Ahsanullah University of Science & Technology

Department of Computer Science & Engineering

Course No:	CSE3223
Course Title:	Industrial System Design and Software Engineering
Assignment No:	01

Date of Submission:	31.12.2021
Submitted To:	Dr. -Ing. Nusrat Jahan Lisa

Submitted By-

Name:	S. M. Tasnimul Hasan
ID:	180204142
Section:	B
Year:	3rd
Semester:	2nd

Difference between Traditional Process Models and Agile Models

Traditional process models are based on pre-organized phases or stages of the software development lifecycle. Here the flow of development is unidirectional, from requirements to design and then to development, then to testing and maintenance. On the other hand, Agile approaches are precise and customer friendly. Users or Customers have the opportunity to make modifications throughout project development phases. Agile models propose an incremental and iterative approach to development.

The main difference between traditional and agile approaches is the sequence of project phases – requirements gathering, planning, design, development, testing and UAT. In traditional development methodologies, the sequence of the phases in which the project is developed is linear where as in Agile, it is iterative. To differentiate between Traditional Process Models and Agile Models I would like to choose System Development Life Cycle (SDLC).

System Development Life Cycle (SDLC):

The SDLC is a phased approach to analysis and design that holds that systems are best developed using a specific cycle of analyst and user activities. It is the overall process for developing information systems from planning and analysis through implementation and maintenance. The SDLC begins with a business need, followed by an assessment of the functions a system must have to satisfy the need, and ends when the benefits of the system no longer compensate its maintenance costs. Therefore, it is referred to as a '**lifecycle**'. The SDLC is generally divided into 7 phases –

Phase 1: Identifying Problems, Opportunities, and Objectives

Phase 2: Determining Human Information Requirements

Phase 3: Analyzing System Needs

Phase 4: Designing the Recommended System

Phase 5: Developing and Documenting Software

Phase 6: Testing and Maintaining the System

Phase 7: Implementing and Evaluating the System

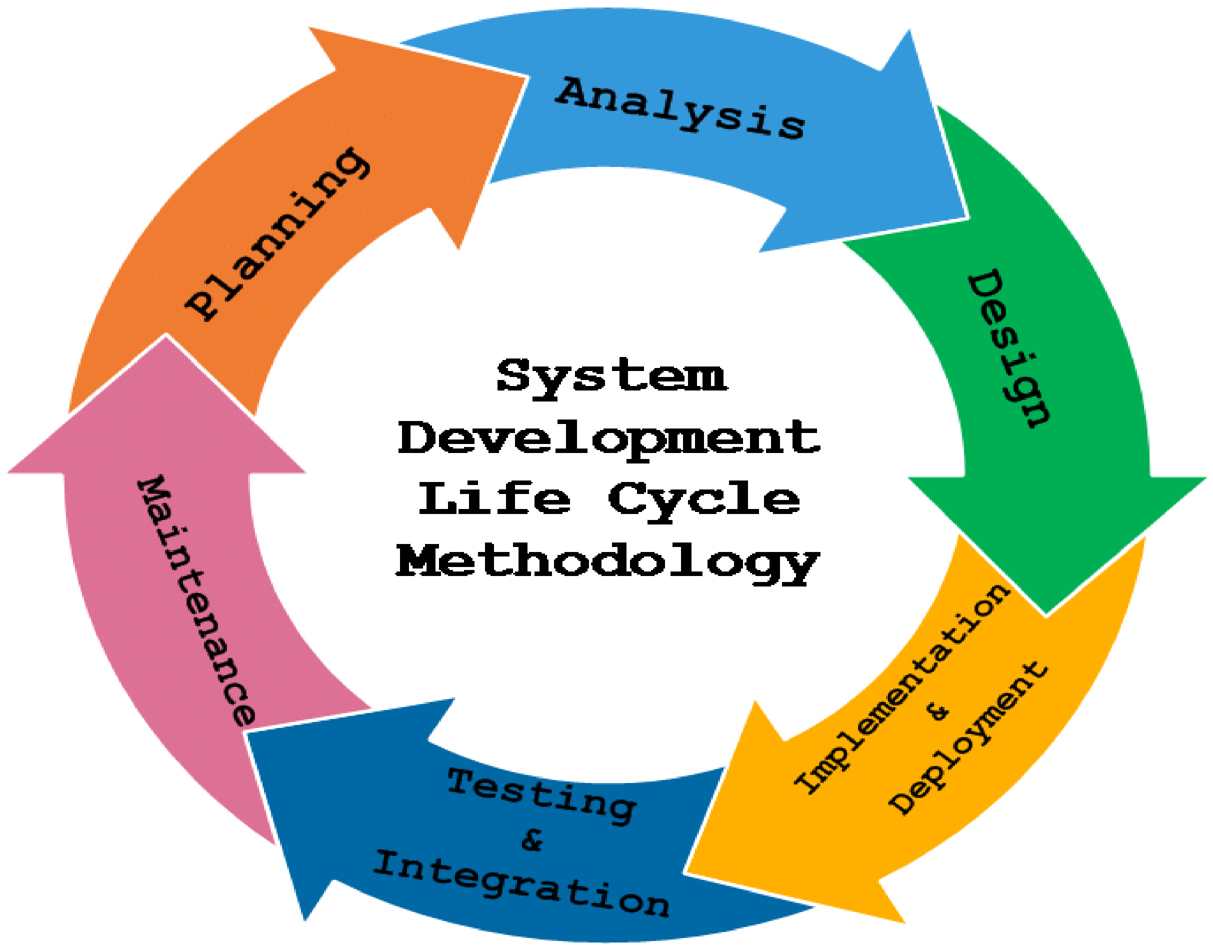
System Development Life Cycle Methodology
Planning
Analysis
Design
Implementation & Deployment
Testing & Integration
Maintenance

Figure: System Development Life Cycle (SDLC)

One of the approaches to the System Development Life Cycle is the traditional SDLC. Some of the traditional SDLC models which are followed during the software development process are defined below –

1. **Waterfall Model**: It is the oldest software lifecycle model where workflow is in a linear fashion. This model is used when requirements are well understood, and risk is low. It doesn't support iteration so changes can cause confusion. It requires customer patience because a working version of the program doesn't occur until the final phase. It is also difficult for the customers to state all requirements at the beginning of the development process.

2. **Incremental Model**: It is used when requirements are well understood. The workflow is in a linear fashion within an increment. This model is iterative in nature and focuses on an operational product with each increment which is delivered sooner, and the other components are delivered later. It is useful when staffing is too short for a full-scale development.

3. **Prototyping Model**: It is used when requirements are not well understood. So, it serves as a mechanism for identifying software requirements. Feedback received from the customers is used to refine the prototype. After the customer sees a "working version" of the software, they want to buy the prototype after a "few fixes" are made. Thus, developers often make implementation compromises to get the software running quickly. This model follows an evolutionary and iterative approach.

4. **Spiral Model**: It is used when requirements are not well understood, and risks are high. This model follows an evolutionary approach and operates as a risk-driven model. A go/no-go decision occurs after each complete spiral to react to risk determinations. This model serves as a realistic model for large-scale software development. At the same time, it requires considerable expertise in risk assessment.

5. **V-Model**: A variation in the representation of the waterfall model is called the V-model. The V-model depicts the relationship of quality assurance actions to the actions associated with communication, modeling, and early construction activities. The V-model provides a way of visualizing how verification and validation actions are applied to earlier engineering work.

Agile Models:

Agile methods were developed to overcome the weaknesses in traditional software engineering. In the modern economy, it is often difficult to predict how a system will evolve with time as market conditions change rapidly, end-user needs evolve, and new competitive threats emerge without warning. In many situations, it is impossible to define requirements fully before the project begins. Agile method adapts to the changes and delivers multiple working software builds after each iteration to collect customer feedback for adaptation. Agile models follow a set of principles and manifesto to achieve agility in the software development process. Another compelling characteristic of the agile approach is its ability to reduce the costs of change throughout the software process. Basically, agile model is a combination of iterative and incremental process models with focus on process adaptability and customer satisfaction by rapid delivery of working software product.

Agile methods involve the following basic framework activities –

1. Communication
2. Planning
3. Modelling
4. Construction
5. Deployment

Various agile process models have been suggested and used across the industry such as –

1. **Extreme Programming (XP):** It is the most widely used agile process which uses an object oriented approach. This process consists of XP planning, XP design, XP coding, and XP testing.

2. **Adaptive Software Development (ASD):** It focuses on human collaboration and team self organization as a technique to build complex software and system. Three phases of ASD are- Speculation, Collaboration, and learning.

3. **Scrum:** It is an agile process model where testing and documentation are on-going as the product is being constructed. Work units occurs in "sprints" and is derived from a "backlog" of existing changing prioritized requirements. Changes are not introduced in sprints but in backlog.

4. **Dynamic Systems Development Method (DSDM):** It is an agile software development approach that provides a framework for building and maintaining systems which meet tight time constraints using incremental prototyping in a controlled project environment.

5. **Crystal:** It is a family of process models that allow "maneuverability" based on problem characteristics. This model suggests the use of "reflection workshops" to review the work habits of the team. Face-to-face communication is emphasized in this process model.

6. **Feature Driven Development (FDD):** It is an object-oriented software engineering process model. It put emphasis on defining features which can be organized hierarchically. A features list is created and "plan by feature" is conducted. A feature template is also used.

7. **Agile Modeling (AM):** It suggests a set of agile modeling principles such as- Model with a purpose, use multiple models, know the models and the tools you use to create them, adapt locally, content is more important than representation etc.

There is considerable debate about the benefits and applicability of agile software development as opposed to the traditional software engineering processes. Jim Highsmith, the creator of **Adaptive Software Development** stated that, "Traditional methodologists are a bunch of stick-in-the-muds who'd rather produce flawless documentation than a working system that meets business needs." As a counterpoint, he stated that "Agile methodologists are a bunch of glorified hackers who are going to be in for a heck of a surprise when they try to scale up their toys into enterprise-wide software." Let us consider a scenario to differentiate between traditional SDLC and Agile models. Microsoft is working on a project to come up with an online video sharing platform like YouTube. The system will include all the features provided by YouTube and any other features requested by the marketing team. The main functions of the system are given below –

1. Search for and watch videos
2. Create a personal channel
3. Upload videos to channel
4. Like/Comment/Share other videos
5. Subscribe/Follow other channels and users
6. Create playlists to organize videos and group videos together
7. Show recommended videos

The final product needs to be ready in 10 months of time.

In traditional Waterfall model –

1. At a high level, the project teams would spend 15% of their time on gathering requirements and analysis (1.5 months)
2. 20% of their time on design (2 months)
3. 40% on coding (4 months) and unit testing
4. 20% on system and integration testing (2 months)
5. At the end of this cycle, the project may also have 2 weeks of user acceptance testing by marketing teams.
6. In this approach, the customer does not get to see the end product until the end of the project, when it becomes too late to make significant changes.

In Agile development model –

1. Each project is broken up into several ‘Iterations’.
2. All Iterations should be of the same time duration (between 2 to 8 weeks).
3. At the end of each iteration, a working product should be delivered.
4. In simple terms, in the Agile approach the project will be broken up into 10 releases (assuming each iteration is set to last 4 weeks).
5. Rather than spending 1.5 months on requirements gathering, in agile software development, the team will decide the basic core features that are required in the product and decide which of these features can be developed in the first iteration.
6. Any remaining features that cannot be delivered in the first iteration will be taken up in the next iteration or subsequent iterations, based on priority.
7. At the end of the first iterations, the team will deliver a working software with the features that were finalized for that iteration.
8. There will be 10 iterations and at the end of each iteration the customer is delivered a working software that is incrementally enhanced and updated with the features that were shortlisted for that iteration.

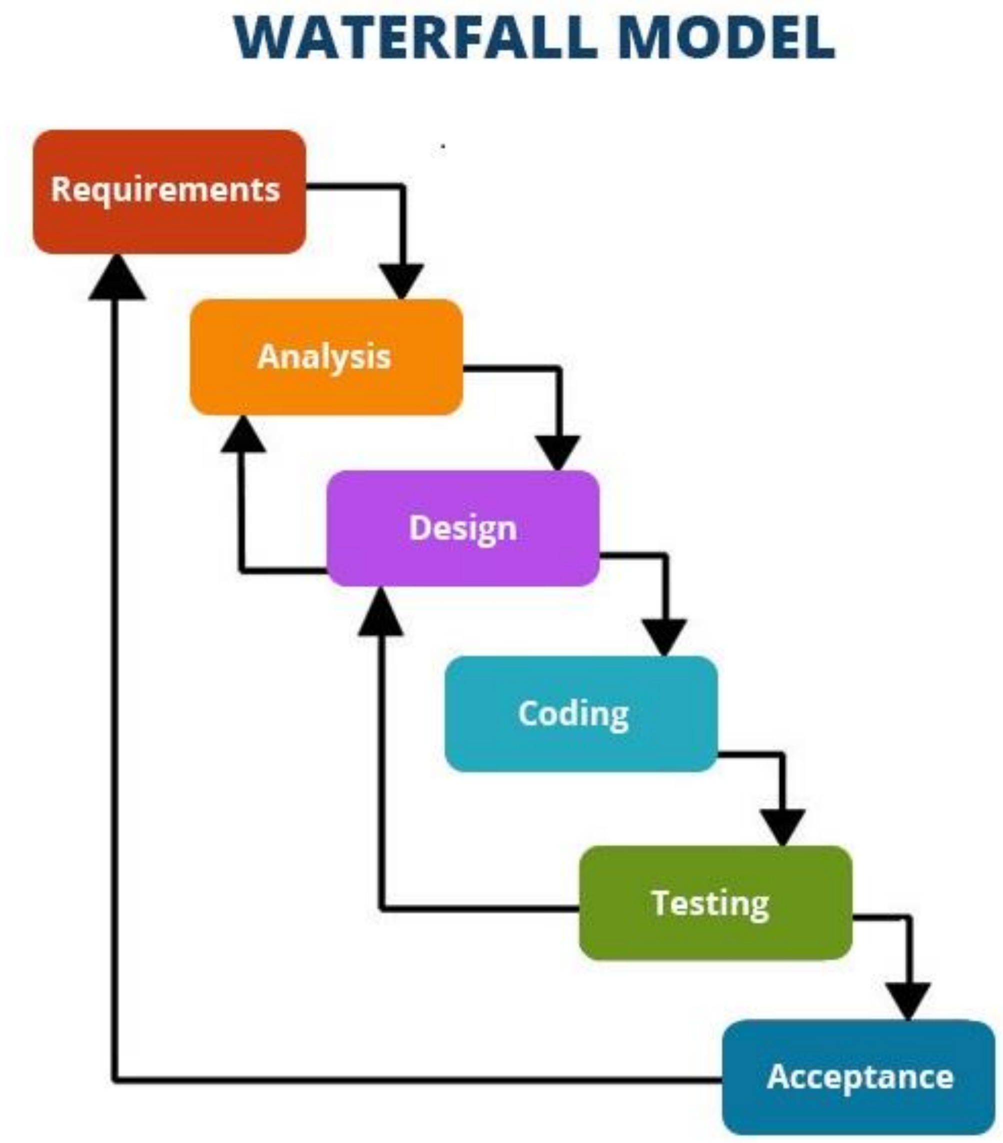

Figure: Traditional SDLC Model (Waterfall)

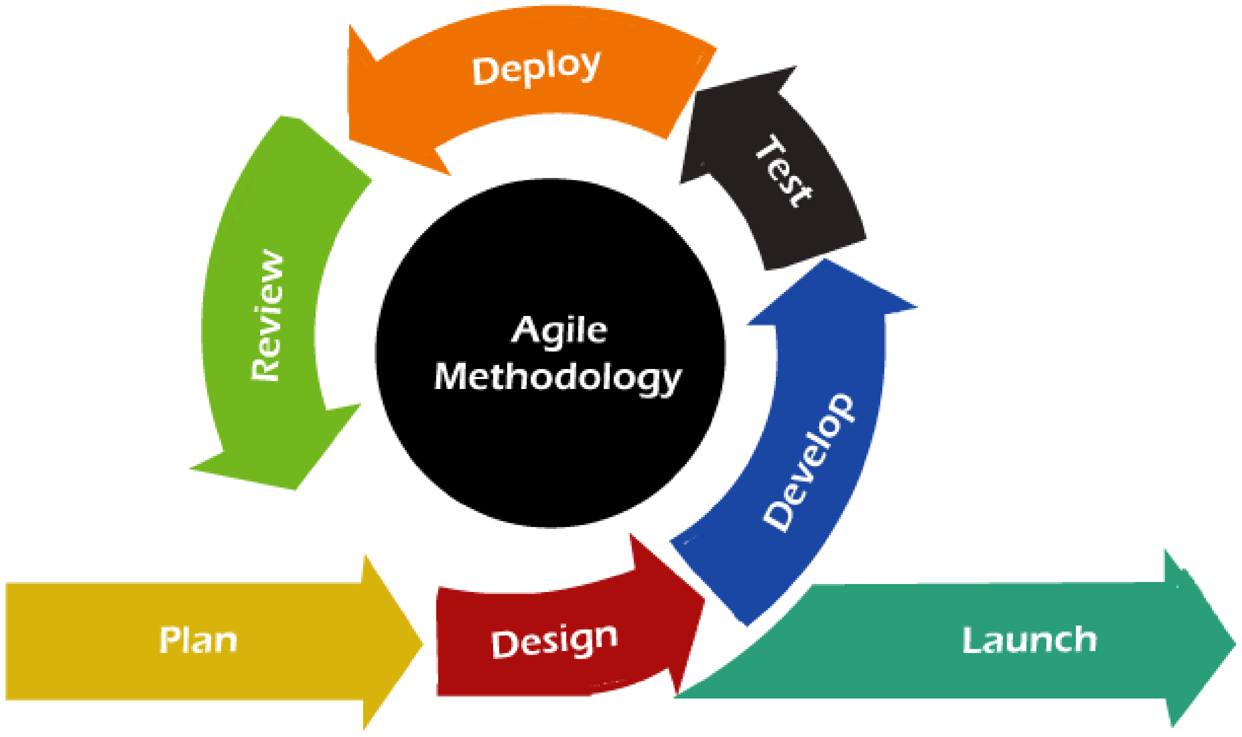

Figure: Agile Model

Agile is based on the adaptive software development methods, whereas the traditional SDLC models like the waterfall model is based on a predictive approach. If Microsoft chooses a traditional SDLC model, the predictive teams will work with detailed planning and have a complete forecast of the exact tasks and features to be delivered in the next few months or during the product life cycle. These predictive methods entirely depend on the requirement analysis and planning done in the beginning of cycle. Any changes to be incorporated go through a strict change control management and prioritization.

On the other hand, agile uses an adaptive approach where there is no detailed planning and there is clarity on future tasks only in respect of what features need to be developed. If Microsoft chooses an agile model, there will be feature driven development and the team will have to adapt to the changing product requirements dynamically. The product will need to be tested very frequently, through the release iterations, minimizing the risk of any major failures in future.

Customer Interaction is the backbone of this Agile methodology, and open communication with minimum documentation are the typical features of Agile development environment. The agile teams within Microsoft will work in close collaboration with each other and will most often be in the same geographical location.

There is a set of guidelines that can help Microsoft choose which method to use when developing their video sharing system. The SDLC approach –

- Systems have been developed and documented using SDLC
- It is important to document each step of the way
- Upper-level management feels more comfortable or safe using SDLC
- There are adequate resources and time to complete the full SDLC
- Communication of how new systems work is important Agile Methodologies –

- There is a project champion of agile methods in the organization
- Applications need to be developed quickly in response to a dynamic environment
- The customer is satisfied with incremental improvements
- A rescue takes place (the system failed and there is no time to figure out what went wrong)
- Executives and analysts agree with the principles of agile methodologies

The waterfall model is not the right process model in situations where informal team communication is possible and software requirements change quickly. Agile methods are better for these systems. Agile development method is better than a waterfall approach for systems whose requirements are likely to change during the development process. This is the case for most business systems and software products as we rarely work out a complete problem solution in advance but move toward a solution in a series of steps, backtracking when we realize that we have made a mistake. It will be cheaper and easier to make changes in the software as it is being developed by using agile model in developing the video sharing platform. The customer or user will be able to evaluate the system at a relatively early stage in the development to see if it delivers what is required. If not, then only the current increment has to be changed and, possibly, new functionality will be defined for later increments.

Agile approaches focus on the code being developed and deliberately minimize formality and documentation. This approach to software development considers design and implementation to be the central activities in the software process. They incorporate other activities, such as requirements elicitation and testing, into design and implementation. By contrast, a traditional SDLC approach to software engineering identifies separate stages in the software process with outputs associated with each stage. The outputs from one stage are used as a basis for planning the following process activity.

In a traditional software development process, iteration occurs within activities, with formal documents used to communicate between stages of the process. For example, the requirements of the video sharing system will evolve, and ultimately, a requirements specification will be produced. This will be then an input to the design and implementation process. In an agile approach, iteration occurs across activities. Therefore, the requirements and the design are developed together rather than separately.

Therefore, agile model has three major advantages over the traditional model-

1. The cost of implementing requirements changes is reduced. The amount of analysis and documentation that has to be redone is significantly less than the traditional model.

2. It is easier to get customer feedback on the development work that has been done. Customers can comment on demonstrations of the software and see how much has been implemented. Customers find it difficult to judge progress from software design documents.

3. Early delivery and deployment of useful software to the customer is possible, even if all the functionality has not been included. Customers can use and gain value from the software earlier than is possible with a traditional process.

From a management perspective, the agile approach has two problems:

1. The process is not visible. Managers need regular deliverables to measure progress. If systems are developed quickly, it is not cost effective to produce documents that reflect every version of the system.

2. System structure tends to degrade as new increments are added. Regular change leads to messy code as new functionality is added in whatever way is possible. It becomes increasingly difficult and costly to add new features to a system. To reduce structural degradation and general code messiness, agile methods suggest regular refactoring of (improve and restructure) the software.

To develop the video sharing system using traditional SDLC models, Microsoft has to adapt organizational standards for coding, naming etc. to reduce interface time and errors. On contrary, it can adopt pair programming for agile models. The nonproductive expansion of work can be reduced by project management and establishing deadlines in traditional SDLC methodologies. This can be achieved by limiting scope in each release in agile methodologies. Agile processes allow an onsite customer to reduce data and knowledge search and storage time and costs where the traditional SDLC processes use structured data gathering techniques, such as interviews, observation, sampling etc. Timeboxing is used in agile methodologies to reduce communication and coordination time and cost. On the other hand, traditional SDLC methodologies separate projects into smaller task to establish barriers. Microsoft will be able to reduce losses from human information overload by sticking to a 40-hour workweek using agile approach. Applying filtering techniques to shield analysts and programmers can be implemented to achieve this in traditional SDLC approach.

Today, software work is fast-paced and subject to a never-ending stream of changes. Thus, agile models can serve as a useful process model in today’s situations. Ambler, an influential agile method developer suggested that “An organization moving to agile methods can expect to see productivity improvement across the organization of about 15% over 3 years, with similar reductions in the number of product defects.”