Software products could be seen across all corners of the globe. The software industry forms the very foundation on which our modern civilization is built. They have become the engine driving mankind toward the future. Likewise, software development methods are also progressing, with new principles and techniques being invented and improved constantly. This essay will discuss the two models of Waterfall and Iterative and Incremental (IID), including their definitions, strengths, and weaknesses.

Software development life cycle models illustrate how to traverse the complicated process of software development. The quality, timeliness, cost, and ability to satisfy clients’ expectations of a software product are all influenced by the model chosen (Shiklo, 2019). The most common models employed in modern professional software development are the Waterfall and IID model.

The Waterfall model was proposed by Royce in 1970 and is often considered a classical or traditional approach. According to Singh et al. (2015), the Waterfall model is linear and sequential, which means it emphasizes planning in the early stages and ensures design flaws before they develop. This methodology starts with defining system and software requirements before moving on to system and software design, coding, testing, and maintenance. In the first stage of requirement analysis, consultation with system users determines the system's services, limitations, and goals, which are then carefully described and used to create a system specification. Next, the design phase creates a system architecture and essential software abstractions. The testing stage ensures the product achieves its specifications and requirements have been met. In the final operation and maintenance phase, the system is installed, errors that were not previously discovered are fixed and the system could be improved in the future (Sommerville, 2015).

Being among the first to be developed, Waterfall serves as a frame for later models as it has some advantages of a predecessor. Singh et al. (2015) noted that the Waterfall model is simple and easy to understand, therefore widely used by inexperienced developer teams for mature products. This approach also identifies clear milestones and relies on documentation, which means a reduction in flaws in later development stages. However, the Water model suffers from a range of weaknesses associated with such a strict progression cycle. Hughey (2009) pointed out that this model makes it difficult for customers to set requirements for the products due to the abstract level of a functional specification, and they will only fully understand what is required after the application is delivered. Re-engineering the application becomes extremely difficult (and expensive) at this point. Furthermore, Waterfall does not account for needs that may change over the development cycle and takes significantly longer to develop a project compared to the IID model.

Sommerville (2015) wrote that the Waterfall model is suitable only for certain systems, which include embedded systems, critical systems, and large software systems that are part of broader engineering systems developed by several partner companies. In circumstances where informal team communication is available and software requirements change often, the waterfall approach is not the ideal process model. These systems benefit from iterative development and agile approaches.

For development, the IID model uses both iterative design or iterative technique and incremental build model. The main idea behind this technique is to create a system in smaller portions over time, enabling the possibility to utilize previous sections or versions of the system. Wherever possible, important stages of the process should begin with a rudimentary implementation of a part of the software requirements and iteratively improve the subsequent versions until the whole system is completed. With each iteration, design modifications are made and new functional capabilities are added. (Farcic, 2014). This implies that the client or user may assess the system at an early stage in its development to see if it meets their needs. If not, just the current increment must be modified, with additional functionality perhaps created for subsequent increments (Sommerville, 2015).

IID model has become more and more popular thanks to its numerous advantages, especially over older models such as Waterfall. Sommerville (2015) confirmed that by applying the IID approach, the overall cost of adding modifications is reduced and the process of analysis and documentation takes less repetition. Furthermore, customer feedback is easier to collect as clients could closely monitor the progress of a project. Finally, even if all of the functionality has not been incorporated, early delivery and deployment of usable software to the clients are conceivable. On the other hand, the IID model nevertheless contains some limits, especially from the management perspective (Sommerville, 2015). Development progress is not always clear, thus making periodical documentation not economical if the system is delivered in a short amount of time. On top of that, as additional increments are added, the system's structure begins to deteriorate because regular modification leads to chaotic code. Adding new features to a system gets increasingly complex and expensive, restricting the capability of the IID model.

In contrast with the Waterfall model, the IID model is not suitable for large and complicated systems as they require a solid framework or architecture, and the roles and responsibilities of the multiple teams working on different aspects of the system must be clearly defined to that architecture. This must be planned ahead of time rather than created in stages (Sommerville, 2015).

In conclusion, both models have their advantages and disadvantages. In the software industry, a combination of these models is used to achieve maximum efficiency. Choosing the right model is critical since the product must be supplied on time and meet the necessary quality standards.

# References

FARCIC, V., 2014. Software Development Models: Iterative and Incremental Development [Online]. *Technology Conversations.* 21 January.[viewed 22 September 2021]. Available from: https://technologyconversations.com/2014/01/21/software-development-models-iterative-and incremental-development/

HUGHEY, D., 2009. *Comparing Traditional Systems Analysis and Design with Agile Methodologies,* St. Louis: University of Missouri.

SHIKLO, B., 2019. *8 Software Development Models: Sliced, Diced and Organized in Charts* [Online]. Science Soft. [viewed 22 September 2021]. Available from: https://www.scnsoft.com/blog/software-development-models

SINGH, D., THAKUR, A. & CHAUDHARY, A., 2015. A Comparative Study between Waterfall and Incremental Software. *International Journal of Emerging Trends in Science and Technology.* **2**(4), pp. 2202-2208.

SOMMERVILLE, I., 2015. *Software Engineering.* 10th ed. Harlow: Pearson Education Limited.