**Requirements Engineering & Architecture:**

**A Sole Stakeholder Perspective**

Samantha M. Hipple

School of Technology and Computing, City University of Seattle

CS 504: Software Engineering

Kendra Schraml

May 8, 2022

**Requirements Engineering & Architecture:**

**A Sole Stakeholder Perspective**

Although requirements engineering is not unique to the software or tech industry, software engineers over the years have greatly contributed to the overall improvement of the field. Traditional product development processes, often based on a waterfall methodology, expect the system to go from requirements to maintenance in one, extremely well-planned and well-documented, run. These methods require significant amounts of time and effort creating specific deliverables and documentation that have been thoroughly reviewed for consistency and correctness prior to even starting any development efforts ([source](https://www.kpipartners.com/blog/traditional-vs-agile-software-development-methodologies#:~:text=The%20main%20difference%20between%20traditional,in%20Agile%2C%20it%20is%20iterative.)).

This traditional approach is great for projects with requirements that are easily understood and predefined (e.g., a wood-working project with a clear endpoint). However, in an industry like IT, it is rare that a predetermined solution can be implemented successfully after zero opportunities for modifications even when provided with a predefined, detailed problem statement based on the results of an extremely thorough requirements elicitation process. The rigid standards of a traditional development model, can create a degree of difficulty that is higher than necessary for most projects – especially within the ever-evolving IT industry ([source](https://www.kpipartners.com/blog/traditional-vs-agile-software-development-methodologies#:~:text=The%20main%20difference%20between%20traditional,in%20Agile%2C%20it%20is%20iterative.)).

So, in February of 2001, 17 software developers came together to discuss more light-weight approaches towards product development. From this meeting came the Agile manifesto for software development, a method that later became adopted by any number of industries worldwide due to its many successes within IT. A common quote that is used to define the Agile manifesto follows:

We are uncovering better ways of developing software by doing it and helping others do it. Through this work, we have come to value −

* Individuals and interactions over Processes and tools
* Working software over Comprehensive documentation
* Customer collaboration over Contract negotiation
* Responding to change over Following a plan

That is, while there is value in the items on the right, we value the items on the left more ([source](https://www.tutorialspoint.com/agile/agile_manifesto.htm)).

Unlike the traditional approach, Agile processes make the customers their highest priority through early and continuous software releases as new functional requirements are met. Additionally, change is welcomed throughout the development process – often based on customer and developer feedback gathered after each release. Agile focuses heavily on development teams self-organizing, working together, face-to-face, both with the customers and with each other, including monitoring nonfunctional requirements such as technical excellence and good design to enhance agility. Lastly, tracking progress should be simplified by making the primary measure be the production of working software while a constant pace of development is maintained with regular review intervals that allow the team to reflect and adjust as needed ([source](https://www.tutorialspoint.com/agile/agile_manifesto.htm)).

In order to fully grasp requirements engineering and its relationship with a system’s architectural design, this paper will be referring to a DBG Discord bot example project throughout that has been developed by a single stakeholder - that is, a team size of one. In general, standardized development processes focus on teams and collaboration and ensuring that everyone involved in the project understands the overall system requirements, what progress has already been made, and what requirements still need to be met. Hobby-like projects often neglect many formal processes within software engineering. However, manipulating the Agile methodology for a solo project would allow a single developer to more readily share their project with others in the future by providing additional, valuable information regarding its overall design, purpose, and progress between releases.