UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Ingeniería

https://revistas.udistrital.edu.co/index.php/reving/issue/view/1148

DOI: https://doi.org/10.14483/23448393.19411

Research

A Model for Documenting Requirements Elicitation

Un modelo para documentar la elicitación de requisites

Edgar Serna M[1]* and Alexei Serna A[1]

[1]Instituto Antioqueño de Investigación (Medellín-Antioquia, Colombia).

Abstract

Context: This work proposes a model to document the elicitation of requirements in the field of Requirements Engineering.
Method: A systematic review of the literature was conducted to determine the validity and effectiveness of the existing models for documenting requirements elicitation.
Results: By analyzing the results of this review, it was concluded that it is possible – and that is required – to take the best documented practices and add principles from logic, abstraction, and formal methods to them in order to structure a semi-formal model for documenting elicitation. Those currently proposed focus on techniques to collect information and pay little attention to documentation. In addition, these models are mainly based on natural language, which makes their interpretation difficult, and they generate re-processing in later stages of the life cycle due to ambiguities.
Conclusions: This article describes a structured model, as well as its application and validation by, comparing it against five models found in the review.

Keywords: software engineering, requirements elicitation, formal methods, specifications, documenting.

Article history

Received:
23rd/May/2022

Modified:
9th/Dec/2022

Accepted:
21st/Feb/2023

Ing, vol. 28, no. 2, 2023. e19411

*Correspondence: eserna@fundacioniai.org

Resumen

Contexto: En este trabajo se propone un modelo para documentar la elicitación de requisitos en el área de Ingeniería de Requisitos.
Método: Se realizó una revisión sistemática de la literatura para determinar la validez y efectividad de los modelos que existen para documentar la elicitación de requisitos.
Resultados: Analizando los resultados de esta revisión, se concluyó que es posible –y así se requiere– tomar las mejores prácticas documentadas y agregarles principios de lógica, abstracción y métodos formales para estructurar un modelo semiformal para documentar la elicitación. Los que se proponen actualmente se centran en las técnicas de recogida de información y prestan poca atención a la documentación. Además, estos modelos se basan principalmente en el lenguaje natural, por lo cual es difícil su interpretación, y generan reprocesos para las etapas posteriores del ciclo de vida debido a las ambigüedades.
Conclusiones: En este artículo se describe un modelo estructurado, así como su aplicación y validación mediante la comparación con cinco modelos encontrados en la revisión.

Palabras clave: ingeniería de *software*, elicitación de requisitos, métodos formales, especificaciones, documentación.

Table of contents

1. Introduction

Software is designed and developed through a series of stages known as a *lifecycle*, which is structured and developed as a solution to the needs of clients. It is teamwork involving different actors, and most of the difficulties in the process are related to requirements engineering, more specifically to the elicitation stage. Thus, if the client's needs are not properly elicited, the product will have problems, since it will not meet the established needs. Therefore, the importance of requirements elicitation lies in the fact that it provides the necessary information for the specification, which in turn is the basis for the design and development of the solution.

Elicitation must meet the needs of software engineering (SE) because the problems it must solve are complex, as well as the fact that the ensuing processes must minimize re-engineering and expedite the transition to design (1). However, the system range, the lack of comprehension of the problem, and the volatility of requirements increase the complexity of the process. This is why the different stages of requirements engineering (RE) are demanding and involve systematic and iterative procedures, during which an analyst must rely on logic and abstraction, was well as on their capabilities for modeling, with the aim to have a mental representation of the problem and start solving it. Nevertheless, the methods currently used to apply RE are informal and based on natural language, which results in a higher complexity regarding the SE process. In such circumstances, it is required to formalize either part of the process or its entirety in order to achieve a better interpretation of the client's needs.

A method is formal if it has a stable mathematical basis, which is normally supported by a specification language that allows defining precise notions such as consistency, completeness, specification, implementation, and correctness (2). By using notations and formal languages, it is possible to structure the system requirements and generate the specifications that will allow defining its behavior according to what must be done, not to how it is done (3). However, this process is incomplete if the elicited requirements are not founded upon solid documentation, which accelerates the procedures in the other stages of RE and supports the comprehension of the problem to be solved. This proposal for designing a semi-formal model to document the elicitation of requirements results from including the textual description of formal elements, with the purpose of establishing the necessary information to minimize comprehension problems regarding client needs, as well as to improve the specification.

To this effect, a systematic review of the literature was conducted in order to analyze the validity and effectiveness of the models to elicit requirements based on a perspective of documentation. By taking the best documented practices and adding principles based on logic and formal methods, a semi-formal method for documenting requirements elicitation was elaborated as a response to the current needs of SE, which were not fully satisfied by the analyzed models.

2. Method

The methodology of this work is based on a systematic review of the literature that applies the proposal made by (39). This review involved searching for standards and regulations for documenting the elicitation of requirements. The proposals with the same purpose were reviewed. During this process, two major fields were identified: background and related works.

2.1. Background

Constructing an information system is a difficult task whose issues are derived from the translation of requirements in a software project. In RE, *i.e.*, the first stage of software engineering, needs must be identified, defined, and documented to properly meet the objectives of the system. Nevertheless, although different techniques to elicit requirements have been proposed (4) (whose application allows understanding such needs), most of them do not describe a model to document them and ensure that the elicitation is visible and understood by the interested parties. To achieve this goal, requirements must be properly documented in order to ensure that the parties establish common agreements about what needs to be modeled and presented as a solution to a specific problem (5). During this review, it was not possible to identify standard guidelines or directly related procedures on how to document the elicitation of requirements. However, different processes, specifications, standards, and initiatives use this stage as the starting point for software projects, among which the following can be mentioned:

1. *Project Management Body of Knowledge* (PMBOK) (6) is a set of management practices that constitute the basis for a project management methodology. PMBOK defines requirements elicitation as the process during which the needs of the interested parties are defined and documented through initiation, planning, execution, monitoring, control, and closing. However, although it has been widely adopted by software engineers, it does not describe what and how to document in each step.
2. *Capability Maturity Model Integration* (CMMI) (7) is a series of practices and high-level processes that help organizations build a model for improving their processes and, to a certain extent, their quality. The development of requirements is part of the third level of CMMI, and its purpose is to identify, determine, and analyze the needs of the client, but, although this level describes the procedure to be followed through 13 activities, none of them are oriented towards documenting the elicited requirements.
3. *Software Engineering Body of Knowledge* (SWEBOK) (8) is defined as a guide to knowledge for SE. Specifically regarding requirements, it defines the analysis processes of client needs in order to discover and solve conflicts, find flaws, and determine how to interact with the context. Nevertheless, this stage has a serious issue with regard to the development of a software product; it does not explicitly reference or describe a procedure for documenting elicitation.
4. There are other specifications and standards related to RE, such as *ISO/IEC 12207 Information Technology – Software Life-Cycle Processes for Software Requirements Specifications, MIL-STD 490A Specification Practices, MIL-STD 498 Software Development and Documentation, SAPSS – 05 European Space Agency, ISO/IEC 9126 Information Technology – Software Product Evaluation – Quality Characteristics and Guidelines for Their Use, CMU/SEI-92-TR-012,* and *CMU/SEI-2006-TR-013*, among others. Nevertheless, although all of them accept that the documentation of requirements is essential to quicken the processes of the lifecycle, none of them describes or proposes a model, method, or process to that effect.

2.2. Related works

In addition to the previously described background, several authors have proposed and disseminated models for eliciting requirements. During this systematic review of the literature, 73 works and relevant studies were found, out of which 10 were excluded because of their title and 45 because of their abstract. The result was a sample of 18 works that met the criteria of the search protocol. During the search, little information about the topic was found, so it was decided to include research with contents on how to document elicitation.

(9) proposes a model that is useful when a set of complex conditions appears in a software component. However, it does not consider that, generally, the interested parties express these conditions in natural language, so these tables do not allow evaluating the resulting ambiguities before recognizing the actions as requirements. This proposal is more oriented towards creating a list of needs that are candidates to requirements. It has not been widely adopted by the community, which is evident because of the model's little referencing and application. The proposal made by (10) acts as a bridge for the parties, which is mainly applied to model functional requirements and does not consider non-functional requirements. Additionally, it lacks well-defined semantics, which could result in different interpretations. Despite the fact that this kind of representation is used as a model of the agreement for the parties and to describe needs, as a descriptive document, it does not provide enough information for the other stages of the lifecycle. Therefore, this proposal is limited to specifications.

(11) structured a model for representing requirements through mathematical variables, which allow defining said requirements without ambiguities. Although this model clarifies the association of client needs with elements and mathematical variables, the problem is that the required level of mathematical knowledge is very high, and most engineers and interested parties are not able to apply it. Therefore, it has not been widely adopted by the community. According to (12), it is necessary to collect and organize requirements based on different approaches, as do the interested parties. However, because this point of view recognizes multiple perspectives, it does not provide a framework to discover conflicts in the proposed requirements, nor does it allow managing inconsistencies. Moreover, a high number of perspectives makes it difficult to define the priority of the requirements, so the generated documentation is not enough to accept them.

The proposal made by (13) describes a process to elicit requirements in which the user has little participation and the formats to document them do not have a defined structure. Requirements must be expressed non-graphically and observe a specific methodology, which is not aligned with paradigms such as UML or POO. Thus, this model does not offer an efficient alternative to document elicitation, nor does it consider non-functional requirements, which must therefore be collected and documented separately. In (14), the documentation of requirements comprises a general vision and a functional description, but the product that it generates is a long document that does not offer a clear description of the needs of the parties. It adopts the IEEE STD-830-1998 standard, which is oriented towards specification, not towards elicitation.

(15) propose a set of structures to better organize requirements documentation. However, they assume that natural language has no specific formal or graphic notation, so they orient their structures towards both algorithms and programming languages, which limits freedom in the subsequent codification process. Besides, these tools cannot represent the descriptions of needs made in natural language. Hence, the interested parties do not easily reach agreements. (16) describe a theoretical framework with a variety of styles for documenting the elicited requirements with regard to the variety of situations in which the software is developed. Although this can be useful and its goal is to explain the different ways in which requirements are documented in practice, until now, and according to the results of the analysis conducted in this proposal, this diversity has not been evaluated by another

work, theoretical framework, or study case, which does not allow understanding its efficiency and effectiveness in the industrial practice.

In their proposal, (17) argue that, when elicitation does not present a semantically transparent model of the problem domain, the analyst must use underlying objects to document the process, but, although this procedure can yield good requirements, its development is usually slow, as one must work simultaneously with characteristics such as multilanguage, sequentiality, hierarchies, multidimensionality, and integrity for all the requirements. This increases the difficulty of documenting elicitation because many of these requirements are still unknown at this stage. Furthermore, the generated document is nothing more than a personal vision of each engineer, and it does not facilitate a proper discussion with the interested parties. (17) use tables to represent requirements, including their properties and relations. The problem emerges when the client makes modifications, as the structure of these tables is not flexible and does not allow re-processing. This lack of dynamism does not provide the engineer with maneuverability to maintain the document of elicitation up to date. Thus, the proposal does not have proper tracing.

(18) use agile methods to elicit most of the essential requirements through incremental development. The point here is that agile processes for requirements do not consider the planning or design of possible future requirements, and the documentation remains insufficient at every stage. Meanwhile, (19) proposes a two-variable model, which is common in Mechanical and Electrical Engineering and is used to document requirements. Although, in theory, it is useful in software systems, it is not practical due to its volatility. Besides, it cannot easily adapt to contexts where the volatility of the client needs is high and a higher dynamism is required to keep the elicitation document up to date.

(20) provide templates for use cases that are informally described along the lines of software products, and they document their use. Each study case provides one or more scenarios that describe how the system must interact, but this interaction is not reflected on the tables. Therefore, any modification of the requirements creates a new data table. (21) developed Athenea, a tool that allows users to translate their experiences and points of view, to later negotiate the requirements collaboratively. This is an experimental tool that, after six years, has not been tested in real software development environments. Thus, it is not possible to verify its efficiency and effectiveness in creating elicitation documents.

(22) describe elicitation as an important step for creating the specifications document and propose a series of templates to document it. The results include process representations, field observation notes, and interview transcriptions. However, by analyzing them, it is found that they suggest different ways for the users of the process to describe their needs, which makes it difficult to obtain the expected documentation. In this regard, the objective of the model elaborated by (23) is to simplify, structure the elicitation process, and generate a document that presents a set of selection criteria. Although it is relevant, few works on this subject were found which describe its application in small-scale projects, so its success on a larger scale cannot be ensured.

(24) propose a method to elicit and document requirements based on collaboration technologies and audio records. To structure their documentation, they propose a template that uses key terms collected and defined in a glossary. Although these terms are useful to establish a common language and avoid ambiguities, the subsequent relationship with the terms used in the documentation is not clear, and traceability becomes difficult. The model presented by (25) constitutes a way to analyze elicited and obtained data based on the information and the documentation regarding system requirements. However, although it is successful in terms of documentation, it does not assign a well-defined role to the interested parties. Thus, the final version is a vision by engineers and for engineers only, and it does not consider the required dialogue with clients and users.

3. Results

During this review, a model that allowed formally or semi-formally documenting requirements elicitation could not be found, so it was decided that a set of principles should be selected, which were deemed necessary or not by the researchers for a model of this type, as well as the better practices described in the reviewed works.

3.1. Better practices

- *Templates.* They are used as a means to collect information and document processes. In elicitation, they constitute a useful tool to store the necessary data to identify and recognize the client's needs. Some authors use them as elicitation and negotiation models, and they apply them as a complement to elicitation techniques, where the final documentation is managed directly through the interactions of the participants.
- *Schemes.* In practice, many works propose making different types of diagrams and schemes to observe elicitation factors in an organized way. Other works use conceptual schemes oriented towards objects that are obtained from the specification model. CASE tools are another option. These are especially used to detail the requirements and produce schemes. In general, the literature uses them to represent subsystems or components of a system, or the relations between them.
- *Variable matrices.* They are used as an analytic way to manage the interaction between factors in the initial layer of the logic of a program. For some authors, the variables that interact and the behavior of the system are correct only when properly documented. In general, they are used as a complement to templates because they also use variables or fields to properly associate and divide information.
- *Indicators.* The use of indicators allows evaluating the way in which elicitation contributes to improving problem interpretation. This vision is oriented to processes aimed at systematically defining the activities that make up a process, identifying the interrelation between them and the persons in charge, and introducing indicators to measure the results regarding the capacity and efficiency of the process. Consequently, criteria must be introduced which allow improving the elicitation process.

- *Scenarios.* Some authors use scenarios to describe the behavior of the system, in order to ensure a better comprehension and collaboration among parties during the requirements elicitation process, as well as to maintain the information they can recognize. This practice allows for the validation of requirements with the users. The main objective, according to the literature, is to capture the vocabulary of an application and its semantics, with the aim to facilitate the understanding of the application's functionality, as each scenario describes a specific situation and behavior.
- *Diagrams.*Some works translate requirements into diagrams to show the interaction between the different actors and the system, which adds value for the user in the form of use case diagrams specifying the system behavior. That is to say, they describe what the system does, not how to do it. This practice indicates the sequence of the processes involved, the units, and the persons in charge of execution. In this sense, diagrams are the symbolic or pictorial representations of administrative processes.

3.2. Principles

- *Logic and abstraction.* These concepts are useful because they allow understanding, analyzing, and modeling both the problem and its possible solution while making it easier for users and clients to understand what they are interpreting, e.g., the requirements. In this research, *logic* is conceived as a set of principles that allows people to judge based on evidence and make decisions supported by an activity. *Abstraction* is the process of removing or extracting the characteristics of something in order to reduce it to a set of essential particularities. In the elicitation of requirements, these notions can be used to establish the common characteristics of the needs, objects, and procedures of the users; or when two functions perform almost the same task and can be combined. To structure the proposed semi-formal model, this research considers understanding, interpretation, modeling, problem-solving, and logical reasoning. Both are necessary principles to understand both the context and the problem to be solved; they are tools that make it easier to find the necessary information to document the elicitation process.
- *Formal methods.* This concept refers to techniques and tools based on principles and mathematical postulates that are used to specify, design, validate, and verify software and hardware systems, among others. The specifications used in formal methods consist of well-established statements with regard to mathematical logic. Formal verification is based on rigorous deductions; it is the logic itself, *i.e.*, each step follows a rule of inference and can be therefore verified through a sequential process (26). During the stage of requirements engineering, formal specification is important; it is a task that requires care because its function is to ensure that both the operation and the performance of the program are correct in any situation. The principles of formal methods considered while structuring the proposed model are propositional calculus (27), decision tables (28), set theory (40), declarative languages (29), and design by contract (30).

4. Structuring, applying, and validating the model

A *semi-formal model* is understood as a set of procedure codes indicating the type of description used to document information (31), focused on creating a system model for a particular stage of the development lifecycle. Here, the objective is to perform transformations of automatic models (32). Unlike a non-formal model, its notations are intuitive, allowing for a better abstraction of the details and applying standardized and well-defined methodologies (33). Semi-formal models are widely used in the software industry because their semantics help to avoid ambiguity, inconsistency, and vagueness, as well as the fact that they are formally reasoned (34).

4.1. Structuring the model

Step 1. *Preparing the process diagram:* A process diagram is used to show the relations between the main components of a system. This diagram is also used to tabulate values of process design for the components in different operating modes. It shows the relations between the sub-processes of a system, but it hides the less important details, such as the responsibilities and denominations of agents. As a graphic representation, the model is used to visualize the process flow, showing the different activities of the system and the connections between them. This allows the interested parties to have a better comprehension of the problem, as well as to analyze the abstraction and the modeling of the solution.

Step 2. *Applying propositional calculus:* Propositional calculus is one of the methods used to formalize natural language, whereby users can express their needs, and to structure them in the form of logical propositions (35). In logic, these propositions are shown as objects of a formal language through different types of symbols, which are concatenated according to recursive rules, with the aim to build chains to which real values are assigned. By representing the needs in this way, it is possible to verify whether the interactions between the actors and the system observe the rules of the business and that, in the subsequent test activities, the expected input and output values are obtained. During requirements elicitation, it is important to represent needs as mathematical formulae, in order to structure the testing plan more easily.

Step 3. *Preparing the path diagram:* A path diagram is the representation of the ideal system behavior model which employs a directed graph. To this effect, a sequential node value and a priority and a dependence of execution are assigned according to the process diagram, whose edges describe the pre- and post-conditions. This graph shows the initial information and facilitates the construction, reading, and interpretation of the elicitation document. The iterated and represented propositions in the path diagram represent the actions that the actors perform on the system.

Step 4. *Completing the elicitation template*: The iterated and represented propositions in the path diagram are actions performed by an actor on the system, *i.e.*, a process that takes place through a series of interactions, equally formed by a set of actions. Moreover, this relation generates a set of actions that the system executes as a response to actor requests. This communication produces a change of state in the variables, the values, the database, and the hardware, which must also be described. To document this actor-system-actor communication, the model proposes a template to collect the process information. The template contains the necessary information to generate a system behavior model: *actor action*, to detail the action performed by the actor as part of an interaction; *business rules*, to verify the operations, *i.e.*, whether the resulting values are as expected and whether the test cases do not break the established rules; *system action/response*, to describe the actions or responses that the system executes as inner actions or as a response to a previous action of the actor; *state change*, to verify that the changes in the DB and those of the operational variables are the expected ones.

Step 5. *Generating the requirements report*: A requirements report details the first approach to the description of requirements extracted from the propositions, their type (functional or non-functional), and their relations. It is a table that serves as a complement for the template and that, put together, constitutes the benefit of the model for the next stage (requirements engineering) and the subsequent specification document. It is worth noting that, because of the constant changes in client needs, and because of the volatility of requirements, all these documentation processes can be updated dynamically, given that, without considering the action to be *bottom-up* or *up-bottom*, these results will be easily reflected on the other steps.

4.2. Application

The ATM network is one of the three pillars of any country's banking infrastructure. It has been increasingly common to use ATMs to make it easier for the users of the financial system to make transactions, and it is useful to promote financial inclusion. An ATM has a screen, a keyboard as the main interface, and a card reader. Additionally, it can print the receipts of the transactions. Before performing operations on an ATM, the client has to log into the system. To this effect, the client slides their card to validate their information. If the card is not valid, it is returned to the client, and, if the password is incorrectly entered three consecutive times, the ATM cancels the transaction and denies access to the system. Among others, the operations supported by ATMs are deposits, withdrawals, balance inquiries, and password changes. Each of them is registered for subsequent verification and validation. In this research, and to apply the proposed semi-formal model to document the elicitation of requirements (REDOC), the *withdrawal module* is taken as a study case.

Step 1. *Preparing the process diagram:* Fig. 1 presents the sequence of activities involved in a withdrawal from an ATM. This diagram represents the relations between the parts of the ATM system, but it must be acknowledged that this system is part of a larger, more complex system that involves the bank and its relations with other systems. A visual representation provides the interested parties with a better understanding of the abstract description of a problem, and it allows them to interpret the requirements to be met for its solution.

Step 2. *Applying propositional calculus:* After using some of the elicitation techniques, observing the users in the context of the problem, and listening to their descriptions in natural language, the process diagram is elaborated, from which the propositions shown in Table I are extracted. Since the work carried out with the interested parties is iterative, these propositions are under permanent evaluation until they are enough to solve the problem, ensuring that they describe the actions necessary to meet the needs of the parties. Table II details the propositions resulting from the iterations.

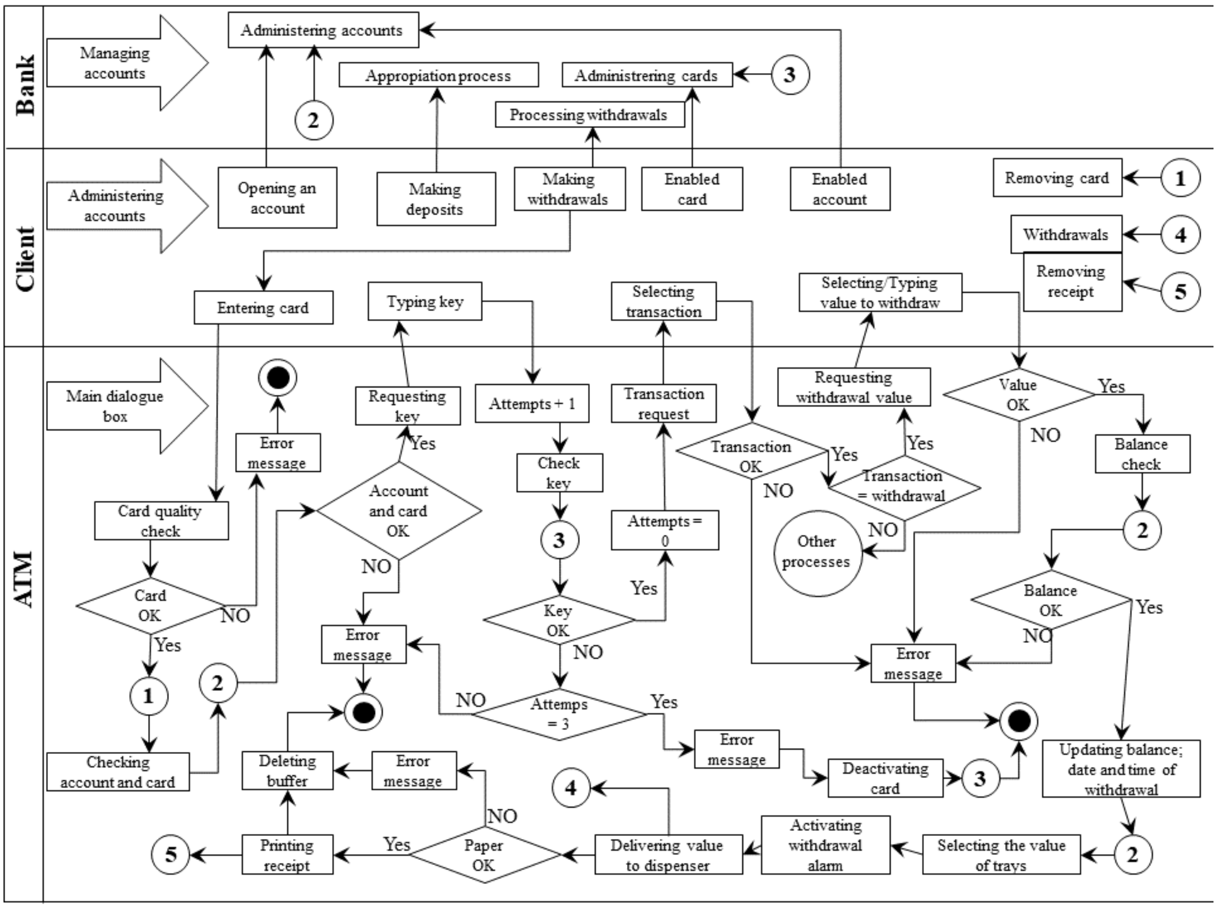

Figure 1. Process diagram for the ATM withdrawal module

Table I. Needs as propositions

Paths	Propositions
	1. The ATM displays the main dialogue box
	2. The CLIENT introduces the card in the ATM
	3. The ATM verifies card quality
	4. The CLIENT removes the card
	5. The ATM verifies the status of the account and the card
	6. The ATM displays the password dialogue box
Main	7. The CLIENT enters the password
	8. The ATM increases the number of attempts by 1
	9. The ATM verifies the password
	10. The ATM display selects the transaction dialogue box
	11. The CLIENT selects the transaction
	12. The ATM verifies the transaction

	13. The ATM displays a select/type withdrawal value dialogue box
	14. The CLIENT selects/types a withdrawal value
	15. The ATM verifies the withdrawal value
	16. The ATM verifies the balance
	17. The ATM updates the new balance
	18. The ATM updates the time and date of the withdrawal
	19. The ATM selects, from the trays, the bills corresponding to the withdrawal value
	20. The ATM activates the withdrawal alarm
	21. The CLIENT withdraws the money
	22. The ATM deactivates the withdrawal alarm
	23. The ATM verifies the existence of paper
	24. The ATM prints a transaction receipt
	25. The CLIENT removes the transaction receipt
	26. The ATM equals the variables of the defined value
	27. The ATM displays the main dialogue box
Alternative 1	1.1 The card is in bad condition
	1.2 The ATM displays an error message
	1.3 The ATM ends the process
	1.4 The ATM displays the main dialogue box
Alternative 2	2.1 The account or card is deactivated
	2.2 The ATM displays an error message
	2.3 The ATM ends the process
	2.4 The ATM displays the main dialogue box
Alternative 3	3.1 The password is incorrect and the number of tries is less than 3
	3.2 The ATM displays an error message
	3.3 The ATM ends the process
	3.4 The ATM displays the main dialogue box
Alternative 4	4.1 The typing password is incorrect and the number of tries is equal to 3
	4.2 The ATM displays an error message
	4.3 The ATM disables the card
	4.4 The ATM ends the process
	4.5 The ATM displays the main dialogue box
Alternative 5	5.1 The selected transaction is incorrect
	5.2 The ATM displays an error message
	5.3 The ATM ends the process
	5.4 The ATM equals the number of tries to 0
	5.5 The ATM displays the main dialogue box
Alternative 6	6.1 The typed value for the withdrawal is incorrect

	6.2 The ATM displays an error message
	6.3 The ATM ends interaction withdrawals
	6.4 The ATM equals the number of tries to 0
	6.5 The ATM displays the main dialogue box
Alternative 7	7.1 The typed value for the withdrawal is higher than the balance
	7.2 The ATM displays an error message
	7.3 The ATM ends the withdrawal interaction
	7.4 The ATM equals the number of tries to 0
	7.5 The ATM displays the main dialogue box
Alternative 8	8.1 The ATM has no paper
	8.2 The ATM displays an error message
	8.3 The ATM ends the withdrawal interaction
	8.4 The ATM equals the number of tries to 0
	8.5 The ATM equals the variables to the defined value
	8.6 The ATM displays a welcome dialogue box

Step 3. *Preparing the path diagram:* The first visual approach to problem modeling, the process diagram, allows understanding the context of the problem, while a path diagram models the possible solution. For this study case, and according to the iterated propositions, the resulting path diagram is shown in Fig. 2.

Step 4. Completing the elicitation template: To document actor-system-actor communication, a template has been proposed to capture the documentation of the requirements elicitation process, as shown in Table III.

- *Actor action.* This column details the action performed by the actor as part of their interaction with the system. To document it, the principle of propositional calculus in discrete mathematics is employed to translate the actions into propositions, which allows representing the requirements without ambiguities.
- *Business rules.* To document this column, the principles of logic and abstraction, design by contract, and formal methods are used. This, in order to transform the propositions into mathematical formulae, thus making it possible to verify the operations and whether the resulting values are the expected ones for the test cases. The changes in the main process and the alternatives, which are described in the path diagram, occur according to the documented conditions, and the appropriate path for each circumstance is taken. Moreover, the rules of the business established by the customer and the users must be observed.
- *System action/response.* This column uses the concept of *scenarios*, taken as good practice based on the literature review. It describes the actions-responses executed by the system as inner processes or in response to a previous action by the actor. Scenarios allow understanding the application and its functionality, and they describe each specific situation in this regard, focusing on its behavior and ensuring understanding and collaboration between the involved parties.

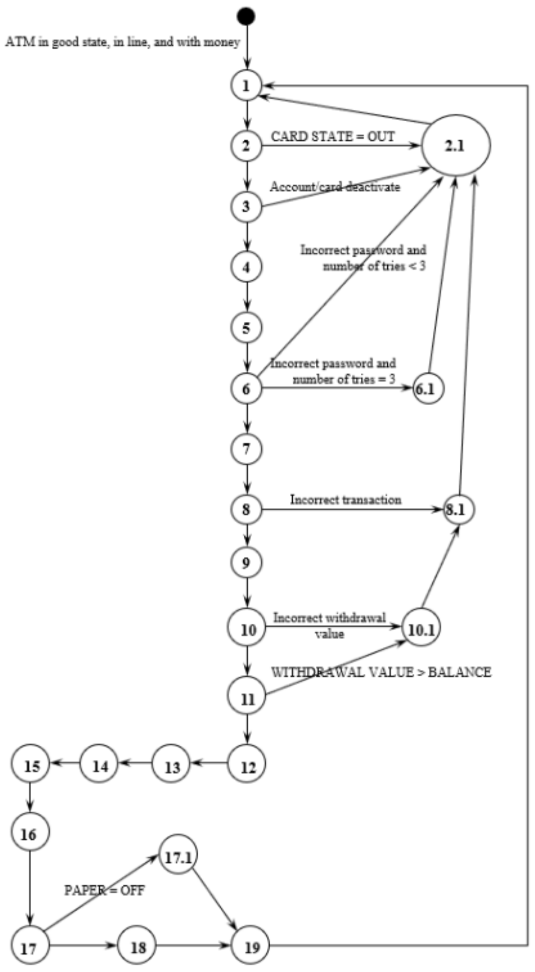
ATM in good state, in line, and with money
CARD STATE = OUT
Account/card deactivate
Incorrect password and number of tries < 3
Incorrect password and number of tries = 3
Incorrect transaction
Incorrect withdrawal value
WITHDRAWAL VALUE > BALANCE
PAPER = OFF
1
2
2.1
3
4
5
6
6.1
7
8
8.1
9
10
10.1
11
12
13
14
15
16
17
17.1
18
19

Figure 2. Resulting path diagram

- *State changes.* It is necessary to know the state changes because, in the model, the tests and quality management are run parallel to the whole lifecycle of the product. This column documents the changes that occur in the database and the operating variables, with the aim to verify that they behave as expected for the system. All the information is expressed through mathematical formulae in order to structure the necessary test plan and observe the business rules. Based on the actor action and the system action/response, it is possible to know the possible state changes regarding the information entered in the system, which will be later validated when the process follows the route established in the path diagram.

Step 5. *Generating the requirements report:* It is worth clarifying that this report does not replace the specifications document to any extent; it only represents the result of the process of documenting the elicitation. Table IV presents the format of this report. The semi-formal language used to express the requirements in this report allows designers to understand the context in which they must work, as it suppresses many of the ambiguities of natural language, which is often used by clients when expressing their needs.

Table II. Iterated propositions

Priority	Resulting proposition
1	The ATM displays the main dialogue box
2	The ATM verifies card quality
3	The ATM verifies card and account statement
4	The ATM displays a type password dialogue box
5	The ATM increases the number of tries to 1
6	The ATM verifies the password
7	The ATM displays a select transaction dialogue box
8	The ATM verifies the transaction
9	The ATM displays a select/type withdrawal value dialogue box
10	The ATM verifies the withdrawal value
11	The ATM verifies the account balance
12	The ATM updates the new balance
13	The ATM updates the time and date of the withdrawal
14	The ATM selects the withdrawal value from the tray of bills
15	The ATM activates the withdrawal alarm
16	The ATM deactivates the withdrawal alarm
17	The ATM verifies paper existence
18	The ATM prints the transaction receipt
19	The ATM equals the variables to the defined value
20	The ATM displays the main dialogue box
2,1	The ATM displays an error message and ends the process
6,1	The ATM deactivates the card
8,1	The ATM equals the number of tries to 0
10,1	The ATM ends the withdrawal interaction
17,1	The ATM displays an error message

4.3. Validation

To carry out this activity, five models were taken from the sample of the systematic literature review, in order to compare and analyze the results obtained in this same study case. This, according to a set of selected variables from other works that performed similar comparisons (36, 37), which were adapted to the objective of this research. Table V shows the variables that were used to validate the application results of the different models *vs.* those of our semi-formal proposal. The process consisted of verifying the efficiency and effectiveness of each model in comparison with the results obtained with the semi-formal model.

Table III. Template for documenting interactions

Interaction Code	001-1	**Interaction Name:** Withdrawals	**Interaction description:** The client withdraws money from their account through an ATM	
Actors:	Bank, Client, ATM	**Pre-conditions:**	ATM in a good state, ATM online, ATM with money	
Previous Dependencies	Opening an account, activating the account and card, making a deposit			
	Actor action	**Business rules**	**System action/response**	**State changes**
Main Paths	1. Enter the card	1. QUALITY CARD = YES	1. Electronically verify card quality	
	2. Remove card	2. Account ∧ active cards	2. Verify ACCOUNT STATE AND CARD: BD	
			3. Dialogue box for typing PASSWORD	
	3. Enter PASSWORD	3. Password = 9999, TRIES <3	4. Verify PASSWORD: BD	TRIES + = 1
			5. Dialogue box: TRANSACTION	TRIES = 0
	4. Select TRANSACTION	4. TRANSACTION Select options	6. Dialogue box: WITHDRAWAL VALUE	
	5. Select/Type WITHDRAWAL VALUE	5. WITHDRAWAL VALUE (option V multiple 10K)	7. Verify WITHDRAWAL VALUE	
		6. WITHDRAWAL VALUE < BALANCE	8. Verify WITHDRAWAL VALUE<BALANCE: BD	
		6. BALANCE = BALANCE – WITHDRAWAL VALUE	9. Update BALANCE: BD	BALANCE = BALANCE – WITHDRAWAL VALUE
		7. DATE = Date() ∧ TIME = Time()	10. Update DATE and TIME of withdrawal: BD	DATE = Date() TIME = Time()
			11. Select WITHDRAWAL in bills	Available money- WITHDRAWAL VALUE
			12. Take WITHDRAWAL VALUE to the dispenser	
			13. Activate the withdrawal dispenser alarm	Alarm activates
	6. Remove money		14. Verify PAPER	
			14. Print transaction receipt	Deactivate alarm
	7. Remove receipt		15. Erase buffer	Values to 0 Variables in white
Alternative Paths		1. CARD QUALITY= NO	1.1 Error Message; ENDS process	
		2. Account ∨ Deactivate the card	2.1 Error message; ENDS process	
		3. Incorrect password ∧ TRIES <3	3.1 Error message; ENDS process	
		4. Incorrect password ∧ TRIES = 3	4.1 Error message; ENDS process	CARD = OFF
		5. Incorrect TRANSACTION	5.1 Error message; ENDS process	TRIES = 0
		6. Incorrect WITHDRAWAL VALUE	6.1 Error message ENDS money withdrawal interaction	TRIES = 0
		7. WITHDRAWAL VALUE > BALANCE	7.1 Error message ENDS money withdrawal interaction	TRIES = 0
		8. PAPER = OFF	8.1 Error message ENDS withdrawal process	Values to 0 Variables in white
Previous Dependencies	Change card; Reactivate account; Reactivate card			
Post-conditions	BALANCE update; DATE and TIME update; Values to 0; Variables in white			
Observations				

Table IV. Requirements report

ID	Requirement - proposition	Requirement	Relations	Justification
1	The ATM verifies electronically the card quality	Non-functional	2	It is a system-solution external routine
2	CARDQUALITY = YES	Functional	1	
3	CARDQUALITY = NO	Functional	1-4-y	
4	The system generates an error message	Functional	1-3	
5	The Count ∧ card is active	Functional	6-7-8-9	
6	The bank verifies BALANCE and CARD	Non-functional	1-7	It is a system-solution external routine
7	The account ∨ the card is deactivated	Non-functional	6-8	It is a system-solution external routine
8	The system generates an error message	Functional	7-9	
9	The system shows dialogue box type PASSWORD	Functional	1-6-y	
10	The PASSWORD = 9999 ∧ TRIES <3	Functional	9-11	
11	The bank verifies PASSWORD	Non-functional	10	It is a system-solution external routine
12	The system increases TRIES + = 1	Functional	10	
13	The password is incorrect ∧ TRIES <3	Functional	11	
14	The system generates an error message	Functional	13	
15	The password is incorrect ∧ TRIES = 3	Functional	10-11-y	
16	The system generates an error message	Functional	15	
17	The bank makes CARD = OFF	Non-functional	15	It is a system-solution external routine
18	The system shows the TRANSACTION dialogue box	Functional	10	
19	The system makes TRIES = 0	Functional	18-20	
20	The system shows a WITHDRAWAL VALUE dialogue box	Functional	10-18-y	
21	The system receives a WITHDRAWAL VALUE	Functional	20	
22	The system verifies the WITHDRAWAL VALUE	Functional	21	
23	The WITHDRAWAL VALUE is incorrect	Functional	22-24	
24	The system shows an error message	Functional	23	
25	The bank verifies WITHDRAWAL VALUE <BALANCE	Non-functional	21	It is a system-solution external routine
26	The WITHDRAWAL VALUE >BALANCE	Functional	25-27	
27	The system shows an error message	Functional	26	
28	The bank makes BALANCE = BALANCE – WITHDRAWAL VALUE	Non-functional	22-29	It is a system-solution external routine
29	The bank makes DATE = Date() ∧ TIME = Time()	Non-functional	22-28	It is a system-solution external routine
30	The system takes WITHDRAWAL VALUE to the dispenser	Functional	29-31	
31	The system activates the dispenser withdrawal alarm	Non-functional	30	Not all ATMs have this function
32	The system verifies PAPER	Non-functional	31-33	Not all ATMs have this function
33	The system prints transaction receipt	Non-functional	32	Not all ATMs have this function
34	The system takes the values to 0 ∧ places variables in white	Functional	28-33-35	
35	The system erases buffer	Functional	34	

Table V. Validation variables

Variable	Detail
Graphic model	Level of use from the graphic representation to describe the process
Requirements readability	The documented requirement is described without ambiguities
Resolution of ambiguities	The ambiguities are solved before documentation
Capability to facilitate communication	The language and the graphics are clear, concise, and precise
Type of relation between requirements	The documented requirement has relation and continuity
Representation by type of requirements	Classification of the requirements according to their type
Semantic definition	It details and explains all the symbols and words that can cause confusion
Requirements traceability	It allows understanding the historical register, location, and trajectory of each requirement
Stage continuity	It allows advancing from one step to another transparently
Specification contribution	It generates documentation regarding the elicitation that is applicable and useful for the specification

Table VI. Summary of the validation of all the models

Variables	D 1	D 2	D 3	D 4	D 5	RE DO C
Graphic modeling	L	H	M	M	L	H
Requirements readability	M	M	M	M	M	M
Resolution of ambiguities	L	M	L	L	L	H
Capability to facilitate communication	L	M	L	M	L	M
Type of relation between requirements	M	M	L	L	L	M
Representation by type of requirements	L	M	L	L	L	H
Semantic definition	L	L	L	L	L	M
Requirements traceability	L	L	L	M	M	M
Stage continuity	L	M	M	L	L	H
Specification contribution	L	M	L	M	L	M

H: High, M: Medium, L: Low

From the literature, five proposals were selected, which were considered to be the most complete because of their steps and products:

- D1: Decision tables in software engineering (9).
- D2: Object-oriented software engineering: A use case driven approach (10).

- D3: Viewpoints for requirements elicitation: A practical approach (12).
- D4: Writing a software requirements document (14).
- D5: A template for requirement elicitation document of software product lines (20).

5. Results analysis

D1. The decision tables do not allow for iterative work between engineers and clients, since the tabular forms used to organize information are rigid. Besides, the model lacks a functional approach because its derivations are extensive and do not involve either the use of abstract or theoretical concepts, nor a structured language to document the elicitation. It is worth noting the usefulness of the Chapin and HIPO diagrams to document requirements, which are closer to the proposed process and path diagrams, although they require a structured programming philosophy, which has not been used for a long time. As a result, it was not possible to identify time ambiguities, most of them camouflaged as non-functional requirements. Besides, the graphic representation is not easily readable, and is not possible to identify the type of requirements. In addition, it does not allow tracking the requirements accepted by the interested parties before including them in the specification.

D2. Applying this model generates a wide number of diagrams that, in a certain way and depending on the system size, make the selection of the requirements difficult. However, this graphic representation is useful to customize requirements, and, with the use cases diagram, it is possible to model the functional requirements, given that it acts as a bridge between the technical actors and the clients. Disadvantages were found mainly when modeling non-functional requirements, as this approach lacks formally defined semantics, which does not allow clarifying the differences between the parties when analyzing the needs. In addition, the documentation in the tables does not represent the dynamism of actor-system-actor interactions, so it is not possible to document the state changes or their responses.

D3. Identifying issues and bad interpretations of the requirements as soon as possible during elicitation is one of the greatest difficulties of this model, too much information is accumulated because each point of view implies a different interpretation, and, in the end, unnecessary information is documented. Although it is possible to combine requirements management, the verification of inconsistencies, and their traceability, there are still multiple perspectives, which does not provide a framework to discover conflicts in the proposed requirements. In addition, this model does not allow managing inconsistencies. It is also difficult to define a requirement's level of priority, for which the generated documentation is not enough. The document generated is too long because it responds to each point of view according to the PREview tool, which is only employed at the end of the process, which makes changing the requirements difficult.

D4. The issue found when applying this model is that it does not consider how to properly document the definitions of the requirements, since the generated document only has two parts: a general vision and a functional description of the system. In addition, appendices must be included depending on the need for additional information which cannot be properly located in the document. The resulting document is extensive and does not offer a clear understanding of the needs. Therefore, it must be re-read several times to find any interpretation differences before specifying the requirements.

D5. This model uses templates to informally describe the lines of software products and document their use. The issue found is that the resulting interaction between requirements is not reflected in the tables.

Hence, any change generates a new data table. The exclusive use of natural language to describe requirements does not allow eliminating the ambiguities that may arise during the negotiation between parties, and, in the end, a kind of puzzle is obtained, which must be delivered for specification.

In general, any of the models could document elicitation in a way that could be considered sufficient to construct the specifications document. The main issue is that these models work in natural language, and they do not provide a sufficiently clear graphic representation to understand the problem and model a possible solution. Thus, the interested parties and the engineer need more time to analyze each item before assigning it as a requirement. Table **VI** summarizes the results of the assessment.

High, medium, and low levels were used to qualify the degree of fulfillment for the evaluated variables. For example, for 'graphic modeling', H was assigned if a graphic representation was used in all steps to describe the process; M if it is used for at least half of them; and L if it is used for less than half of them. Although this valuation can be subjective due to the lack of experience in the use of these models, a trend of use can be identified, where D2 and the proposed semi-formal model stand out.

To summarize, and after analyzing these data, it can be concluded that the semi-formal model's efficiency and effectiveness in documenting requirements elicitation are higher than those of most models proposed in this context. This validates the hypothesis that a semi-formal description helps to suppress the ambiguity of natural language, which allows the parties and engineers to more easily reach agreements regarding the needs that must be characterized as requirements, as well as to deliver an elicitation document from which specifications can be easily generated.

6. Conclusions

Although the community widely recognizes the importance of requirements engineering and elicitation as one of its key stages. The fact that there are few models for documenting requirements is concerning. Proper documentation is important at this stage because it allows for a better understanding of the client's needs, and it helps the engineers to better perceive the problem and model a solution that could be properly reflected in the specification. In addition, it provides a better basis to approach the rest of the stages of the lifecycle, as it can be consulted every time that the needs are not understood, or when modifications must be made. This is necessary for software projects not to exceed the initially established times or costs (38).

The REDOC semi-formal model proposed in this research starts by including the textual description of formal elements, such as those of templates, to document interactions. This constitutes an advantage over the other models evaluated because they are static and use natural language, which results in considerable differences in the interpretation of requirements. This model includes characteristics such as storage, state changes, and business rules, so it coherently, clearly, and completely represents the functionality of the project, in a way that is scalable and adaptable to the medium. In the same way, it improves communication between the actors involved in order to generate the requirements report. After implementing the model and analyzing the results, the future possible lines of work to approach the process of documenting the requirements elicitation stage are the following:

- Increasing research on formal methods in requirements engineering, as they are already widely used in other fields of software engineering, and, although they have a broad background and their usefulness and efficiency in critical development have already been demonstrated, there is still a need for more work, so that most of the engineers understand and are able to apply them.
- The automation of tests must be included in a new requirements management model. Current software testing models employ methodologies that suggest acquiring different levels of maturity during the process, but they do not provide details, which results in a complex development.
- Better training students to work with the mathematization of software engineering. Thus, the tasks and activities carried out by a software engineer could involve real engineering, which, in turn, could result in reliable software products that meet the established needs.
- It is necessary to validate the REDOC model in real cases of the industry, in order to verify the logical consistency of all its components and address the failures and complications that emerge in real applications.
- It is necessary to experiment with processes for including the results of the model in specification techniques and documents, with the aim to determine the efficiency and effectiveness of the propositions provided by the elicitation stage.

7. Contribution of authors

All authors contributed equally to the research.

References

[1] E. Serna and A. Serna, "La especificación formal en contexto: actual y futuro," in Serna E., Ed., *Métodos Formales, Ingeniería de Requisitos y Pruebas del Software*, Medellín, Colombia: Editorial Instituto Antioqueño de Investigación, 2021, pp. 124-138. ↑3

[2] E. Serna, "Formal methods and Software Engineering," *Rev. Virtual Univ. Católica del Norte*, vol. 30, pp. 158-164, 2010. ↑3

[3] M. Bolstad, "Design by contract: A simple technique for improving the quality of software," in *31st Annual Int. Symp. Comp. Arch.*, München, Germany, 2004. https://doi.org/10.1109/DOD_UGC.2004.10 ↑3

[4] E. Serna, “Analysis and selection to requirements elicitation techniques,” in *7th Col. Comp. Cong.*, Medellín, Colombia, 2012. https://doi.org/10.1109/ColombianCC.2012.6398001 ↑3

[5] A. Davis, P. Davis, and H. Davis, *Great software debates*, New York, USA: John Wiley & Sons, 2004. ↑4

[6] PMI, *Guía de los fundamentos para la dirección de proyectos (PMBOK®)*, New York, USA: Project Management Institute, 2009. ↑4

[7] SEI, “Software Engineering Institute,” 2015. [Online]. Available: http://www.sei.cmu.edu/cmmi/ ↑4

[8] IEEE, *Guide to the Software Engineering Body of Knowledge SWEBOK®*, New York, USA: IEEE Computer Society, 2014. ↑4

[9] R. Hurley, *Decision tables in Software Engineering*, London, UK: Van Nostrand Reinhold, 1982. ↑5, 18

[10] I. Jacobson, *Object-oriented Software Engineering: A use case driven approach*, New York, USA: Addison-Wesley, 1992. ↑5, 18

[11] D. Parnas and J. Madey, “Functional documents for computer systems,” *Sci. Comp. Prog.*, vol. 25, no. 1, pp. 41-61, 1995. https://doi.org/10.1016/0167-6423(95)96871-J ↑5

[12] I. Sommerville, P. Sawyer, and S. Viller, “Viewpoints for requirements elicitation: A practical approach,” in *3rd Int. Conf. Req. Eng.*, Colorado Springs, USA, 1998. ↑5, 19

[13] K. Beck, *Extreme Programming explained: Embrace change*, San Diego, USA: Addison-Wesley, 1999. ↑5

[14] T. Berezin, “Writing a software requirements document,” 2015. [Online]. Available: http://home.adelphi.edu/~siegfried/cs480/ReqsDoc.pdf ↑5, 19

[15] K. Cooper and M. Robert, “Formalizing a structured natural language requirements specification notation,” in *Twelfth Annual Int. Symp. Int. Council Syst. Eng.*, Las Vegas, USA, 2002. ↑5

[16] N. Power and T. Moynihan, “A theory of requirements documentation situated in practice,” in *21st Annual Int. Conf. Doc.*, San Francisco, USA, 2003. https://doi.org/10.1145/944868.944887 ↑5

[17] L. Mich, F. Mariangela, and N. Pierluigi, “Market research for requirements analysis using linguistic tools,” *Req. Eng.*, vol. 9, no. 2, pp. 40-56, 2004. https://doi.org/10.1007/s00766-003-0179-8 ↑6

[18] J. Atlee, S. Pfleeger, *Software Engineering*. New York, USA: Prentice Hall, 2005. ↑6

[19] D. Parnas, “From requirements to architecture,” in *New Trends in Software Methodologies, Tools and Techniques*, H. Fujita, Ed., Netherlands: IOS Press, 2006, pp. 3-36. ↑6

[20] B. Gallina, N. Guelfi, A. Monnat, and H. Perrouin, “A template for requirement elicitation document of software product lines,”, Univ. Luxembourg, Luxembourg, Tech. Rep. TR-LASSY-06-08, 2007. ↑6, 19

[21] V. Laporti, M. Borges, and V. Braganholo, “Athena: A collaborative approach to requirements elicitation,” *Comp. Ind.*, vol. 60, no. 6, pp. 367-380, 2009. https://doi.org/10.1016/j.compind.2009.02.011 ↑6

[22] C. Crabtree, C. Seaman, and A. Norcio, "Exploring language in software process elicitation: A grounded theory approach," in *3rd Int. Symp. Emp. Software Eng. Meas.*, Florida, USA, 2009. https://doi.org/10.1109/ESEM.2009.5315984 ↑6

[23] G. Murtaza, N. Ikram, and A. Basit, "A framework for eliciting value proposition from stakeholders," *WSEAS Trans. Comp.*, vol. 9, no. 6, pp. 557-572, 2010. ↑6

[24] A. Menten, S. Scheibmayr, and L. Klimpke, "Using audio and collaboration technologies for distributed requirements elicitation and documentation," in *Third Int. Work. Manag. Req. Knowledge*, Sydney, Australia, 2010. https://doi.org/10.1109/MARK.2010.5623808 ↑7

[25] A. Sajid, A. Nayyar, and A. Mohsin, "Modern trends towards requirement elicitation," in *2010 Nat. Software Eng. Conf.*, Rawalpindi, Pakistan, 2010. https://doi.org/10.1145/1890810.1890819 ↑7

[26] C. Burgess, "The role of formal methods in Software Engineering education and industry," in *4th Software Quality Conf.*, Dundee, UK, 1995. ↑8

[27] M. Soares and D. Sousa, "Analysis of techniques for documenting user requirements," *Lect. Notes Comp. Sci.*, vol. 7336, pp. 16-28, 2012. https://doi.org/10.1007/978-3-642-31128-4_2 ↑8

[28] C. Smith and L. Williams, "Best practices for software performance engineering," in *29th Int. Conf. Comp. Meas. Group*, Dallas, USA, 2003. ↑8

[29] J. Lloyd, "Practical advantages of declarative programming," in *Joint Conf. Dec. Prog.*, Peñiscola, Spain, 1994. ↑8

[30] R. Mitchell and J. McKim, *Design by contract by example*, New York, USA: Addison Wesley, 2001. ↑8

[31] A. Matta, C. Furia, and M. Rossi, "Semi-formal and formal models applied to flexible manufacturing systems," *Lect. Notes Comp. Sci.*, vol. 3280, pp. 718-728, 2004. https://doi.org/10.1007/978-3-540-30182-0_72 ↑9

[32] H. Ehrig, F. Orejas, and M. Wirsing, *Semi-formal and formal specification techniques for software systems*, Berlin, Germany: Technical University, 2000. ↑9

[33] C. Snook and M. Butler, "UML-B: Formal modeling and design aided by UML," *ACM Trans. Software Eng, Meth.*, vol. 15, no. 1, pp. 92-122, 2006. https://doi.org/10.1145/1125808.1125811 ↑9

[34] D. Harel and B. Rumpe, "Meaningful modeling: What's the semantics of "semantics¿" *Comp.*, vol. 37, no. 1010, pp. 64-72, 2004. https://doi.org/10.1109/MC.2004.172 ↑9

[35] W. Grassmann and J. Tremblay, *Matemática discreta y lógica*, Madrid, Spain: Pearson Education, 1998. ↑9

[36] M. Rebernik and M. Bradač, *Idea evaluation methods and techniques*, India: Institute for Entrepreneurship and Small Business Management, 2010. ↑15

[37] ICCD, *Developing and implementing an evaluation plan - Evaluation methods*, USA: Innovation Center for Community and youth Development, 2005. ↑15

[38] E. Serna, *Prueba funcional de Software – Un proceso de Verificación constante*, Medellín, Colombia: Fondo Editorial ITM, 2013. ↑20

[39] E. Serna, "Metodología de investigación aplicada", in Serna E., Ed., *Ingeniería: Realidad de una Disciplina*, Medellín, Colombia: Editorial Instituto Antioqueño de Investigación, 2018, pp. 4-32. ↑3

[40] S. Smith, L. Lai, and R. Khedri, "Requirements analysis for engineering computation – A systematic approach for improving reliability," *Reliable Comp.*, vol. 13, no. 1, pp. 83-107, 2007. https://doi.org/10.1007/s11155-006-9020-7 ↑8

Edgar Serna M

Eng., MsC, PhD, and Senior Researcher (IS). Professor, researcher, and business consultant. Analyst, logician, visionary, and computational theorist. Business advisor in the design, implementation, and maintenance of IT Architectures, as well as in the management of innovation and new technologies. Consultant for the innovation of education and the creation of new programs. Theoretical computational scientist with over 20 years of industry experience as a project leader in information systems and as a software architect. He is a university professor and researcher with over 30 years of experience. His areas of research are education, software engineering, computer science, complex thought, and computer mathematics, around which he has published books and articles and participated with papers in national and international events.

Email: eserna@eserna.com

Alexei Serna A

Eng., MsC, and Associate Researcher (I) in the Universus Group of Instituto Antioqueño de Investigación. His interests include digital animation and videogames, as well as formal methods and software engineering.

Email: alexei.serna@fundacioniai.org