**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 12, 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 significantly contributed to the overall advancement 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 starting any development efforts (Mavuru, 2018).

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 (Mavuru, 2018).

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* (2022):

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.

Unlike the traditional approach, Agile processes make customers their highest priority through early and continuous 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 in self-organizing teams that working together, face-to-face, both with the customers and with each other - including the monitoring of nonfunctional requirements such as technical excellence and an agile design. Tracking progress of an Agile project should be simplified by making the primary measure be the production of working software. A constant pace of development is maintained with regular review intervals that allow the team to reflect and adjust as needed (tutorialspoint, 2022).

In order to fully explore the requirements engineering process and its relationships within the architecture of a system, we will be referring to an example discord bot project throughout this report that has been developed by a single stakeholder - that is, a team size of one. Standardized development processes tend to focus heavily on teams and collaboration - 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; requirements 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, our 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 types including a requirements gathering checklist and a template bundle for an Agile backlog and Sprint planning. Smartsheet enables users to create separate workspaces for various projects that can be filled with customizable sheets, reports, and dashboards of dynamically displayed information via specified automation using either custom or template workflows and conditional formatting (smartsheet, 2021).

**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” (GeeksforGeeks, 2020) of a system, i.e., a mechanism used to gather, clarify, analyze, assess, negotiate, and specify solutions before 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 (JavaTpoint, 2021). The following sections define each of these activities and provide examples from our bot project when applicable. Figure 2 displays a dynamic requirements report for our 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 (GeeksforGeeks, 2020). 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*

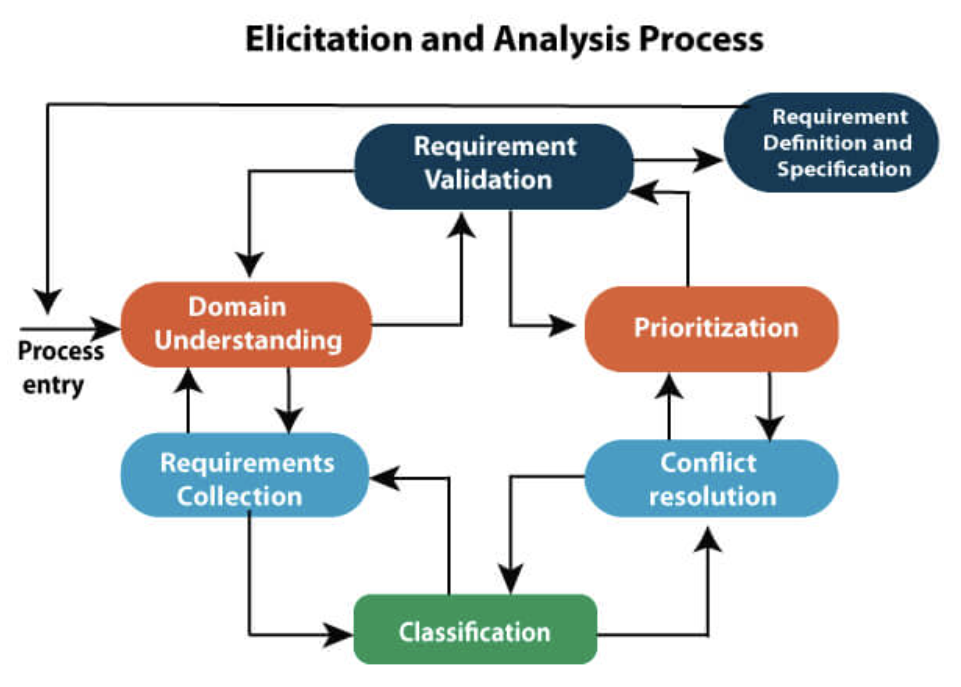
Graphical user interface, text, application

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 (JavaTpoint, 2021).

**Figure 4:**

*Diagram of Requirements Elicitation and Analysis Process (JavaTpoint, 2021)*

****

**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, which 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 (JavaTpoint, 2021).

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:**

Requirements verification and validation is the process of ensuring that the product in development meets its specifications and requirements so that it can fulfill its intended purpose. Verification comes first by using tests to ensure the program is being built according to the specifications formalized in the previous step. The goal of verification is quality assurance. Activities such as reviewing code, documents, design, and the program as well as completing walkthroughs and inspections are all a part of the verification process. Discovering any missing or invalid requirements during this phase can reduce overall rework and cost, as fixing a bug during development is significantly easier and cheaper than creating patches post-production (Jama Software, 2022).

Validation testing confirms the accuracy of the requirements – ensuring all relevant stakeholder requirements have been met, and that the requirements are clearly understood by the developers. Validation is concerned with meeting customer expectations, regardless of how the expectation was met. Tasks in this process may include unit testing, integration testing, and user testing. Software validation should ensure that each verified requirement (1) correctly outlines end user needs, (2) has only one meaning, (3) can be modified, if necessary, (4) documents the properties customers truly need, and (5) is easily linked to system designs, codes, and tests (Jama Software, 2022). During the development of our discord bot project, verification tests were completed as the functional requirements (FR) were implemented and reviewed. Figure 6 shows our bot successfully logging on as a discord user – verifying (and validating) the implementation of our first functional requirement.

**Figure 6:**

*FR 1 – bot must be able to log into discord*

Text

Description automatically generated

The verification test results for functional requirements 2, 3, and 4 are displayed in Figures 7 and 8 as well.

**Figure 7:**

*FR 2 – bot must be able to accept invite into discord server*

*Graphical user interface, text, application, chat or text message

Description automatically generated*

**Figure 8:**

*FR 3 & 4 – bot must be able to read/respond to messages sent by server members*

***Graphical user interface, application

Description automatically generated***

The verification test result displayed in Figure 8 covers two functional requirements – that the bot can read messages sent by server members and that the bot can post messages in response to server members. Additionally, in the spirit of providing an example of validation versus verification – although our bot at this point in time has proven itself able to read and respond to messages – the keywords it picks up on and the responses it can send would not necessarily meet validation standards at this point as the end user would not find this bot particularly fun or useful thus far, which is the end goal of this project.

**Requirements Management:**

As business needs change and a better understanding of the overall system is developed, new requirements can emerge, requirement priorities can shift, and the business and technical environment of the system can change. Requirements management focuses on any changing requirements that occur during the requirements engineering process and system development (JavaTpoint, 2021). Common activities during this process include analyzing, documenting, tracking, prioritizing, and agreeing upon any requirement modifications - as well as controlling any communication with relevant stakeholders. The Software Requirements Specifications (SRS) should be designed to be modifiable at any stage of production, including after receiving end user feedback (GeeksforGeeks, 2020).

According to IBM (n.d.), a typical requirements management process for software engineering includes the following steps:

* Initial requirements gathering from stakeholders
* Requirement analysis
* Defining and recording requirements
* Requirement prioritization
* Requirement agreement and approval
* Requirement tracing to work items
* Stakeholder feedback after implementation for any potential modifications
* Requirement verification and validation
* Impact assessment of any requirement modifications
* Requirement revisal
* Update any impacted documentation

The benefits of following this general process for tracking requirements and their evolutions include a lower overall cost of development, fewer defects, faster delivery, reusability, traceability, and clearly defined connections between requirements and their test cases (IBM, n.d.).

Although our discord bot project is still in its early stages of development, our sole stakeholder has created a document that would prove useful during the requirements management stage – a use case diagram. Use case diagrams are developed during the requirements analysis stage, generally, in order to describe the high-level functions and the overall scope of a system through the depiction of expected user interactions. They can be used to specify the context of a system, capture system requirements, validate the system architecture, drive implementation, and generate test cases (Visual Paradigm, 2022). Within the use case diagram for our discord bot, ideally, the two actors (member and admin) would not have all of the same access permissions to call upon the different bot functions defined within the diagram. This is not currently the case, however, as seen in Figure 9 below.

**Figure 9:**

*Use case diagram for DBG discord bot project*

*Diagram

Description automatically generated*

**Conclusion**

Developing a software product solo is not an easy feat as there is no one to provide feedback or guidance on the system making error catching and design flaws more likely to slip through the cracks. Due to this, the best development methodology to follow as a solo developer is an Agile methodology – one that relies on iterations and observations to release a product early and continuously as improvements or modifications are made. Additionally, performance monitoring is a critical part of the feedback loop for solo developers. Creating capture logs and metrics, as well as logging any unexpected errors, will go a long way in helping troubleshoot any deployment failures or thrown exceptions during release. Although it may be tempting to skip many formal practices of software engineering while a solo developer, taking the time and effort to produce proper documentation throughout a project is an effective way to both advance their software engineering skills and display their overall knowledge of the engineering process to others (Shalvah, 2019).

Good documentation ensures that the knowledge gained and used throughout the development process can be readily passed on to others or reviewed in the future without requiring the human developer to remember every detail of the production. Additionally, documentation is where high-level views of a project are determined, which can help the developer maintain focus on the end goals. Documentation does not have to be complex and should be done before, during, and after coding. After coding, it is also important to double-check that any documents created prior to this stage are consistent with any changes that may have occurred during. If documenting a specific work item appears too complex, this can be a sign for the developer to break down that item into smaller units as well (Shalvah, 2019).

When working alone, developers learn less, do not have a team to rely on for motivation or moral, and could potentially allow obscure and undocumented shortcuts to propagate. However, attempting to follow proven software engineering practices when you are the sole stakeholder of a project, will improve your ability to work with teams in the future, improve your overall abilities as a developer, and provide you with a proper workflow that focuses on tracking development issues, reviewing your code regularly, and maintaining proper version and change management control.

**References**

GeeksforGeeks. *Requirements Engineering Process*. Software Engineering. (2020, February 27). Retrieved May 12, 2022, from https://www.geeksforgeeks.org/ software-engineering-requirements-engineering-process/

IBM. (n.d.). What is Requirements Management? IBM. Retrieved May 12, 2022, from https://www.ibm.com/topics/what-is-requirements-management

Jama Software. (2022, March 29). *Requirements Verification and Validation for Product Teams*. The Essential Guide to Requirements Management and Traceability. Retrieved May 12, 2022, from https://www.jamasoftware.com/ requirements-management-guide/requirements-validation-and-verification/requirements-verification-and-validation-for-product-teams

JavaTpoint. (2021). *Requirement Engineering*. Software Engineering. Retrieved May 12, 2022, from https://www.javatpoint.com/software-engineering-requirement-engineering

Mavuru, I. (2018, June 22). *Traditional vs. Agile Software Development Methodologies.* Retrieved May 12, 2022, from https://www.kpipartners.com/blog/traditional-vs-agile-software-development-methodologies#:~:text=The%20main%20difference%20 between%20traditional,in%20Agile%2C%20it%20is%20iterative.

Shalvah. (2019, April 1). *Good Engineering Practices while Working Solo*. Bits and Pieces. Retrieved May 12, 2022, from https://blog.bitsrc.io/good-engineering-practices-while-working-solo-ad872e727af4

smartsheet. (2021, April 6). *Agile Backlog and Sprint Planning*. Setup Dashboard. Retrieved May 12, 2022, from https://app.smartsheet.com/b/publish?EQBC T=dc9947a9e8f34d1ca6ad207b18505e59

tutorialspoint. (2022). *Agile - Manifesto*. Learn Agile. Retrieved May 12, 2022, from https://www.tutorialspoint.com/agile/agile\_manifesto.htm

Visual Paradigm. (2022). *What is Use Case Diagram?* Support. Retrieved May 12, 2022, from https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/