Exploratory Analysis, International Journal of Applied Business and Management Studies, Vol. 6, No. 1; 2021.

7. The Waterfall Model and the Agile Methodologies : A comparison by project characteristics, February 2017.

8. https://study.com/academy/lesson/concurrent- models-in-software-engineering-types- applications.html.