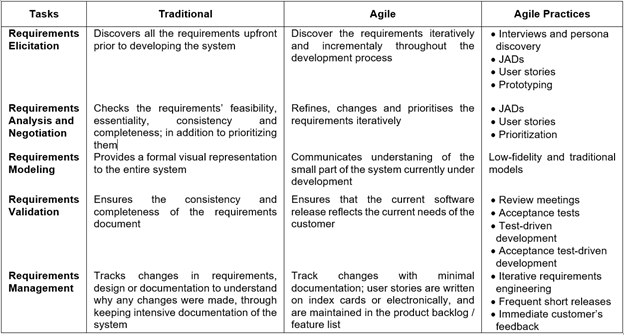
Nowadays, the software industry forms the very foundation on which our modern civilization is built. They have become the engine driving humanity toward the future. In software engineering, it is essential to set requirements for a particular product, which involves the process of requirements engineering. This essay will discuss and compare traditional (Waterfall) requirements and requirements in Agile context.

Requirements engineering could be interpreted as a series of processes, including identifying, inspecting, and reporting requirements and limitations of a software system. Requirements engineering plays a crucial role in software development, as defining the necessary functions of software is difficult due to the complex nature of systems.

In the Waterfall model development, all phases are sequential and each must be completed before moving on. Requirements engineering remains the very first stage of the Waterfall method and examines both the software and system requirements. Processes within requirements engineering are not defined as they could variate depending on the software developed. According to Sommerville, they overlap each other and include elicitation, analysis, specification, validation, and modification (2015, pp. 113). Requirements elicitation outlines the possible system specifications and necessities, which form the basis of a system and includes different methods such as interviews, ethnography, and prototyping. The next step is analysis, which involves examining whether statements acquired in the previous step are harmonious and achievable. The analysis stage could further be broken down to prioritization, which shifts the focus to satisfying essential constraints, and modeling, which further clarifies descriptions. After that, specification comes and establishes specific requirements. The validation process investigates the correctness and fulfillment of requirements, culminating in a requirements document. In the final stage of modification, requirements are actively updated by the management team and adapted to alternations provided by stakeholders. All these stages form a chain of strict procedures that must be followed before a requirements document could be produced.

On the other hand, requirements engineering in Agile context takes a different approach to better suit Agile development principles. According to Elshandidy and Mazan (2013), due to the flexible nature of Agile, conforming to strict and sequential practices of traditional requirements engineering is not feasible. Cao and Ramesh noticed that there are 7 practices in Agile requirements engineering (2008). The first one is face-to-face communication, which brings clients closer to developers through cutting down formality in favor of simple documentation and vague statements, enabling versatility and quick delivery. The second practice is iterative requirements engineering, in which requirements are discovered and defined in conjunction with the development process, creating a better business relationship and clarifying descriptions for developers. In Agile requirements engineering, extreme prioritization ensures the highest quality product as prioritization before each development cycle occurs, not just once like in the traditional model. The next practice is constant planning, which conforms to changes by avoiding fixed plans and dramatically reduces the cost of implementing a new feature. One important characteristic that is also seen in the traditional model is prototyping – releasing a sufficiently functional product instead of a concrete final software – which helps customers to gain insight into development progress and developers to refine requirements. Another practice of Agile requirements engineering is test-driven development, in which tests take precedence over codes. These tests usually fail when the first lines of code are written but will succeed when the last execution takes place. Finally, there are review meetings, where developers, managers, and stakeholders participate; completed features are presented and feedbacks are collected. These 7 practices may seem to be disconnected from traditional requirements engineering; however, they still fall under the same categories, which are summarized in this table by Elshandidy and Mazan (2013):



According to Cao and Ramesh, Agile requirements engineering inherits the iterative approach to development from mainstream Agile development, hence the difference between it and traditional requirements engineering. Agile does not stick to a formal process to create comprehensive requirements documentation, enabling flexibility and adaptivity. Agile requirements engineering is not centralized in advance but distributed evenly throughout the development process (2008).

In conclusion, requirements engineering is necessary for software development. In the software industry, Agile requirements engineering has many different principles as well as some similarities with traditional methods. Choosing the right model ensures efficient software development.

# References

CAO, L., RAMESH, B., 2008. Agile Requirements Engineering Practices: An Empirical Study. *IEEE Software* [online]. **25**(1), pp. 60-67. [viewed 9 October 2021]. DOI: 10.1109/MS.2008.1

ELSHANDIDY, H., MOZAN, S., 2013. Agile and Traditional Requirements Engineering: A Survey. *International Journal of Scientific and Engineering Research*. **4**(9), pp. 473-482. DOI: 10.14299/ijser.2013.09.002

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