Informatics Bulletin, Faculty of Computers and Artificial Intelligence, Helwan University
Published Online ***Vol 7 Issue 1, January 2025***
(https://fcihib.journals.ekb.eg)

A Methodology for Visualizing User Requirements

[1,2] A. Ahmed, [1] M. Nasr, and [1] L. Abd Elhamid
[1] Information System Dept., Faculty of Computers and Artificial Intelligence, Helwan University, Egypt
[2] ahmed_awad_1052@fci.helwan.edu.eg

***Abstract*—The process of ensuring that stakeholder or customer requirements are met is known as requirement validation in the software industry. Requirement validation is a critical stage in ensuring that a system's requirements are gathered with high accuracy. Its major goal is to get accurate data from the organization/users and implement it correctly step by step. This concept suggests that the finding's result is correct and that anyone may verify it easily and smoothly. Although there are different requirement validation methodologies accessible in the literature, the majority of models do not specifically incorporate a full validation strategy into the requirement engineering process. Consequently, we have put out a commendable approach for integrating requirement validation with the requirement engineering procedure.**

This paper discusses how to visualize the user requirement validation language, its various characteristics, and how to apply requirement validation visualization in an industrial context to build high-quality software products while saving time.

***Index Terms*—Requirements validation, visualize Requirement engineering, Software development, Design description languages, Graphical notation.**

I. Introduction

SOFTWARE is employed in greater and more diverse ways than ever before. A software engineer follows a specific procedure. Verification and validation ensure that the programs meet their specifications and that the clients receive exactly what they bought. A project might fail or become more expensive than expected, if requirements are not managed appropriately, and the quality of the software generated can suffer. It is critical to detect all problems throughout the development phase of the software development process utilizing requirement validation methodologies, and then to resolve the complete problem step by step. [1]

User requirements explain the services that the user expects from the system, as well as the constraints that must be overcome to achieve them. They also describe how the system fulfills the requirements. It must be written in such a way that a person with no technical background or experience may understand it. [2]

Validation guarantees that the conclusions of the Requirements Analysis are accurate, and hence the specification can be trusted. Additionally, validation facilitates the exchange of information between the system analyst and stakeholders. "The generation of the Specification" is defined by Fuck as "not so much a translation process as an interactive issue solving process." Validation techniques vary with respect to the representation format that is used to deliver the requirements to the users. [1]

Getting the greatest level of end-user satisfaction for the least amount of money and time is the main goal of requirement validation. Requirement validation is important because errors found in a requirements document during development or after the system has been deployed can lead to large rework costs. As a result, the primary goal of requirement validation is to detect and remedy all flaws, rather than to verify that the requirements are correct. Various requirements approaches are available to assist analysts in validating requirements. The following are some of the validation techniques.[1]

By walking through a section of the specification to follow a scenario, the procedure gives an indicator of the dynamic behavior of the system. Put another way, it's a potent technique to show faults, which will help to provide a more accurate characterization of each problem. By communicating user objectives to teams, this requirement validation technique enables them to promptly pinpoint places where errors or misunderstandings occur. [3]

All of the major parts of the requirements model are determined in Animation by providing motivation that results in a reaction. Since animation keeps track of issues related to requirements discovered during validation, asynchronous communication is the most widely used method for transferring documents. However, the lack of interactivity causes less capability to deal with ambiguity or other critical requirements concerns.

We've discovered that existing requirement validation models don't include all the elements of requirement validation methodologies, thus they're not appropriate for major software development projects. To address this, propose a model that visualizes the requirement validation technique during the requirement engineering process. This paradigm can be

effective in businesses that working on major projects with a high number of data sets and databases.[1]

Conditions Validation is easy, cheap, time-consuming, and frequently requires laborious manual tasks that are prone to error. As the application size, domain, and inherited textual requirements constructs increase, the challenge gets harder. Existing methods are categorized as fine-grained with formal requirements, domain-specific, or passive-defect aggregations.

This study introduces a methodology that automates and visualizes the validation process, increasing software engineers' online productivity while meeting customer needs. The applicability of our solution to requirements inconsistency defects is demonstrated in a proof of concept. [11]

II. VISUALIZATION

The importance of information visualization in Big Data analysis cannot be overstated. The amount of data that needs to be interpreted is always increasing. Users, on the other hand, are rarely specialists in information visualization.

As a result, defining the visualization that best suits a given situation is a difficult task for them.

Furthermore, consumers frequently lack a clear understanding of the goals for which the visualizations are being created. As a result, it's possible that graphics will be misconstrued, leading to poor decisions and missed chances.[16]

The lack of methods and resources to help non-expert users of visualizations define their objectives and visualizations is one of the fundamental problems with this process.

This paper aims to facilitate the communication of analytical needs by non-expert users of data visualization, (ii) help them create visualizations that best suit their needs, and (iii) evaluate the effectiveness of our proposal through a case study that details an experiment with 97 non-expert users of data visualization.[16]

III. RELATED WORK

Because the cost of repairing a problem that develops due to a problem in requirements is substantially higher than fixing a design or coding error, requirements validation is critical. Errors in requirements cause the project to fail since it is unable to meet the needs of the consumers. Any changes in requirements at a later stage will necessitate adjustments to the design, architecture, or implementation. [4]

According to [5] the bulk of errors in a software project, are caused by incorrect specifications. As seen in Figure 1, which provides steps for a structured approach to gathering and validating software requirements, essential for successful software development. Almost half of the issues are due to requirements that are incomplete or ambiguous, while the rest are due to omitted requirements. Because there are minimal deliverables to be produced during the requirements engineering phase, the cost of addressing an issue is lower at this stage.

Step 1: Point of origin
Step 2: Get the user's and stakeholders' requirements.
Step 3: The following methods were used to begin the elicitation of requirements:
(A) Examining the records.
(B) Talk with someone.
(C) Survey.
Step 4: Choose one of the aforementioned techniques. If not, proceed to step 1.
Step 5: Incorporate the need into the SRS (software requirements and specifications).
Step 6: Choose the validation methods based on the following criteria:
First, prototyping.
(2) Motion Pictures.
(3) Examination/Review.
(4) Assessment.
(5) Summarizing in natural language.
(6) Techniques for Expert Systems.
If not, proceed to Step 5.
Step 7: To extract the last requirements, repeat steps 2 through 6.
Step 8: Gather the last set of requirements.
Step 9: Integrate it into the process of developing software.
Step 10: End point.

Fig. 1. User validation requirement algorithm [1]

Requirements validation approaches are important to support requirement defect discovery, which can be difficult in later stages of the SDLC. If organizations effectively detect all defects during the requirements phase, they will save money on bug fixes and the project will be delivered on time. If the majority of bugs are discovered and eliminated during the requirements phase, only a small amount of effort will be required during the regression testing phase, resulting in a low cost. Table 1 shows a comparison of the strategies covered in the paper. [4]

Table. 1. Comparison of requirements validation techniques

	Requirements Inspection	Requirements Prototyping	Requirements Testing	Viewpoint - Oriented Requirements Validation
Team Size	Large	Small	Large	Small
Cost	More Costly	Less Costly	More Costly	Less Costly
Organization Size	Large	Small & Large	Large	Small & Large
Reuse	N/A	Yes	Yes	N/A
Customer Involvement	No	Yes	No	Yes

CosTest [6] can generate test cases and test results automatically from a UML model that has been modified using Action Language All for requirements validation. All provide the UML implementation details, which comprise activity diagrams and class diagrams. This means that in addition to requirements specifications, CosTest will also need a design that outlines how system operations should be implemented and encapsulated into classes. Additionally, consumers and clients find test-case based validation to be inconvenient, which makes it challenging for them to use CosTest to verify their requirements in real-world scenarios.

IV. Validation Requirements Approaches

To test consistency and referencing features, a requirement meta model with formal relationship types based on first-order logic was developed with assistance from the QuadREAD project (Quality-Driven Requirements Engineering and Architectural Design) [12]. AIC (Automatic Inconsistency Checker) provides automated requirement traceability as well as visual support for identifying and highlighting inconsistencies, incorrectness, and incompleteness in collected requirements. [13]

Using textual abstract interactions and a pattern library, the model generates an essential use case (EUC) model that reveals inconsistency, incompleteness, and incorrect features.

Several machine learning and linguistic models are fully utilized in the validation and administration of requirements. [14]

Using patterns based on an empirical examination of a railway system, the SREE (Systemized Requirements Engineering Environment) program finds comprehensive, clear, exact, unequivocal, verifiable, testable, and maintainable features. The method counts likely errors in each necessary sentence on the presumption that the weight of each sentence is the same [15]

Animation is another helpful tool for making sure that requirements are met [8]. In [9], a method for interactively verifying requirements using animation is provided. It generates a mock-up prototype of the user interface using BPMN as input. The study [10] presents an online animation tool that uses goal and scenario exploration to validate requirements. It represents requirements using linear temporal logic. Objectives and XML is used to define state transitions and user interface components.

In [16], the author outlines a heat map-based visualization method for analyzing a source code repository's evolution. As for the visualization method, [16] suggests one that, as seen in Figure 2, closely integrates and synchronizes important elements of the development process with the product artifacts it generates.

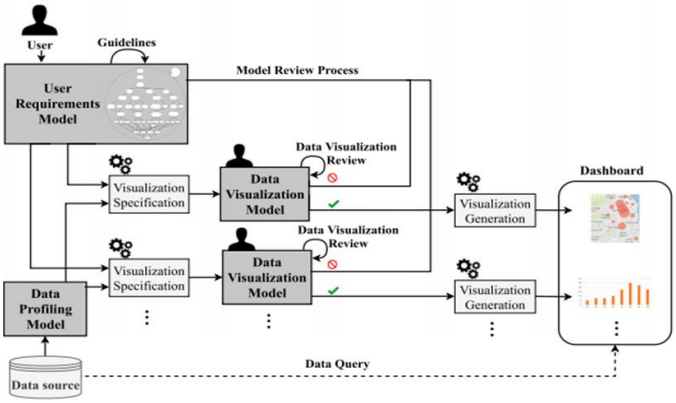

Fig. 2. Requirements Visualization [16]

Numerous authors are working in this field because of its importance. Techniques for automatically generating visuals or dashboards are proposed in [17, 18]. All of them, however, rely on the user to select the sort of visualization that will be utilized. As a result, various alternative approaches suggest methods for determining the best type of visualization.

A graphical toolkit known as Displays is presented by the author [19] to make it easier to evaluate how well software implementations adhere to clearly defined functional and operational requirements. By combining simulation and prototype technologies, the toolkit enables domain experts, designers, and pilots to assess how implementation performs about its formal specification. An actual software implementation for unmanned aircraft systems has been validated using the toolkit against a formalized standard reference algorithm.

[25] When user stories are ignored while collecting requirements, many bugs and errors are identified. The requirements verification framework defined requirements not approved in the program, and new requirements were identified in the face-to-face meeting that led to the completion of the program. Identifying unsupported requirements has saved developers from complex bugs and failures.

Future engineers could save the excess cash allotted to the maintenance phase by using the requirement validation architecture. The capacity to form conclusions on a software component's fit for a given problem stems from its attribute suitability. The program's development time has also been shortened. For one-year projects, the time needed for corrective, adaptive, optimizing, and preventive maintenance is cut down to about 1-2 months; however, in regular operations, it takes 2-3 times longer than the estimated period.

[22] introduces participants to a method for modeling and analyzing requirements in UML and automatically creating prototypes from requirements models using a CASE tool called RM2PT. The stakeholders can readily verify whether the requirements accurately reflect their needs by looking into how the use cases are implemented in the prototypes that are

generated. Furthermore, the created prototype's features allow for the automatic detection and correction of requirements inconsistencies.

[23] Presents the formalism to model the requirements of system software by combining the different use cases and defining the relationship with actors to introduce the Petri net. The functional requirements are modeled by using use case diagrams and the descriptions of use case scenarios are modeled by petri net. That phenomenon is validating the specifications of requirements in the real world. To achieve modeling of system requirements specification using use cases and a Petri net and verify their analysis to validate the specification.

[24] Sometimes the misunderstanding of the different concepts within software engineering, or what is known as evidence-based misconceptions. It is one of the most important reasons that require care and attention to the process of verifying the validity of the requirements of the software industry in general. Therefore, it was important to study and analyze the concepts of software engineering and the extent of their assimilation.

[25] Aims to find the factors influencing the selection of requirement validation techniques and evaluate critical factors from the factors. Each participant indicated his or her own list of key factors, some of which were restated. Some factors are vaguely described as to why they should be considered the main factor. The reasons for the reclassification of factors are summarized. That participant includes time, business, stakeholders, resources, and size.

[26] Reports on a laboratory-based investigation into decision support systems (DSS) prototypes for hypothetical human extravehicular activity (EVA). Demonstrating the clear connections between DSS design criteria obtained from work domain demands and the validation and verification process to assess the usefulness of DSS design solutions is a key component of this work.

[27] Give an experimentally based model for RE artifacts. Its accuracy, completeness, applicability, and unreliability have all been confirmed by ten practitioners from Big Data software development projects in business. The validation results show that the relationships and essential RE elements involved in creating Big Data software applications are captured by the model.

[26] The resultant artifact model is anticipated to help in such activities as requirements elicitation and specification; definition of specific RE processes; customizing and creating a common vision in Big Data RE projects; and creating traceability tools linking the artifacts.

[28] focuses on adding support for the subset of SCORM 1.2 criteria to the previously created content generation tool EMMA, hence expanding its usefulness. The procedure for acquiring, putting into practice, and validating the specified requirements is covered in the article.

[31] Outlines a process for requirement validation in the development of aviation systems. The requirements validation process model and methodologies are described in this paper, along with a detailed identification of each phase in the process, including inputs, outputs, roles and responsibilities, and activities. This suggested methodology complies with certification guideline ARP4754A and can be put into practice. Its completeness and applicability are evaluated using a specific case study. Refer to Figure 3.

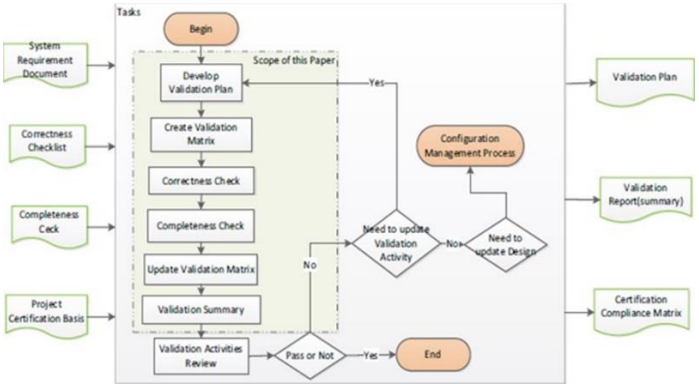

Fig. 3. The proposed validation process guidance [31]

As shown in Figure 3, we have several tasks for the validation process. These jobs start with Describing who oversees completing these duties and how the requirements will appear accurate and complete. Then, depending on the requirements management plan for each project, the validation matrix is desired to track the status of the requirements validation process in various formats.

Verify that the collection of criteria is clear, verifiable, compatible with other requirements, and necessary to meet the requirements to ensure correctness. [31] Fullness The job is to verify that this collection of right requirements, when satisfied by a system, meets user interests. After that, it is necessary to update the validation matrix by gathering, categorizing, and updating the validation data and the records in this matrix's invalidation plan. The last task is to create a validation report, which is necessary for certification and will guarantee that the requirements are correctly checked.

The thorough explanation of the processes offers direction for defining validation criteria in real-world scenarios. Furthermore, this approach may be put into practice and complies with the certification guideline ARP4754A, which considers the unique development life-cycle of civil aircraft and systems.

The requirements management phase of the validation

process deals with creating and upholding user and analyst consensus on both technical and non-technical needs. Working with well-defined, reasonable, and mutually agreed-upon requirements is the primary objective of requirements management. As a result, the tools you require will vary based on your project's goals and technique.

Therefore, we propose making a methodology that goes through these same stages to validate the software requirements at a very early stage of the software development life-cycle in the form of graphs, to help work most appropriately for the user's participation in the validation process. For this methodology to be more efficient, it should focus and target the software industry and be specific only to the task of "validating the software requirements".

V. Discussion And Proposed Work

The reviewed papers emphasize the vital necessity of requirements validation as a foundation for effective software development. According to research, approximately half of all software issues are caused by imprecise or insufficient requirements, highlighting the importance of structured validation approaches [5]. Tools such as CosTest [6] and AIC [13] automate test case development and inconsistency checks, demonstrating how technology can improve the accuracy of validation operations.

Furthermore, the combination of machine learning with visualization approaches, such as heat maps [16] and animations [8], opens up new possibilities for engaging stakeholders and boosting requirement clarity. However, these achievements are not without hurdles, such as implementation complications and the need for specialized expertise. For example, the SREE tool [15] provides a framework for measuring demand clarity but may not fully address subjective aspects of stakeholder acceptance.

Thus, future research should introduce more user-friendly technologies that speed the validation process while supporting a wide range of user proficiency. Furthermore, future research should look into combining best practices from these studies into a unified framework that supports demand assessment across several domains, particularly in new fields like Big Data and Decision Support Systems. By addressing existing gaps and increasing stakeholder engagement through improved visual and interactive techniques, the proposed work aims to bridge the current shortcomings in requirements validation, fostering a more efficient and inclusive process across software engineering projects.

VI. Conclusion

Validation of requirements is an important part of the requirements engineering process. In the literature, various validation strategies have been examined, along with their benefits and drawbacks. To successfully complete software projects without issues with accuracy and completeness of requirements, software organizations must use some sort of approach for performing requirement validation.

The technique(s) to be used for requirements validation should be explicitly defined in this approach. By doing this, they will be able to reduce requirements-related difficulties like conflicting requirements, unclear requirements, or inconsistent requirements.

References

[1] D. Pandey, “Framework for Requirement Validation (REVAF),” Int. Conf. Adv. Eng. Technol., no. August, 2018.
[2] S. Lauesen, “Software Requirements: Styles and Techniques”, Addison-Wesley, 2002.
[3] D. Pandey, U. Suman, A. K. Ramani, “Design and Development of Requirements Specification Documents for Making Quality Software Products”, National Conference on ICIS, D.P. Vipra College, Bilaspur, 2009.
[4] M. Ilyas, “Requirements Validation Techniques: An Empirical Study,” Int. J. Comput. Appl. (0975 – 8887), no. August, 2016.
[5] Gary E. Mogyrodi, “What are Requirements- Based Testing”, CROSS TALK The Journal of Defence Software Engineering, March 2003.
[6] M. F. Granda, N. Condori-Fernȗndez, T. E. J. Vos, and O. Pastor, “CoSTest: A tool for validation of requirements at model level,” in Proceedings of IEEE 25th International Requirements Engineering Conference (RE’17), pp. 464–467, Sep. 2017.
[7] A. A. Prototyping, “RM2PT: Requirements Validation through Automatic Prototyping,” IEEE 27th Int. Requir. Eng. Conf., pp. 484–485, 2019.
[8] A. Gemino, “Empirical comparisons of animation and narration in requirements validation,” Requirements Engineering, vol. 9, no. 3, pp. 153–168, Aug. 2004.
[9] G. Gabrysiak, H. Giese, and A. Seibel, “Deriving behavior of multi-user processes from interactive requirements validation,” in Proceedings of the IEEE/ACM International Conference on Automated Software Engineering (ASE’10), pp. 355–356, Sep. 2010.
[10] S. Uchitel, R. Chatley, J. Kramer, and J. Magee, “Fluent-based animation: exploiting the relation between goals and scenarios for requirements validation,” in Proceedings of the 12th IEEE International Requirements Engineering Conference (RE’04), pp. 208–217, Sep. 2004.
[11] I. Atoum, “A Scalable Operational Framework for Requirements Validation Using Semantic and Functional Models,” Assoc. Comput. Mach., pp. 1–6, 2019.
[12] A. Goknil, I. Kurtev, K. van den Berg, and J.-W. Veldhuis, “Semantics of trace relations in requirements models for consistency checking and inferencing,” Softw. Syst. Model, vol. 10, no. 1, pp. 31–54, 2011.
[13] M. Kamalrudin, J. Hosking, and J. Grundy, MaramaAIC: tool support for consistency management and validation of requirements, vol. 24, no. 1. Springer US, 2017.
[14] A. Vision, A. Ferrari, F. D. Orletta, A. Esuli, and S. Gnesi, “Natural Language Requirements Processing: A 4D Vision,” IEEE Softw, 2017.
[15] F. Fabbrini, M. Fusani, S. Gnesi, and G. Lami, “The linguistic approach to the natural language requirements quality: benefit of the use of an automatic tool,” Proc. 26th Annu. Softw. Eng. Work, pp. 97–105, 2001.
[16] A. Lavalle, A. Maté, J. Trujillo, M. A. Teruel, and S. Rizzi, “A methodology to automatically translate user requirements into visualizations: Experimental validation,” Inf. Softw. Technol., vol. 136, no. March, 2021.
[17] Kintz, Maximilien, Monika Kochanowski, and Falko Koetter. "Creating user-specific business process monitoring dashboards with a model-driven approach." International Conference on Model-Driven Engineering and Software Development. Vol. 2. SCITEPRESS, pp. 353–361, 2017
[18] Vázquez-Ingelmo, Andrea, Francisco J. García-Peñalvo, and Roberto Therón. "Application of domain engineering to generate customized information dashboards." International Conference on Learning and Collaboration Technologies. Springer, Cham, pp. 518–529, 2018.
[19] P. Masci and A. Mu, “A Graphical Toolkit for the Validation of Requirements for Detect and Avoid Systems,” pp. 1–11, 2020.
[20] M. C. Silva, V. J. P. Amorim, and R. A. R. Oliveira, “Field Research Cooperative Wearable Systems: Challenges in Requirements, Design and Validation,” pp. 1–24, 2019.

[21] N. Iqbal, J. Sang, M. Gao, H. Hu, and H. Xiang, "Forward Engineering Completeness for Software by Using Requirements Validation Framework", In SEKE pp. 523-686, 2019
[22] Y. Yang, X. Li, and Z. Li, "Rapid Prototyping for Requirements Validation: A Best-Practice with RM2PT," pp. 2–3, 2020
[23] M. A. Abrar, "Model Validation of System Requirements using Object Oriented Petri Nets," no. mxl, pp. 1–7, 2019.
[24] Carolin Gold-Veerkamp, "A Systematic Literature Review on Misconceptions in Software Engineering", pp. 1–8, 2020.
[25] S. R. Nidamanuri, "Requirements Validation Techniques and Factors Influencing them Santosh Kumar Reddy Peddireddy," no. February, 2021.
[26] D. L. Parnas and H. D. Rombach, "Requirements Capture, Documentation, and Validation," pp. 13–18, 1999.
[27] Miller, M. J., & Feigh, K. M, "Assessment of Decision Support Systems for Envisioned Human Extravehicular Activity Operations: From Requirements to Validation and Verification" Journal of Cognitive Engineering and Decision Making, 14(1), 54-74. 2020
[28] D. Arruda, N. H. Madhavji, and I. Noorwali, "A Validation Study of a Requirements Engineering Artefact Model for Big Data Software Development Projects," no. Icsoft, pp. 106–116, 2019.
[29] H. Morita and S. Matsuura, "validation method to improve behavioral flows on uml requirements analysis model by cross-checking with state transition model" v1 Mon, 1 Mar 2021
[30] S. Petrovica, A. Anohina-naumeca, and A. Kikans, "Definition and Validation of the Subset of SCORM Requirements for the Enhanced Reusability of Learning Content in Learning Management Systems," vol. 25, no. 2, pp. 134–144, 2020.
[31] X. Fei, C. Bin, and Z. Siming, "A Methodology of Requirements Validation for Aviation System Development," no. 525, pp. 4484–4489, 2020.