**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 woodworking 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 modification even when provided with a predefined, detailed problem statement based on the results of an extremely thorough requirements engineering process. The rigid standards of the traditional 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. The following is a common quote used to self-define the Agile manifesto:

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 incremental 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 explore the requirements engineering process and its relationships within the architectural design of a system, we will be referring to a discord bot project throughout this report that has been developed by a single stakeholder - that is, a team size of one. Generally, standardized development processes focus on teams and collaboration and ensuring that everyone involved in the project understands all system requirements, what progress has been made, and what requirements still need to be met. Solo projects often neglect many formal processes within software engineering; requirement management should not be one of them. Manipulating the Agile methodology for use in tracking the development of a solo project would enable a single developer to share their project more readily with others in the future by providing valuable information regarding its overall design, purpose, and progression between product releases/versions.

**Requirements Management Systems**

There are many requirements management tools available; it can be difficult to choose one for your project. Two of the most popular ones are Jama Software and Modern Requirements. Unfortunately, they are not easy to access for trial. Many of these software/SaaS companies require the user to engage in a 1-on-1 consultation regarding how they intend to use the management tool prior to being approved for the free trial. Simply wanting a tool to assist with understanding the requirements engineering process, the bot project signed up for the 30-day free trial with Smartsheet that is displayed in Figure 1 on the next page.

In addition to being a web application, Smartsheet is also a full featured mobile app. Smartsheet offers a large variety of templates for many different project needs. requirements gathering as well as a template bundle for an Agile backlog and Sprint planning. Smartsheet enables users to create separate workspaces for separate projects that can be filled with customizable sheets, reports, and dashboards of dynamically displayed information via automation using either custom or template workflows and conditional formatting ([source](https://app.smartsheet.com/b/publish?EQBCT=dc9947a9e8f34d1ca6ad207b18505e59)).

**Figure 1:**

*Smartsheet Offers an Instant, Free, 30-day, Trial of Their SaaS*

Graphical user interface, website

Description automatically generated

**Requirements Engineering**

Requirements engineering is defined as “the process of defining, documenting, and maintaining the requirements” ([source](https://www.geeksforgeeks.org/software-engineering-requirements-engineering-process/)) of a system; i.e., a mechanism used to gather, clarify, analyze, assess, negotiate, and specify solutions and then validating said specifications and managing the requirements as they are transformed into a functioning system. The requirements engineering process consists of four main activities: (1) requirements elicitation and analysis, (2) requirements specification, (3) requirements verification and validation, and (4) requirements management ([source](https://www.javatpoint.com/software-engineering-requirement-engineering)). The following sections define each of these activities and provide examples from our bot project when applicable. Figure 2 displays a dynamically created requirements report for the bot project, in card view, organized by each requirement’s current approval status.

**Figure 2:**

*Discord Bot Project Requirements Report by Approval Status*

Graphical user interface, application, Word

Description automatically generated

**Requirements Elicitation and Analysis:**

Also known as the gathering of requirements, elicitation is the process of identifying the domain knowledge and project requirements necessary to develop a product that successfully meets its stakeholders needs. Various sources to gather domain knowledge include the customer, business manuals, existing software of the same type, project stakeholders and more. Methods for the requirements elicitation process include brainstorming, interviewing, document analysis, prototyping, and more ([source](https://www.geeksforgeeks.org/software-engineering-requirements-engineering-process/)). Our bot project used Smartsheet to share the initial brainstorming notes and a document analysis that describes the freeCodeCamp YouTube tutorial on which the project is based – both of which heavily influenced the next step. Figure 3 is a screenshot of the bot project’s document analysis proof within our Smartsheet workspace.

**Figure 3:**

*Demonstration of Smartsheet Proof, Comments, and Review Features*

Graphical user interface, text, application

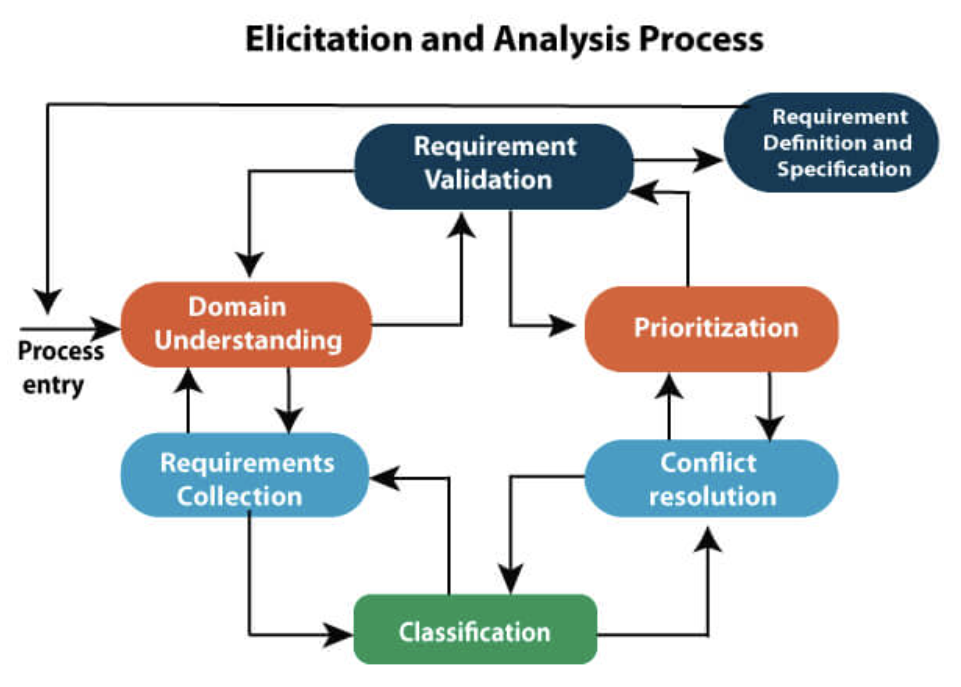
Description automatically generated

Common problems encountered during elicitation include involving all, and only, the right people; stakeholders being unsure of what exactly they want while still expressing requirements in their own terms for the system – some of which may be in conflict with other requirements; requirements changing during the analysis process; and outside influence on system requirements from organizational or political factors.

The elicitation and analysis process are not designed to produce formal requirement models, instead they are designed to help expand the domain knowledge of the analyst in order to prepare for the next stage, requirements specification. Figure 4 is a diagram that illustrates the back-and-forth flow of the requirements elicitation and analysis process in requirements engineering ([source](https://www.javatpoint.com/software-engineering-requirement-engineering)).

**Figure 4:**

*Diagram of Elicitation and Analysis Process in Requirements Engineering (*[*source*](https://www.javatpoint.com/software-engineering-requirement-engineering)*)*

****

**Requirements Specification:**

The requirements specification process is designed to produce formal models for both the functional and nonfunctional requirements discovered during the previous phase. This stage can also produce artifacts such as data flow diagrams, entity-relationship (E-R) diagrams, and data dictionaries, that are designed to illustrate the flow of data through the system that is being developed. The data dictionaries should define the data items included in the data flow diagrams to encourage customers and developers to be using the same terminologies whenever discussing requirements ([source](https://www.javatpoint.com/software-engineering-requirement-engineering)).

Our bot project used a Smartsheet template called *Requirements Collection Checklist* to help guide its requirement engineering process. This template offered nine separate categories under Requirements Analysis header by breaking up nonfunctional requirements into eight independent subcategories (timing, business, technical, UI, performance, operational, expectations & boundaries, and future/deferred requirements). Our project is not very complex, and the experience of our developer is not very advanced, so these categories were taken apart and reworked into a simple drop-down list under a nonfunctional requirements header column that can be seen in Figure 5 below.

**Figure 5:**

*Requirements Analysis Template Customized for the Discord Bot Project*

Table

Description automatically generated with low confidence

**Requirements Verification and Validation:**

Text

**Requirements Management:**

Text