American Academic Scientific Research Journal for Engineering, Technology, and Sciences
ISSN (Print) 2313-4410, ISSN (Online) 2313-4402
https://asrjetsjournal.org/index.php/American_Scientific_Journal/index

Bridging the Gap Between Stakeholder and Software Products: A Review of Software Requirement Engineering Techniques

Nzechukwu Chizaram Otuneme[a*], Monday Eze[b], Shade Kuyoro[c], SettingsFolasaded Ayankoya[d].

[a]*Email: nzegod4u@gmail.com,* [b]*Email: ezem@babcock.edu.ng,* [c]*Email: kuyoros@babcock.edu.ng,* [d]*Email:ayankoyaf@babcock.edu.ng*

Abstract

Effective software requirement engineering plays a crucial role in bridging the gap between stakeholders and software products. The success of any software project heavily relies on accurately capturing, analysing, and documenting stakeholders' needs and expectations. This article provides a comprehensive review of various software requirement engineering techniques that facilitate the alignment of stakeholder requirements with software product development. Software requirements are extracted from a variety of stakeholders, but the decision of "what to develop" is a difficult one. Stakeholders' lack of clarity about what they want makes requirement elicitation a difficult and vital task.

It explores the significance of understanding stakeholders' perspectives, discusses popular requirement engineering approaches, and highlights their strengths and limitations. The article concludes by emphasizing the importance of selecting appropriate requirement engineering techniques based on the project's context and offers recommendations for future research in this domain.

Keywords: Software; Requirement Engineering; Software Products; Stakeholders.

1. Introduction

In the rapidly evolving software industry, understanding and effectively addressing stakeholder requirements are essential for successful software product development. Software requirement engineering serves as a critical bridge between stakeholders and software development teams.

Received: 4/25/2023
Accepted: 5/14/2023
Published: 5/30/2023

---

* Corresponding author.

It involves a systematic process of capturing, documenting, analyzing, validating, and managing stakeholders' needs and expectations to ensure the development of software products that meet their requirements. Software Requirement engineering (RE) is a transdisciplinary function that bridges the gap between the system stakeholders and developers to define and maintain the requirements that the software of interest must meet, it focuses on the identification, eliciting, building, analysing, verifying (including verification methods and strategy), validation, communication, recording, and management of requirement [1]. The most significant artifacts that describe the characteristics and behaviours of software applications are software requirements specifications [2].

These consists of Functional Requirements (FR) and Non-Functional Requirements (NFR) they outline the intended functionality required by users. Functional requirements define the software, stating the expected tacks that the software can perform, which are mostly listed by stakeholders. Non-functionals are the expected but largely unlisted requirements to accomplish functional tasks, and they also serve as a measuring standard for functional requirements. Distinguishing non-functional requirements from functional requirements is one of the most difficult jobs in dealing with them in requirement[3]. Robertson & Robertson in a research in 2013 distinguished between functional and non-functional requirement as the task that should be performed by the system and how the system shall perform a task respectively. It is overly simplistic, somewhat misleading, and unhelpful to label every requirement that cannot be called functional as a non-functional requirement. According to [5] terming every requirement that is vague, ambiguous, and poorly defined as "non-functional" is incorrect.

Institute of Electrical and Electronics Engineers (IEEE) 2430-2019 Trial-Use Standard for Software Non-Functional Sizing Measurements describes Non-functional Requirements as “Any requirement for a software-intensive system or for a software product, including how it should be developed and maintained, and how it should perform in operation, except any functional user requirement for the software. Non-functional requirements (NFRs) concern: the software system or software product quality, the environment in which the software system or software product must be implemented and which it must serve, and the processes and technology to be used to develop and maintain the software system or software product and the technology to be used for their execution” [6,7]. Also, Common Software Measurement International Consortium/ International Function Point Users Group (COSMIC/IFPUG) Glossary of NFR and Project terms defines NFRs as “Any requirement for a software system or for a software product, including how it should be developed and maintained, and how it should perform in operation, except any functional user requirement for the software [8,9].

2. Understanding Stakeholders

Software development projects rely heavily on software stakeholders for their success. Stakeholders represent individuals or entities with a vested interest in the software product. They may include end-users, clients, project managers, developers, and quality assurance personnel. Understanding the diverse perspectives, goals, and expectations of stakeholders is a fundamental step in requirement engineering. These stakeholders play a crucial role by shaping the project's strategic direction, providing financial support, and possessing specialized knowledge. These contributions are essential for the ongoing development and maintenance of the software[10].

As software environments become more complex and the demand for seamless integration between different products grows, there is a greater need for collaboration and coordination among individuals and organizations involved in software ecosystems (SECOs). This collective effort is necessary to ensure the production of reliable and efficient integrated software [11]. When it comes to requirements engineering, stakeholders differ from project to project. Users and acquirers are the bare minimum of stakeholders (who may not be the same). Complex projects might have a large number of users and acquirers, each with its own set of problems. As part of the basic set of stakeholders, two additional categories may be necessary. To begin with, the company or companies that are developing, maintaining, or managing the system or software have a legitimate interest in earning from it. Second, regulatory authorities may be bound by statutory, industry, or other external constraints, necessitating a thorough review. The requirements are generated from a variety of stakeholders, but the decision of "what to create" is a challenging one. Stakeholders' lack of clarity about what they want makes requirement elicitation a difficult and vital task. Elicitation, analysis, specification, validation, negotiation, and requirements management are all part of the RE phase of software development [12]. There are some processes to developing software, and developers must have a thorough understanding of the software's requirements. According to [13] one of the most critical parts of software development is requirements engineering. However, developers often construct software without having a thorough understanding of the requirements because they believe that doing so would be a waste of time in an environment where requirements are continuously changing [14,15].

Requirement engineering is an important aspect of software engineering that was brought to bear due to the number and diverse nature of stakeholders involved in software development[16]. Functional requirements and non-functional requirements are the two basic categories of software requirements [17]. The merit and experience of the Requirement Engineers influence the quality of the requirements gathered, and many times, due to variables like inexperience or a lack of domain familiarity, an exact and complete set of needs is not gathered. Manual requirement elicitation is time-consuming and error-prone, especially when there are a significant number of requirements [18,19].

In other, for a software development team to have a quality output in terms of the final product both the functional and non-functional requirements must be thoroughly addressed [20]. Requirement engineering deals with problems to be done and not how they will be done. Requirement engineering enables stakeholders to know the build-up of a project before the actual implementation to avoid rework of the project. The goal of requirement engineering is based on some assumptions.

1. The longer a mistake is discovered, the more costly it is to repair it.
2. Prior to system design, a stable set of requirements can be determined. implementation starts.

3. Transformation of needs into requirements

Stakeholder needs (or goals, or objectives) are improved and evolved into genuine stakeholder requirements as

the process of defining requirements progresses. Initial stakeholder concerns are rarely used as stakeholder needs because they lack definition, analysis, and, in certain cases, consistency and feasibility. Requirements engineering guides stakeholders from those initial, often latent needs to a set of objectively adequate, structured, and more formal statements of stakeholder requirements and goals, using the Concept of Operations to aid understanding of stakeholder concerns at the organizational level and the System Operational Concept from a system perspective.

When the system of interest is a system element at the second or lower level of the overall system structure, stakeholder needs and requirements must be identified at that level as well. These requirements are not limited to the highest system in the system hierarchy. Additional stakeholders may emerge only after lower-level architecture or design decisions have been made. The stakeholder-owned system requirements are then translated into lower-level system element requirements for the target system. Where there is a high level of risk owing to technology or complexity, requirements are truly changing due to a change in demand, or requirements were never adequately established in the first place, this transition may take a lot of iteration. The same methods are applied recursively to lower physical level system elements, which are similarly subject to design and development, resulting in the generation of lower-level system element requirements.

4. Requirements construct

The development of well-defined stakeholder requirements, system requirements, and system element requirements is required. This approach aids in the validation of requirements with stakeholders and ensures that the requirements appropriately reflect stakeholder needs. One or more of the following elements can be found in a well-formed defined requirement:

i it must be met or held by a system in order for it to solve a problem, achieve an aim, or address a stakeholder issue;

ii it is limited by constraints;

iii it must be met or held by a system in order for it to solve a problem, achieve an aim, or address a stakeholder issue;

iv it specifies the system's performance when utilized by a certain stakeholder or the system's associated capacity, but not the user's, operators, or other stakeholder's capability; and

v it is verifiable (e.g., the realization of the requirement in the system can be demonstrated).

A demand at lower levels aligns with the design of the higher physical-level system, which is another factor to consider. This description establishes a distinction between needs and the attributes associated with those requirements (conditions, assumptions and constraints). A requirement is a statement that interprets or expresses a need, as well as the limits and criteria that go along with it. A demand might be written in plain language or in any other linguistic format as exemplified in Figure 2.1. If the statement is written in natural language, it should have a subject and a verb, as well as any other elements necessary to appropriately describe the requirement's information content. A requirement must define the requirement's subject (e.g., the system, software, etc.), what must be done (e.g., run at a given power level, supply a field for), or a system restriction.

5. Requirement Elicitation Techniques

Software requirements are extracted from a variety of stakeholders, but the decision of "what to develop" is a difficult one. Stakeholders' lack of clarity about what they want makes requirement elicitation a difficult and vital task[21]. Most requirements elicitation (RE) approaches or tools are designed to minimize ambiguity in requirements and improve clarity. Only a few RE approaches, on their own, can support accuracy, consistency, and completeness of requirements, as well as address ambiguity[22]. By properly following requirement elicitation and gathering techniques, high quality mobile applications can be produced.

a. Interviews

Interviewing is a technique for gathering information and opinions from future users and other stakeholders of a system in development. It is possible to identify and correct errors and misunderstandings. There are two types of interviews: formal and informal.

i. the closed interview, in which the requirements engineer asks a set of questions and waits for responses.

ii. the open interview, in which the requirements engineer and stakeholders discuss what they want from a system in an open-ended manner without any pre-defined questions.

There is no clear distinction between the two types of interviews. You begin with a few questions, which are then discussed and followed by fresh ones. Interviews have the advantage of assisting the developer in gathering a large amount of data. Their drawback is that analysing such a large volume of qualitative data can be difficult, and different stakeholders may present contradictory information.

b. Use cases / Scenarios

Use cases illustrate interactions between users and systems, with a focus on what users must accomplish with the system or tasks they can perform on the system. A use case is a sequence of interactions that a system can conduct with an external actor (e.g., a person, a piece of hardware, or another software product), including variants and extensions. Use cases indicate the software system's functional needs and can be employed in the early phases of development. To validate a given use case, analysts and customers should study it. Scenarios are interactive sessions that model a specific sort of a user's contact with a system Scenarios should include a description of the system's state before and after the scenario, as well as what activities can be carried out concurrently, the typical flow of events, and exceptions to the events.

c. Observation and social analysis

Observational approaches comprise an investigator studying users while they work and noting any unusual behaviour. Direct observation, with the investigator present during the task, or indirect observation, with the task observed through another means, is possible (e.g., recorded video). It is useful for researching current tasks and procedures. Observation allows the observer to see what users do in context, which helps to overcome

challenges with stakeholders who describe idealized or oversimplified work processes. This type of method is time consuming as the observer will not be able to carry out this task when the much stakeholders are involved in this kind of requirement gathering. It can also lead to error because what the observer is perceiving might not be the correct experience of the stakeholders.

d. Focus Groups

The majority of existing strategies are ineffective in reaching a big number of end-users. Interviewing all end users of a system where the system involves thousands or millions, would be nearly impossible [23]. It's more reasonable to focus on end-user representatives, as in a focus group. As a result, it's probable that the system won't be able to satisfy all of its users' requirements. Focus groups are an informal technique in which a group of four to nine users from various backgrounds and skill levels discuss issues and concerns regarding aspects of a system prototype in a free-form manner. Focus groups are useful for determining user needs and perceptions, as well as what matters to them and what they expect from the system. They frequently elicit unexpected reactions and ideas. Because there is typically a significant gap between what people say and what they do, observations should be used in conjunction with focus groups. Focus groups may help people articulate their ambitions, design ideas, and product concepts. They also assist users in identifying items that should be updated and in developing a ‘shared meaning' for the system.

e. Brainstorming

This aids in the development of innovative solutions to specific issues. The generation phase, in which ideas are collected, and the assessment phase, in which the collected ideas are debated, are the two phases of brainstorming. Ideas should not be judged or evaluated during the generation phase. The concepts should be produced quickly, and they should be broad and unusual. Brainstorming leads to a greater knowledge of the problem and a sense of shared responsibility of the outcome.

f. Prototyping

A system prototype is a preliminary or initial version of the software system that is available and assessable during the first stages of the software development process. Software prototypes are frequently used to enhance software requirement elicitation and validation. Software Prototypes are categorized into two major and distinct areas: (i) Throwaway prototypes: this type of software prototype help both developers and other stakeholder comprehend and understand complex software requirements. (ii) Evolutionary prototypes: this type of software prototype presents a working system to the stake holders and are frequently incorporated into the final product. Paper prototypes (where a mock-up of the system is created on paper), "Wizard of Oz" prototypes (where a human replicates the system's replies in response to particular user inputs), and automated prototypes are all examples of prototypes (where a rapid development environment is used to develop an executable prototype).

g. Crowd-based requirements engineering (CrowdRE)

Crowd-based requirements engineering (CrowdRE) refers to automated or semi-automatic ways of obtaining

and analysing data from a crowd in order to develop validated user requirements [24,25]. In traditional RE methods, a small number of people are interviewed or gathered in focus groups. Advanced RE approaches used in market-driven RE allow businesses to communicate directly with key stakeholders through ad hoc feedback channels [26].

Crowd requirement engineering is a software elicitation strategy in which the crowd is used to cater the requirements to provide the desired solution. Because of its potential to solve complex real-world problems, crowd requirement engineering is an emerging field of software engineering that has piqued the interest of most researchers[27]. It also places a heavy emphasis on a participatory approach, in which fundamentally motivated users become crowd members as a result of applications that meet their needs. Members of the crowd report on a range of topics which range between functional and non-functional requirements such as bus and new product ideas. RE necessitates this differentiation, which some research prototypes readily provide, even though many input methods do not explicitly differentiate between these aspects. Some feedback channels go beyond simply eliciting feedback; for example, social media platforms like Facebook can be used to collect, prioritize, and negotiate feedback[28].

Trying to gather requirements from the crowd through social media highlights many promises has more people can be reached with ease. However, from the point of view of requirement engineering, this strategy can result in new issues. Popular social networking sites were not created with RE in mind, and as a result, they do not have sufficient support for RE activities. Other approaches which could be adopted in CrowdRE presented a crowdsourcing-based approach for a German medium-sized company – myERP to help get requirements from non-German customers [29]. The technique proposed was used to extract ERP software requirements. The presented method has a major flaw in that it does not allow for the elimination of redundant requirements or the ranking of certain requirements. requirements.

6. Requirements Management Tools

Requirements management tools are the perfect sidekick for your requirements gathering process. Effective requirement management is essential for maintaining the alignment between stakeholders and software development teams throughout the project lifecycle. This section discusses techniques for requirement documentation, version control, change management, and prioritization. It also explores the use of requirements management tools to streamline the process and enhance collaboration. There are so many requirement management tool but the following are listed as to tools to be considered by practitioners.

i. Jama Software: Managing requirements in Jama Connect speeds the product development process by saving time, strengthening alignment, and ensuring quality and compliance. Teams can create, review, validate, and verify requirements in one solution. The key features are: Live Traceability, Decision Tracking & Faster Reviews and Real Time Collaboration.
ii. ReqSuite® RM: ReqSuite® RM from OSSENO Software is a very powerful solution for managing requirements, test cases, and other conceptual artifacts along the development cycle. The software provides a rich toolkit with powerful customization options as well as the ability to collaboratively manage, track, analyze, approve, review, export, import and reuse requirements. Additionally, its ease

of use and simple set-up, as well as free premium support means you can get a customized solution quickly running.

iii. Visure Requirements: Visure provides an easy to use and customized solution for requirements, risk, and test management. It's a must have tool for teams building complex products, systems and softwares, which require end to end traceability from conception to testing and deployment, along with standard certification compliance. It's one of the most flexible and customizable RM platforms, making it easy for stakeholders to collaborate on simple or complex requirements.

iv. Modern Requirements: Turn your Azure DevOps project into a collaborative workspace and ditch old copy/paste-based manual processes. Create, automate, manage, analyze, and report directly from your project.

v. IBM Engineering Requirements Management DOORS Next: IBM® Engineering Requirements Management DOORS® Next provides a scalable solution to optimize communication and collaboration among teams and stakeholders to maximize productivity and quality. It enables you to capture, trace, analyze and manage changes to requirements while maintaining compliance to regulations and standards. With DOORS Next, you can improve the management of project scope and cost throughout your organization and supply chain. Because this software is available on premises and in the cloud, you can deploy as best fits your environment.

7. Constraints and Limitations of the study

The article is a theoretical review of the subject matter, it does contain any practical illustrations. This article does not recommend any method above others but exposes the leading methods of in requirement elicitation and management to the reader so that informed decisions can be made by the stakeholders.

8. Conclusion and Future Directions

This article provides a comprehensive review of software requirement engineering techniques for bridging the gap between stakeholders and software products. It highlights the significance of understanding stakeholder perspectives and reviews various techniques for requirement elicitation, analysis, validation, and management. The article concludes by emphasizing the need for selecting appropriate techniques based on project context and offers recommendations for future research, including the exploration of AI-based approaches and the integration of agile practices into requirement engineering.

In conclusion, successful software product development relies on effective requirement engineering techniques that facilitate the alignment of stakeholder needs with software development processes. By employing appropriate techniques, software development teams can enhance stakeholder satisfaction, reduce project risks, and deliver high-quality software products. In making choices of techniques the stakeholders should consider the type of system being built, the composition to the team, time, budget and the skill set of the team before making such decision.

This research recommends that further research be carried out to determine the major factors in requirement

engineering tht hinder efficient and effective software products that is accepted to stakeholders.

References

[1] IEEE Standard, "INTERNATIONAL STANDARD ISO / IEC / IEEE Systems and software engineering - Architecture description," *Ieee Standards*, vol. 2018, 2018.

[2] C. W. Ho, M. J. Johnson, L. Williams, and E. M. Maximilien, "On Agile performance requirements specification and testing," in *Proceedings - AGILE Conference, 2006*, 2006, pp. 47–52. doi: 10.1109/AGILE.2006.41.

[3] C.-Y. Pang, "Product Backlog and Requirements Engineering for Enterprise Application Development," in *Software Engineering for Agile Application Development*, 2020. doi: 10.4018/978-1-7998-2531-9.ch001.

[4] S. Robertson and J. Robertson, *Mastering the Requirements Process: Getting Requirements Right*, Third. Addison Wesley, 2013.

[5] M. Broy, "Rethinking Nonfunctional Software Requirements," *Computer*, vol. 48, no. 6, pp. 72–75, 2015, doi: 10.1109/MC.2015.172.

[6] IEEE, "IEEE 610.12 - Standard Glossary of Software Engineering Terminology | EnIEEE. 1990. IEEE 610.12 - Standard Glossary of Software Engineering Terminology | Engineering360.gineering360," 1990. Accessed: May 27, 2021. [Online]. Available: https://standards.globalspec.com/std/398645/IEEE 610.12

[7] IEEE Computer Society, *IEEE Trial-Use Standard for Software*. The Institute of Electrical and Electronics Engineers Standards Assosiation, 2019. doi: 10.1109/IEEESTD.2019.8870263.

[8] COSMIC and IFPUG, "Glossary of terms for Non-Functional Requirements and Project Requirements used in software project performance measurement , benchmarking and estimating," no. September, 2015.

[9] IFPUG, "COSMIC and IFPUG Glossary of terms ," *international Functions Point Users Group (IFPUG)*, May 30, 2019. https://www.ifpug.org/cosmic-and-ifpug-glossary-of-terms/ (accessed Aug. 16, 2021).

[10] K. Power, "Stakeholder Identification in Agile Software Product Development Organizations: A Model for Understanding Who and What Really Counts," in *2010 Agile Conference*, IEEE, Aug. 2010, pp. 87–94. doi: 10.1109/AGILE.2010.17.

[11] S. Lewellen, "A comprehensive approach to identifying key stakeholders in complicated software ecosystems," *Proceedings of the IEEE International Conference on Requirements Engineering*, pp. 492–497, 2021, doi: 10.1109/RE51729.2021.00074.

[12] H. S. Dar, "Reducing Ambiguity in Requirements Elicitation via Gamification," *Proceedings of the IEEE International Conference on Requirements Engineering*, vol. 2020-Augus, pp. 440–444, 2020, doi: 10.1109/RE48521.2020.00065.

[13] S. Alzu'Bi, B. Hawashin, M. Eibes, and M. Al-Ayyoub, "A Novel Recommender System Based on Apriori Algorithm for Requirements Engineering," *2018 5th International Conference on Social Networks Analysis, Management and Security, SNAMS 2018*, pp. 323–327, 2018, doi: 10.1109/SNAMS.2018.8554909.

[14] F. Dwitam and A. Rusli, "User stories collection via interactive chatbot to support requirements gathering," *Telkomnika (Telecommunication Computing Electronics and Control)*, vol. 18, no. 2, pp. 890–898, 2020, doi: 10.12928/TELKOMNIKA.V18I2.14866.

[15] R. Pressman, *Software engineering: a practitioner's approach*. MCGRAW-HILL US HIGHER ED., 2019.

[16] S. L. Buitron, F. J. Pino, B. L. Flores-Rios, J. E. Ibarra-Esquer, and M. A. Astorga-Vargas, "A Model for Enhancing Tacit Knowledge Flow in Non-functional Requirements Elicitation," *Proceedings - 2017 5th International Conference in Software Engineering Research and Innovation, CONISOFT 2017*, vol. 2018-Janua, pp. 25–33, 2018, doi: 10.1109/CONISOFT.2017.00011.

[17] R. K. Gnanasekaran, S. Chakraborty, J. Dehlinger, and L. Deng, "Using recurrent neural networks for classification of natural language-based non-functional requirements," *CEUR Workshop Proceedings*, vol. 2857, 2021.

[18] F. Khan, S. R. Jan, M. Tahir, S. Khan, and F. Ullah, "Survey: Dealing Non-Functional Requirements at Architecture Level," *VFAST Transactions on Software Engineering*, vol. 9, no. 2, p. 7, Apr. 2016, doi: 10.21015/vtse.v9i2.410.

[19] R. K. Surana, Chetan Surana, Shriya, Di. B. Gupta, and S. P. Shankar, "Intelligent Chatbot for Requirements Elicitation and Classification," *2019 4th IEEE International Conference on Recent Trends on Electronics, Information, Communication and Technology, RTEICT 2019 - Proceedings*, pp. 866–870, 2019, doi: 10.1109/RTEICT46194.2019.9016907.

[20] G. A. Alencar, F. V. S. De Oliveira, J. Da Silva Correia-Neto, and M. M. Teixeira, "Non-functional requirements in health information systems: A systematic mapping research," *Iberian Conference on Information Systems and Technologies, CISTI*, vol. 2019-June, no. June, pp. 1–5, 2019, doi: 10.23919/CISTI.2019.8760720.

[21] H. Dar, M. I. Lali, H. Ashraf, M. Ramzan, T. Amjad, and B. Shahzad, "A systematic study on software requirements elicitation techniques and its challenges in mobile application development," *IEEE Access*, vol. 6, pp. 63859–63867, 2018, doi: 10.1109/ACCESS.2018.2874981.

[22] B. B. Chua, D. V. Bernardo, and J. Verner, "Understanding the use of elicitation approaches for effective requirements gathering," *Proceedings - 5th International Conference on Software Engineering Advances, ICSEA 2010*, pp. 325–330, 2010, doi: 10.1109/ICSEA.2010.89.

[23] J. Kontio, L. Lehtola, and J. Bragge, "Using the focus group method in software engineering: Obtaining practitioner and user experiences," in *Proceedings - 2004 International Symposium on Empirical Software Engineering, ISESE 2004*, 2004, pp. 271–280. doi: 10.1109/ISESE.2004.1334914.

[24] E. C. Groen *et al.*, "The Crowd in Requirements Engineering: The Landscape and Challenges," *IEEE Software*, vol. 34, no. 2, pp. 44–52, Mar. 2017, doi: 10.1109/MS.2017.33.

[25] T. Y. Lim, F. F. Chua, and B. B. Tajuddin, "Elicitation techniques for internet of things applications requirements: A systematic review," *ACM International Conference Proceeding Series*, pp. 182–188, 2018, doi: 10.1145/3301326.3301360.

[26] E. C. Groen *et al.*, "The Crowd in Requirements Engineering: The Landscape and Challenges," *IEEE Software*, vol. 34, no. 2, pp. 44–52, 2017, doi: 10.1109/MS.2017.33.

[27] O. Adetunji, E. Oyenuga, and N. Otuneme, "Crowd Requirement Rating Technique (CrowdReRaT) Model for Crowd Sourcing," *Int J Comput Appl*, vol. 176, no. 22, pp. 9–14, 2020, doi: 10.5120/ijca2020920178.

[28] N. Seyff, I. Todoran, K. Caluser, L. Singer, and M. Glinz, "Using popular social network sites to support requirements elicitation, prioritization and negotiation," *Journal of Internet Services and Applications*, vol. 6, no. 1, p. 7, Dec. 2015, doi: 10.1186/s13174-015-0021-9.

[29] P. K. Srivastava and R. Sharma, "Crowdsourcing to elicit requirements for MyERP application," *1st International Workshop on Crowd-Based Requirements Engineering, CrowdRE 2015 - Proceedings*, pp. 31–35, 2015, doi: 10.1109/CrowdRE.2015.7367586.