**Module 2- Critical Assignment**

Anju Mishra

Colorado State University Global

CSC 505: Principles of Software Development

Dr. [Steven Evans](https://csuglobal.instructure.com/courses/104932/users/1373)

Due Date: 2.1.2025

**Content**

1. Introduction
2. UML Diagram to show shortcoming of waterfall
3. Python code to show additions in the Waterfall to share shortcoming
4. Python code execution screenshot
5. Github link
6. References

Introduction

The Waterfall model, often considered the traditional approach to software development, follows a linear and sequential design process. While it can work for simple projects with well-defined requirements, it presents significant limitations when applied to complex or evolving systems. These limitations include:

1. **Inflexibility**: Once a phase is completed, it is difficult to go back and make changes without disrupting the entire project timeline.
2. **Late Testing**: Testing typically happens late in the development cycle, which can lead to costly fixes when defects are discovered.
3. **Lack of Stakeholder Feedback**: In Waterfall, there is limited opportunity for customer or stakeholder involvement once the project starts.
4. **Assumption of Clear Requirements**: The model assumes that all requirements are understood at the beginning, which may not be the case for many projects.

To better illustrate these issues, I will demonstrate with a UML diagram and Python code that visually represent how Waterfall's rigid structure can be an issue however with suggested modification issues can be mitigated.

UML Diagram to show shortcoming of waterfall

UML diagram (Unified Modeling diagram) shown below is the showing the waterfall modle steps.

A diagram of a process

AI-generated content may be incorrect.*Figure 1: Waterfall model steps*

To overcome the shortcomings of above-mentioned model, I am approaching step by step and showing shortcoming and solutions details below:

1. **Requirements Gathering:** Instead of trying to define every detail upfront, focus on high-priority features and define with customer with collaboration.
2. **Planning: Instead of planning upfront total project,** break the project into smaller, manageable chunks (sprints).
3. **Modeling & Construction**: Instead of creating a detailed, final design at the beginning, consider developing a prototype or a minimum viable model. Rather than assuming all requirements are fully understood and fixed upfront, the improved modeling phase incorporates **requirements validation** at each iteration which I have mentioned as develop and test.
4. **Deployment**: In the Waterfall model, deployment happens only after the entire development cycle is completed, and evaluation typically occurs after the product is fully developed, which often leads to discovering issues late in the process. In suggested steps deployment is done in smaller, iterative cycles (e.g., sprints), with frequent evaluations after each cycle.

A diagram of a project

AI-generated content may be incorrect.I have used Giffy ( <https://go.gliffy.com/go/html5/13867614> ) to indicate the shortcoming of this step and improvement:

Figure 2: UML of Waterfall and Waterfall ver 2

Python code:

Below is the python code which is showing waterfall and waterfall version 2.

class Mishra:  
 def \_\_init\_\_(self, name):  
 self.name = name.lower()  
  
 def process(self):  
 if self.name == "waterfall":  
 steps = [  
 "Communication",  
 "Planning",  
 "Modeling",  
 "Construction",  
 "Deployment"  
 ]  
 description = "Sequential process, each phase must be completed before moving to the next."  
 elif self.name == "waterfall\_ver2":  
 steps = [  
 "Meet",  
 "Plan",  
 "Develop & Test",  
 "Evaluate"  
 ]  
 description = "Iterative, flexible, and adaptable approach with continuous customer involvement."  
 else:  
 print("\n⚠️ Invalid choice! Please enter 'Waterfall' or 'Waterfall\_ver2'.")  
 return  
  
 self.display\_steps(steps, description)  
  
 def display\_steps(self, steps, description):  
 print("\n" + "=" \* 40)  
 print(f" {self.name.capitalize()} Methodology Steps")  
 print("=" \* 40)  
 for step in steps:  
 box\_width = max(len(step) + 4, 25)  
 print("+" + "-" \* (box\_width - 2) + "+")  
 print(f"| {step.center(box\_width - 4)} |")  
 print("+" + "-" \* (box\_width - 2) + "+")  
 print("\n" + "-" \* 40)  
 print(f"📌 Note: {description}")  
 print("-" \* 40 + "\n")  
  
# Run script with user input  
if \_\_name\_\_ == "\_\_main\_\_":  
 user\_choice = input("Enter methodology ('Waterfall' or 'Waterfall\_ver2'): ").strip()  
 methodology = Mishra(user\_choice)  
 methodology.process()

Python code execution screenshot:

In this screenshot if user put input as Waterfall, then it will show the steps of the waterfall model shown in figure 1:

A screenshot of a computer

AI-generated content may be incorrect.

Figure 3: Code showing Waterfall steps

In this screenshot if user put input as Waterfall ver 2 then it will show the improvements suggested steps of the waterfall version 2 model:

A screenshot of a computer

AI-generated content may be incorrect.

Figure 4: Waterfall improved version- Waterfall ver\_2

In this screenshot if user put input other than Waterfall or Waterfall\_ver 2 then it will show invalid choice.

A screen shot of a computer

AI-generated content may be incorrect.

Figure 5: Showing invalid choice selection in code

Github link:

I have made a github link where I have uploaded the python code and put this document for reference. Also included the UML.

<https://github.com/anjumishra16/CSC-505>

Reference:

* Simplilearn. (n.d.). Agile vs. Scrum: Key differences you should know. Simplilearn.

https://www.simplilearn.com/agile-vs-scrum-article

* Planview. (n.d.). Disadvantages of Agile. Planview. Retrieved February 1, 2025, from

https://www.planview.com/resources/articles/disadvantages-agile/

* Pressman, R. S. (2020). Software engineering: A practitioner’s approach (9th ed.). McGraw Hill.