This is a previous version of the article published in International Journal of Software Engineering and Knowledge Engineering. 2023. https://doi.org/10.1142/S0218194023500407

SPECIFYING REQUIREMENTS FOR MODERN SOFTWARE DEVELOPMENT: A TEST-ORIENTED METHODOLOGY

Alejandro Miguel Güemes Esperón[*], Francisco Maciá Pérez[†], José Vicente Berna Martínez[‡], Martha Dunia Delgado Dapena[§], Iren Lorenzo Fonseca[**]

[*, §] *Department of Software Engineering*
Technological University of Havana (CUJAE)
114th St. No. 11901 b/ Ciclovía and Rotonda, Marianao
Havana 19390, Cuba

[†, ‡, **] *Department of Informatics Technology and Computation*
University of Alicante
Carretera San Vicente del Raspeig s/n, 03690 San Vicente del Raspeig
Alicante 03690, Spain

[*]*aguemes@tesla.cujae.edu.cu*
[†]*pmacia@dtic.ua.es*
[‡]*jvberna@ua.es*
[§]*marta@ceis.cujae.edu.cu*
[**]*iren.fonseca@ua.es*

Most modern computer systems operate in distributed environments. To develop and test such applications, services and systems, it is necessary to consider the physical devices, architectures, communication, security and deployment mechanisms involved. However, the requirements specification process still replicates that of traditional applications: details remain implicit and are hidden in the description. As a result, specifications are difficult to identify and, ultimately, tests are designed in the traditional way: they overlook constraints and fail to achieve the desired effects. Our objective is to design a methodology for specifying requirements in both traditional software and applications deployed in distributed environments. We present an iterative and incremental requirements specification methodology. This methodology allows us to describe functional requirements and incorporate non-functional or quality constraints, which is the main contribution of this proposal. To ensure that quality requirements are specified during the design phase, the methodology proposes a series of phases, stages and artefacts that ensure the discovery and consideration of these requirements. In order to find out the strengths and weaknesses of our methodology, we have carried out a comparative study with other similar proposals in the literature. To this end, evaluation criteria were defined by considering standards and good practices in Requirements Engineering. The results of the comparative study show that our methodology constitutes a solid procedure for a detailed requirements specification from the beginning of the software development cycle. This represents an advance over the rest of the proposals studied. Our methodology contributes to the simplification of the design and execution phases of software testing, enabling traceability between the specified requirements and the designed test cases.

[‡] Corresponding author.

Keywords: Software Requirements Specification; Distributed Systems; Software Testing; Quality Requirements.

1. Introduction

The application industry has forged ahead relentlessly, marked by intense competition. A key factor in the sector is therefore software quality [1]. Managing and executing software development is a challenging task in which the system Functional Requirements (FRs), Quality Requirements (QRs) or non-functional requirements must be clearly established from the start. A software testing strategy must be based on the analysis of requirements as it needs to detect as many deficiencies as possible, thus contributing to quality assurance.

Today, the software industry focuses on delivering quality software in order to satisfy customers and end users. Yet FRs tend to be favoured over QRs, leaving QRs insufficiently specified and documented [2]. Recent research on Requirements Engineering (RE) has revealed the need to specify all requirements at an early stage of development. It also illustrates how, in the absence of a complete specification, a technical gap can lead to a project setback, increased maintenance costs, and the hindering of the testing process [3]. Several proposals for RE methodologies, methods, strategies, and artifacts have been advanced but they do not describe concrete mechanisms for specifying QRs in detail and establishing relationships with FRs. Quality aspects – such as security, availability, confidentiality, usability, integration, deployment, communication, and performance – often fail to be addressed due to insufficient requirements specification [4]. This leads to the overlooking of certain elements when designing the system architecture, when defining an appropriate testing strategy, in the design of the initial tests, and during the implementation of the system functionalities. The effects of undervaluing QR in the case of systems deployed in distributed environments are even worse – given their heterogeneity [5].

According to Sommerville [6], requirements can be defined as the description of what software should do, what services it will offer, and the restrictions that apply to the operations. Requirements reflect what customers need from the software: the software serves a particular purpose that can be characterised as FRs or QRs. FRs are statements of necessary system functions, of how the system should react to specific inputs, how it should behave in certain situations, and, in some cases, what the system should not do. For their part, QRs are not directly related to the specific functions provided by the system: they may be related to properties such as reliability, response time, storage space or deployment architecture features [6]. These requirements are often as important and critical as FRs [7]. Failing to consider a QR can lead to system interruptions and result in high costs to correct faults [8].

Requirements Engineering (RE) is a crucial stage to ensure that any software project is successful. It is divided into several phases: elicitation, analysis, specification, and validation of requirements [6]. Errors or deficiencies during the capture and specification phase of the requirements make the development process less productive and therefore increase the needed investments. Including adequate RE in the software lifecycle makes it less likely that such errors will occur [9]. RE plays a key role in measuring the quality of a computer system: it is the starting point of the definition of the test cases, guaranteeing that

the system meets the established requirements and, therefore, that the system is valid and functionally appropriate.

Although agile-methodology RE differs from traditional-methodology RE, it is essential in both to use tools and mechanisms that allow capturing customer needs and feedback. It is also important to integrate and establish adequate requirement traceability throughout the development phase. Agile software development promotes minimal documentation and often prioritises FRs over QRs. A minimal emphasis on documentation can help to reduce software time to market. Nevertheless, poor documentation can be counterproductive because the QRs are difficult to specify and no proof is provided of their correct monitoring and implementation [10].

As systems become increasingly distributed and hyperconnected, environments become more complex, interdependent, and therefore more intricate to specify. Distributed Systems (DSs) are a series of independent systems that operate transparently, acting as a single entity. The purpose of DSs is to decentralise information storage and processing, which also provides benefits such as more efficiency, greater fault tolerance, scalability, higher speed, and distributed processing [5].

According to [5], a DS consists of hardware and software components, located in networked computers. These components communicate and coordinate their actions simply by sending messages to each other, they are connected over a network, and can be separated by any distance: on separate continents, in the same building, or in the same room. Given their heterogeneity, DSs generate new problems that rarely occurred in traditional applications, e.g., concurrency problems, independent component failures, communication and integration issues, etc. The construction of systems in distributed environments poses multiple challenges linked to heterogeneity, safety, availability, scalability, fault detection, and appearance [5, 11].

Taking these elements into account in the requirements specification, using the identified functionalities would largely benefit the design, implementation, deployment, and testing phases. That is why several authors address the need to strengthen the formal QR specification during the initial phases.

Recent requirements specification proposals are based on the use of User Stories (US), UML artifacts, the IEEE standard 830-1998, and self-defined patterns or artifacts [2]. These existing specification mechanisms and artifacts are not always applicable to all requirements. This is especially the case when a functionality involves several tasks and some of them depend on others – because no mechanism reflects how those relationships take place in detail [3].

The works studied suggest that QRs should be kept in mind and that they are important during the software development process. There are several proposals, e.g. the NFR method, QUPER, quality models and i*, which consider QR but as a separate process from FR specification [12]. A common practice in agile development is the design of test cases in early phases, which then guide software development [6]. However, existing proposals are not directly focused on the testing phase, so the requirements descriptions that are obtained miss important elements, negatively impacting implementation, deployment, and testing.

In this work, we set out an iterative and incremental requirements specification methodology based on the results of our review of Requirements Engineering (RE) proposals. The methodology includes aspects of both agile and traditional software development. Its major contribution is a standardised procedure that ensures the definition at an early QR stage with all that this entails for the generation of acceptance criteria, scenarios and test cases.

The present article is structured as follows: Section 2 introduces the proposed methodology; Section 3 details the results of a comparative analysis of the present methodology with other similar proposals in the literature; Section 4 focuses on the results analysis; Section 5 shows an example of application of the methodology; and lastly, Section 6 presents the conclusions.

2. The Requirements Specification Methodology

The methodology is composed of three defined phases, as shown in Fig. 1:

- *"User Requirements Specification" (P1):* specifies the requirements in detail through User Stories (US).
- *"Detailed Specification of Functional Requirements" (P2):* focuses on an intricate specification of the Functional Requirements (FR) and identifying quality characteristics related to FRs.
- *"Detailed specification of Quality Requirements" (P3):* specifies the Quality Requirements (QR) in detail.

If the user makes new requirements requests after the three phases have been completed, the methodological process is re-iterated starting from P1. The implementation methodology generates the Requirements Specification File (RSF), containing the requirements specifications organised by phases.

Each phase consists of three stages:

- *Requirements description*: focuses on documenting relevant aspects that represent a source of information for the development, architecture, and testing of the system.
- *Requirements analysis*: centres on the semantic and functional analysis of the requirements description. From the artefacts obtained in the previous stage, which contain the requirements description, it is verified that the information requested is complete, functional, and syntactically coherent. The quality of the analysis will depend on the analyst's experience in software development and in working with the proposed artefacts.
- *Change management*: manages all the changes that occur while the phases are executed.

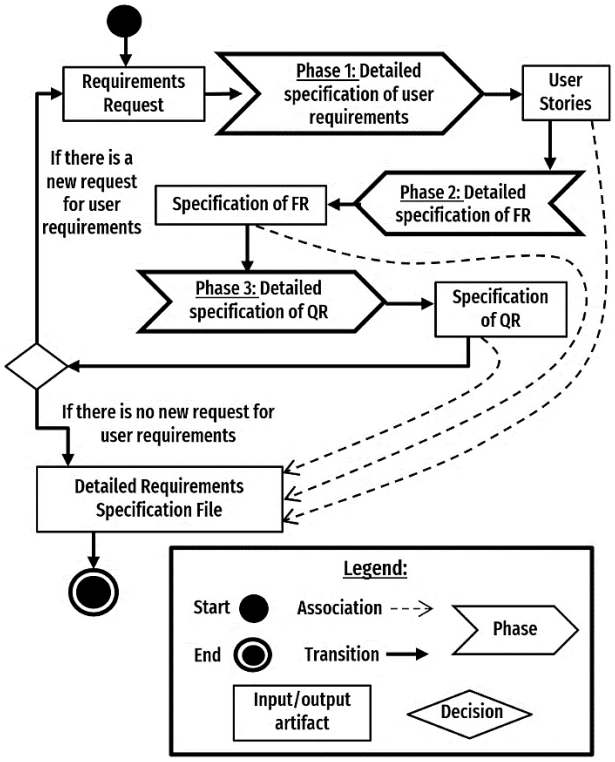

Fig. 1. Proposed methodology for requirements specification.

Since the requirements are subject to continuous improvement, each stage consists of a set of activities that can be performed several times. The methodology should start to be applied in the early stages of the system design or redesign. Requirements can, however, be specified at any time. The execution of the activities planned for the requirements description and requirements analysis stages varies depending on the phase. This is not the case of the change management stage, in which the activities will be the same across all phases.

The requirements specification process in each phase involves users, the software analyst, and the software architect. The software analyst can consult developers whenever appropriate, even if they do not directly take part in documenting the requirements. Given that several actors are involved, it is necessary to establish a collaborative environment, thus allowing to accomplish the activities of each stage more effectively.

The change management stage must ensure that the actions performed in the previous stages are properly documented. All documentation generated throughout the methodological cycle should remain accessible and available at any time to interested parties. At this stage, it is necessary to work with the RSF: it stores the different versions of the specifications as well as the respective change controls and records of the improvements made to the requirements documentation.

To ensure maximum accuracy in the description of our methodology and to avoid ambiguous definitions, we chose to define it using a Business Process Model (BPM) and the Eriksson-Penker notation for BPM notation (BPMN), together with the Universal Modeling Language (UML). This makes the methodology easier to understand and apply in various software development contexts.

2.1. *Phase 1: Detailed specification of user requirements*

The "Detailed specification of user requirements" (P1) phase unfolds during the survey requirements. This is when you interact for the first time with the customer, who is at that moment lacking a solution or is exposing a series of needs. This step is essential to ensure that the developed product meets the user's outlined expectations.

It is important to understand the user's needs in any software development project. Different techniques can be used to ensure this such as conducting interviews, brainstorming, and administering questionnaires. Procedures and artifacts that allow capturing and documenting the user-provided information should be used following the principle that every single user need represents a requirement to be met by the system.

As shown in Fig. 2, this phase contains three stages: "Description of User Requirements" (P1S1); "Analysis of User Requirements" (P1S2); and "User Requirements Change Management" (P1S3).

Requests for user-provided requirements are met during the P1S1 stage. The User Stories (US) are collected in a US document that describes the characteristics or functions to be developed, the roles concerned, and the results – in accordance with the good practices of the agile SCRUM methodology. In addition, Acceptance Criteria (AC) are included: they will be defined by the user and verified by the analyst. The CAs will reflect different scenarios associated with the requirement to be described in the US. The description should contain the name of the AC, the context in which an event will occur, the event itself and the expected outcome or behaviour after the occurrence of the event. The result of P1S1 is the generation of the US.

In stage P1S2, the software analyst performs a semantic and functional analysis of the information contained in the USs. The analyst must verify that the description is inambiguous, coherent, complete, does not hide other user needs, and is not contradictory from a functional perspective. Based on the experience gained in other development projects, the analyst may suggest changes in the US. He or she may even propose new acceptance criteria to the user. These criteria are defined in the final AC that determine the development of the system. In this proposal, the analyst must include quality characteristics that fit the needs raised by the user. These ACs will describe different requirement-based scenarios which govern the development, deployment, testing and delivery of the system. If contradictory elements are found, whether from a functional or syntactical perspective, the user requirements description stage will be re-iterated. Once the detected conflicts are solved, we proceed to identify possible system functionalities, classifying the user's FRs or QRs, and identifying any possible dependencies among the USs.

Some QRs are not directly related to an FR but influence the system as a whole, such as the availability, confidentiality and integrity of the information. In this case, the analyst

must explain what these QRs consist of to the user and guide the description of the user's requirements, in order to obtain preliminary information. The latter will then allow to specify the QRs in detail in Phase 3 with the software architect.

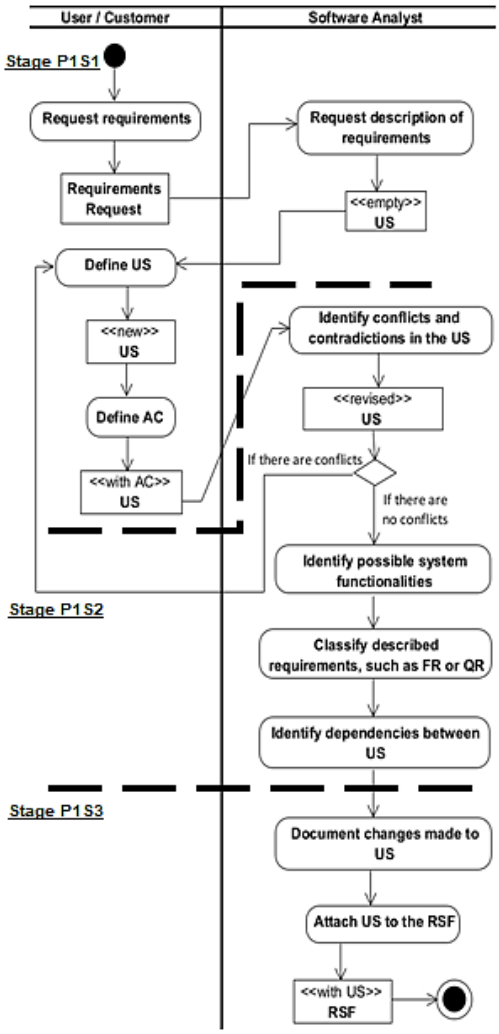

Fig. 2. UML 2.X activity diagram associated with Phase 1 “Detailed specification of user requirements”.

Upon completion of the first two stages, stage P1S3 is executed. At this stage, the changes made in the previous stages are documented and the USs are incorporated into the project’s Requirements Specification File. The phase is considered to be finalised when all user requirements have been specified.

2.2. *Phase 2: Detailed specification of functional requirements*

The "Detailed specification of functional requirements" (P2) phase takes place after P1 is completed. The software analyst also intervenes in this second phase, as he or she must identify and describe the FR based on the US obtained in P1.

As shown in Fig. 3, P2 is divided into three stages: "Detailed description of functional requirements" (P2S1), "Functional Requirements Analysis" (P2S2), and "Functional Requirements Change Management" (P2S3).

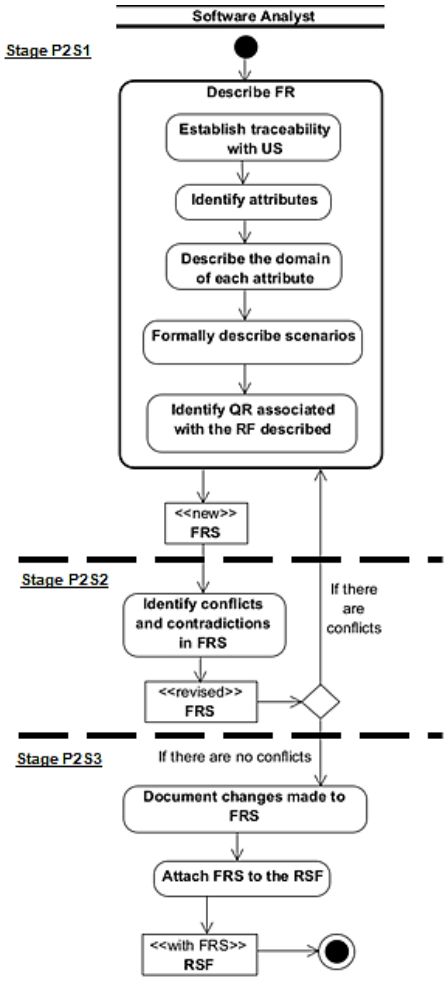

Fig. 3. Activity diagram (1) UML 2.X associated with Phase 2 "Detailed specification of functional".

In P2S1, the software analyst must include a description of the US information relating to the Functional Requirement (FR) to be described in the Functional Requirements Specification (FRS) document. It must also contain the attributes and possible scenarios in accordance with the acceptance criteria defined in P1. The data of the software analyst and the developer in charge of implementing the functionalities can also be reflected in the

document. Once the prior FR elements have been described, the software analyst must identify the quality constraints associated with this FR that can become QRs in P2S2.

The P2S2 stage centres on identifying the semantic and functional problems existing in the FRS, verifying each element in the description. The emphasis should be placed on describing the variable domains and constructing the conditions that make up the scenarios, in the latter case verifying that relational operators are properly used. If conflicts are detected in the description, P2S1 is re-iterated in order to correct the identified deficiencies. Stage P2S3 documents all changes made to the FRs, including the conflict solutions found. In addition, the detailed FR specifications must be annexed to the RSF.

This phase produces the definition of FRs and identifies the associated QRs, which will be documented as a solution to the user requirements described in P1 through User Stories. This enables to ensure the proper tracking via the historical record contained in the RSF. The detailed FR description offers an overview of how to implement the functionalities, since it allows to define specific programming tasks. In addition, using the study of the variables and scenarios, it is possible to apply software engineering and artificial intelligence techniques in order to design functional and unit test cases that cover each scenario at the early stages, even before the implementation.

2.3. *Phase 3: Detailed specification of quality requirements*

The "Detailed specification of quality requirements" (P3) phase begins once P2 is completed. The objective of P3 is to specify the QRs early and in detail, thus providing important elements for the software development cycle of design, implementation and testing. This phase involves the software analyst and software architect.

Fig. 4 shows the three stages of P3: "Detailed description of quality requirements" (P3S1), "Quality Requirements Analysis" (P3S2) and "Quality Requirements Change Management" (P3S3). The proposed procedure is illustrated in a graph below.

In P3S1, the analyst must obtain the Quality Requirements Specification (QRS). The stage involves analysing the FRs and their quality characteristics. In addition, the QRs that are not directly associated with a specific FR must be identified, because overlooking them would affect the global functioning of the future system.

The QRs are typified according to the quality attributes present (availability, integrity, confidentiality, legality, etc.). The elements to be included in the QR description must provide information allowing to conceive an adequate deployment architecture and to design test cases that enable the detection of defects associated with these QRs. Therefore, QRs will present their own attributes and scenarios which need to be described.

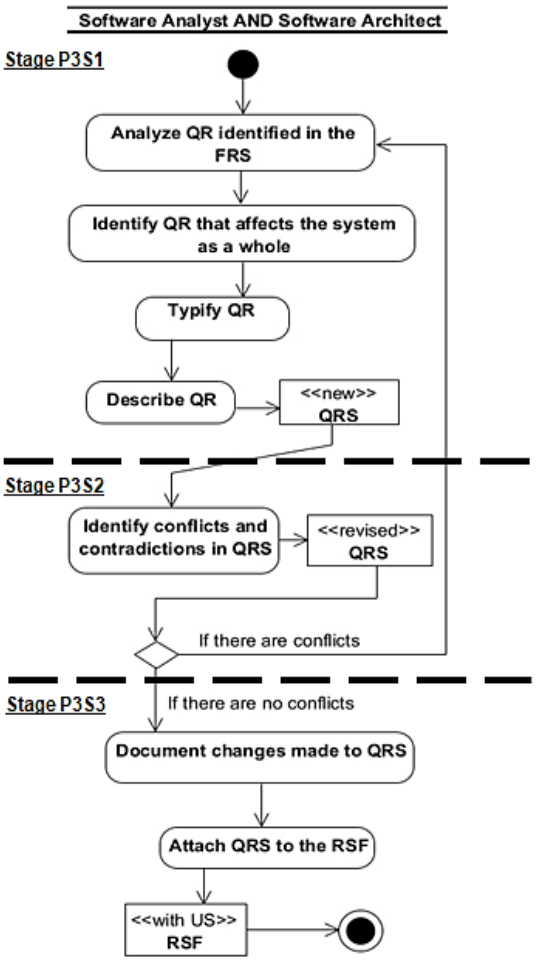

Fig. 4. UML 2.X activity diagram associated with Phase 3 "Detailed specification of quality requirements".

The "Analysis of quality requirements" (P3S2) stage takes place next. In this step, contradictions or semantic and functional conflicts are identified, taking into account each characteristic that must be specified in P3S1. As in the rest of the phases, these conflicts must be solved, implying a re-iteration of P3S1. Once the QR analysis stage has been completed and no conflicts and contradictions are found, P3S3 is executed. In this stage, all changes made during the description and analysis of the quality requirements must be documented and the QRS must be annexed to the Requirements Specification File.

3. Results

To evaluate the results, a comparative study was performed. The objective was to detect strengths and weaknesses with respect to the other included proposals. To identify studies similar to our proposal, a Systematic Mapping Study (SMS) [13] was conducted.

The SMS research question was: “Have any software requirements specification methodologies been proposed in any other works?” To answer this question, a search chain containing concepts of Population and Intervention was developed, in accordance with Petersen [13].

The terms considered were “Requirements Engineering” for the Population, and “methodology” for the Intervention. The final search string defined was (“software requirement” OR “requirements engineering”) AND (methodology OR process OR method) AND (elicitation OR analysis OR specification OR validation) NOT (“goal-driven” OR goal OR driven OR “natural language processing” OR “machine learning” OR “ontologies” OR “ontology”). The chain contained main terms, alternative terms, and terms derived from the exclusion criteria.

The selected databases were IEEE Xplore, Scopus and Web of Science (WoS). The inclusion criteria defined were:

- *Type of study*: conference proceedings and articles;
- *Language*: English;
- *Publication date period*: 2018-2022.

We decided to exclude works based on Natural Language Processing (NLP) and Machine Learning (ML), since they did not contribute to answering the research question of the designed SMS.

The consultation took place in December 2022. Applying the defined search chain to answer the research question, a total of 718 initial studies were obtained, including 440 papers presented at conferences and 278 articles, as shown in Fig. 5.

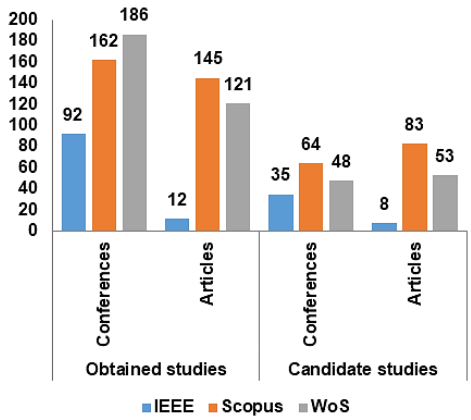

Fig. 5. Evolution of the study selection process.

In a second step, the set of studies obtained was refined, excluding articles that had been cited less than twice by articles belonging to the same database. As a result, an initial set of studies was obtained, consisting of 147 conference papers in the field and 144 articles.

After reading the titles and abstracts of the preselected studies, 27 works were chosen, as the rest did not focus on the subject of interest.

To perform the final selection, we read the proposals and the citation index of each in Google Scholar in detail. As a result, we selected the 4 most cited proposals that established concrete requirements specification procedures. Table 1 shows the 5 studies finally selected.

Table 1. Studies selected to carry out the comparative analysis.

Studies	***Databases***	***Publication date***	***Citations in Google Scholar***	***Quartile of the journal in JCR Impact Factor / Category***
[14]	Scopus	2020	22	Q1 (Computer Science, Software Engineering)
[15]	WoS	2020	9	Q2 (Computer Science, Software Engineering)
[16]	WoS	2021	9	Q3 (Computer Science, Theory & Methods, Software Engineering)
[17]	IEEE	2020	2	-
[18]	Scopus	2017	1	-

In addition, we decided to include [18] because, despite being published in 2017, it presents the result of our research and an antecedent of the methodology proposed here.

The evaluation criteria to perform the benchmarking study are shown in Table 2. They were proposed by the authors, based on the study of similar proposals, Requirements Engineering concepts, good practices and activities described in ISO/IEC/IEEE 29148 [19]. Table 3 shows the results of the comparative study.

4. Discussion

The 5 works selected for the comparative study present similarities and differences with the methodology we propose in this article. The main contributions of each work are summarised below.

The Requirements Specification for Developers (RSD) approach [14] proposes to create Software Requirements Specifications (SRS) that provide necessary information for developers. This approach does not include a mechanism for identifying and describing dependencies that may exist between requirements, sometimes hindering the readability of requirements descriptions. On the other hand, the findings showed that the practices used to specify requirements using the RSD approach potentially produce a more objective SRS, adapted to the development team.

ScrumScale [15] was developed based on the fundamental principles of the agile Scrum method, complex adaptive systems theory, game theory, and object-oriented technology. This new method incorporates a vision of the system and organisation scalability in the early stages, reinforcing this quality characteristic in the defined requirements specification process [20]. However, it does not provide tools that facilitate the specification of other quality requirements.

The methodology proposed in [16] addresses a major emerging issue: the specification of a new generation of health platforms capable of monitoring people's quality of life,

given that older adults are expected to represent the majority population in the years to come. This methodology supports the entire elicitation and requirements specification process: from the initial phase of gathering clinical, technological and end user requirements, to the choice of the most appropriate solution. The main limitation of this proposal is that it is only applicable to the field of smart health.

A framework to solve the problem of converting informal ideas into well-structured ideas – that make up the requirements specification document – is proposed and developed in [17], based on automating the Software Requirements Specification (SRS) method. SRS follows the requirements specification IEEE format and dedicates activities to describe the FR and other requirements. It does not support the detailed specification of all types of QRs, but does include some isolated elements related to user interface and communication.

The Model to automatically generate search-based early tests (MTest.search) is presented in [18]. It is a result of our research group that constitutes a background to the work we are presenting. The proposal includes activities to specify requirements through User Stories (US), the generation of test cases, as well as the optimisation model – that reduces the number of functional test cases considering scenario coverage criteria and using heuristic search algorithms. The US contains information on variables and user-defined scenarios. The methodology proposed in this paper will be integrated with MTest.search for the generation of test cases.

As shown in the comparative study results, the selected proposals do not offer a framework allowing for an automatic requirements specification process, with the exception of [17] and [18].

The proposed procedures generally focus on a generic context of application, except [16], which only applies to Smart Health Software and thus facilitates its execution. This can cause, however, characteristic elements of certain application domains to be overlooked in the requirements description. As can be seen in the comparative study and especially in Table 3, our methodology does the same as the proposals studied. Moreover, it incorporates the QR specification. Therefore, it is applicable to at least the same domains as all of them..

Following the current trend of software development, the proposals analysed in the comparative study include concepts, artifacts, and good practices of agile methods. However, they do not propose activities that contribute to specifying USs, FRs and QRs in detail, leading to a loss of relevant information that could be used in the design, development and testing stages of software development. In addition, the way the procedures are defined in these works do not ensure traceability among the requirements. In our methodology we establish, from the artefacts built in each phase, the traceability between US (with its acceptance criteria), FR and QR, showing the dependency relationships between them.

The methodology exposed in this work constitutes a solid procedure to perform a detailed requirements specification from the very beginning of the software development cycle. This represents a step forward with respect to the rest of the proposals studied.

Table 2. Evaluation criteria.

Identifier	*Evaluation Criteria (EC)*
C1	Automated Framework
C2	Type of proposed solution
C3	Context of application of the proposal
C4	Roles involved
C5	Requirements Specification Tools
C6	User Requirements (UR) Specification
C6.1	Definition of Acceptance Criteria (AC)
C6.2	Identification of dependencies between UR
C6.3	Identification of Functional Requirements (FR) based on UR
C6.4	Identification of Quality Requirements (QR) based on UR
C7	FR specification
C7.1	FR traceability with UR
C7.2	Attribute identification
C7.3	Domain description of the identified attributes
C7.4	Description of scenarios
C7.5	Identify QR associated to specified FR
C7.6	Identification of dependencies between FR
C8	QR specification
C8.1	Analysis of QR identified in FR
C8.2	Identification of QR that can affect the system in general
C8.3	QR typing
C8.4	Description attributes and scenarios
C9	Identification of conflicts and contradictions in requirements specifications
C10	Solution to the conflicts and contradictions identified in the requirements specifications
C11	Traceability between UR, FR and QR
C12	Management of changes in requirements
C13	Up-to-date requirements documentation accessible to all members of the development team

Table 3. Results of the comparative study.

EC	*Studies (Ss)*					
	Our methodology	**S1 [14]**	**S2 [15]**	**S3 [16]**	**S4 [17]**	**S5 [18]**
C1	-	-	-	-	X	X
C2	Methodology	Approach	Method	Methodology	Framework	Model
C3	Generic and Distributed Systems	Generic	Generic	Smart Health Software	Generic	Generic
C4	User/Custo-mer, Software Analyst and	Analyst, Product	Product Owner, Scrum	User/Customer	User/Custo-mer,	User/Custo-mer,

	Software Architect	Owner and Customer	Master, Scalability Expert		Software Analyst	Software Analyst
C5	User Stories	Templates, Mockup	User Stories	User Stories	Templates	Templates, User Stories
C6	X	X	X	X	X	X
C6.1	X	-	X	X	-	-
C6.2	X	X	-	X	-	-
C6.3	X	-	X	-	-	-
C6.4	X	-	Or	-	-	-
C7	X	X	X	X	X	X
C7.1	X	X	X	X	-	X
C7.2	X	X	-	-	-	X
C7.3	X	-	-	-	-	X
C7.4	X	-	-	-	-	X
C7.5	X	-	-	-	-	-
C7.6	X	-	-	X	-	-
C8	X	Or	Or	Or	Or	-
C8.1	X	-	Or	Or	-	-
C8.2	X	-	Or	-	Or	-
C8.3	X	-	Or	-	Or	-
C8.4	X	-	-	-	-	-
C9	X	-	X	X	X	-
C10	X	-	X	X	X	-
C11	X	Or	Or	-	-	-
C12	X	-	X	-	X	X
C13	X	X	X	X	X	X

-Legend-

X: Yes Or: Partially -: No

5. Example of application of the methodology

The proposed methodology was applied for the specification of requirements for the Safety and Security System of an educational institution. In this project, students and specialists integrate a multidisciplinary team to develop the system. The requirements specification process involved the Users/Customers, the Software Analyst and the Software Architect, each in their corresponding phase, keeping in mind the flow defined in the methodology. As a result, 11 requirements were specified, including 8 FRs and 3 QRs.

The result of applying the methodology to the project is shown below. For this purpose, the artefacts obtained during the specification of the requirement "Staff access control in an educational institution" are shown.

Table 4 shows the US described, along Phase 1, by the person wishing to access the educational institution. In addition, the ACs are included.

Table 4. Template for the specification US-0001.

US Identifier	***Priority***	***Estimate (days)***	***Creation date***
US-0001	1	20	20-03-2023
US name	Staff access control in an educational institution		
User/Customer name	Randy Orozco Vázquez		
Rol	**AS a person**		
Characteristic/Functionality	**I WANT access to the institution**		
Reason/Result	**TO develop or coordinate activities in the institution**		
Acceptance Criteria (AC)			
AC name context	**Event**	**Expected result/behaviour**	
Because I work in the institution	When I present my identification	Access must then be authorized at the physical barrier, the date and time of access must be recorded, my details as an employee must be recorded and the security officer must be shown on the screen: my photo, the area in which I work and my level of access.	
Because I study at the institution	When I present my identification	Access must then be authorized at the physical barrier, the date and time of access must be recorded, my details as a student must be recorded and the security officer must be shown on the screen: my photo, the area in which I work and my level of access.	
Because the person does not have authorised identification	Whe he/she presents his/her identification	Access should then not be authorised at the physical barrier and an alarm should be indicated to the officer.	

Table 5 shows the specification, made by the Software Analyst in Phase 2, of the FR associated to US-0001. As shown in the template obtained, this FR has an associated availability QR that was described in Phase 3 by the Software Analyst and the Software Architect as shown in Table 6.

Table 5. Plantilla para la especificación del FR-0001

FR identifier	***US identifier***	***FR name***	***Creation date***
FR-0001	US-0001	Entrada de personal	23-03-2023
Software Analyst	Juan Pablo Gómez Estrada		
Description	Once the person arrives at the educational institution he/she should be allowed entry or not depending on the role and the area to which he/she has access.		
¿QRs identificated?	**X** Yes __ No		
QRs associates	__ Scalability __ Performance **X** Availability __ Integrity __ Maintainability __ Portability __ Legality __ Culture and politics __ Appearance __ Reliability Others: _____		
Variables			
Nombre	**Tipo o Dominio**	**Descripción**	

role of the person	Enumerated	The roles are: security officer, area manager, worker, visitor, temporary and student.
person	Object	Each person's: ID, photo, name, age, area, gender and role are known.
date	Date	Date of access to the educational institution
people	Set	Set of persons who have links with the institution
Scenarios		
Name	**Condition**	**Expected result**
Worker entry	**IF** you have identification from the institution **AND** you are a worker	Access must then be authorized at the physical barrier, the date and time of access must be recorded, the worker's ID must be recorded, and the security officer must be shown on the screen: my photo, the area in which I work and the level of access.
Student entry	**IF** you have identification from the institution **AND** you are a student	Access must then be authorized at the physical barrier, the date and time of access recorded, the student's ID recorded, and the security officer shown on the screen: the student's photo and the area where the student is studying.
Visitor entry by appointment	**YES** you have visitor identification	You must then authorize access at the physical barrier, record date and time of access, record appointment ID, disable visitor ID for future entries, and show the security officer on the screen: your photo, appointment area and authorizer ID
Walk-in visitor entry	If **NO** identification **AND** authorized by an area manage	Access should then be recorded with the following data: date-time of access, reason for visit, identification of the security officer who released the physical barrier, identifier of the person authorizing access, area accessed and identifier of the person gaining access.
Unauthorised access attempt	**IF** it has an ID that does not correspond to its bearer	The officer then registers an incidence of attempted security breach related to the last access at the physical barrier and prevents the person from passing. The system issues an alert to the security officer of the institution.
Access attempt with invalid identifiction	**IF** you have an ID that is not in the register of IDs issued by the institution	Then the physical barrier is not released and sends an alert to the security officer.

The methodology proposes a template for specifying QRs that contains variables and scenarios as well as FRs, and includes a specific template for QRs associated with system availability, such as the one identified in FR-0001. Tables 6 to 10 show the sections of the detailed specification template for QR-Availability-0001.

Table 6. Section "Header" of the QR-Availability-0001 detailed specification.

RC-D Identifier	***RF identifier***	***QR name***	***Creation date***
QR- Availability -0001	FR-0001	Availability of the system during staff entry to the educational institution.	25-03-2023
Software Analyst	Laura Fundora Padilla		
Software Architect	Kendry Mora Loos		
Description	The system has to be available for online consultation when scanning the person's ID at the physical barrier and authorise access or not. In case of no connection the barrier must function as an island and let authorised persons through.		
Number of device types	**3** *(physical barriers, security officer's mobile phone, server with the sensor platform)*		

	** In this paper only the description of physical barriers is shown.*
Number of types of services/systems	**3** (online identity verification service, 1 SMS alarm notification service and 1 service for the operation of physical barriers in island mode)

Table 7. “Services/Systems" section of the detailed specification of the QR- Availability -0001.

Services / Systems			
Type of Services / Systems	**Name**	**Version**	**Frecuencia de ejecución**
Online identity verification service	Identity verification	-	By concurrency. It is executed each time an access attempt is generated.
SMS alarm notification service	Alarm notification to security officer	-	Whenever an unauthorised access attempt occurs
Service for the operation of physical barriers in island mode	Updating of authorised personnel identifications	-	Every 12 hours

Table 8. “Other Attributes" section of the detailed specification of the QR- Availability -0001.

Attributes	***Name***	***Type***	***Unit of measurement***	***Maximum Quantity***	***Minimum Quantity***
Packet filter	-	__Hardware __Software	-	-	-
Connections/requests			-	-	-
Bandwidth/throughput			-	-	-
Mean time between failures			-	-	-
Average recovery time			minutes	3	2
Response time to a request			seconds	3	2

Table 9. “Devices" section of the detailed specification of the QR- Availability -0001.

Dispositivo 1			
Device identifier	D-0001		
Device name	Physical barrier		
Description	Physical barrier for access control		
Maximum Quantity	8		
Minimum Quantity	4		
Number of deployment zones	3		
Deployment areas			
Deployment area 1			
Attributes	**Name**	**Maximum Quantity**	**Minimum Quantity**
Country	Cuba		
Province	La Habana		
Location	Main entrance 1 of the institution		
Devices in the area		2	1
Users in the area		2000	30
Deployment area 2			
Attributes	**Name**	**Maximum Quantity**	**Minimum Quantity**
Country	Cuba		
Province	La Habana		

Location	Main entrance 2 of the institution		
Devices in the area		4	2
Users in the area		2000	30
Deployment area 3			
Attributes	**Name**	**Maximum Quantity**	**Minimum Quantity**
Country	Cuba		
Province	La Habana		
Location	Service entrance		
Devices in the area		2	1
Users in the area		1500	10

Table 10. Section "Scenarios" of the detailed specification of QR-Availability-0001.

Scenarios		
Name of the scenario	**Conditions**	**Expected result**
Normal operating mode	**IF** connection is available	**Then,** queries are made to the sensorisation layer, responses are generated to authorize or deny access and the information for island mode operation is updated according to the established periods.
Island mode of operation	**IF** the connection is interrupted	**Then**, an alarm is emitted, it switches to island mode and keeps checking the status of the connection**.**
Re-establishing the connection	**IF** it was operating in island mode **AND** the connection is re-established.	**Then,** the sensorisation layer is notified of events occurring during island mode execution and normal operation mode is activated.

Further attributes of QR-Availability-0001 are described in Table 8. As shown there are several elements that in the conditions of this project it does not make sense to describe them. If relevant, each one is described. As specified in Table 6 the physical barriers must function as an island when there is no connection. This characteristic is described by the attribute Average Recovery Time, which refers to the time it takes for a physical barrier to start operating in island mode. The devices should be described as shown in Table 9, with emphasis on the zones where they will be located. In addition, it is important to define the scenarios in which the system should be available. For the case analyzed in this section, three scenarios were identified, as shown in Table 10, and in each case conditions and expected results were specified.

The information contained in the templates will directly assist in the design of test cases. In this way we have scenarios, user-defined acceptance criteria and variable descriptions from which test value combinations can be generated.

6. Conclusions

The requirements specification process in software development, both traditional and agile, contemplates descriptions in natural language, UML artefacts, user stories, among others, in which elements associated with quality requirements are generally implicit. So much so that, in complex environments such as distributed systems, the requirements specifications do not favour the design of validation tests applicable to all the aspects involved in this type of system.

The requirements specification methodology presented in this work is flexible and especially designed to develop applications and services in large distributed systems. The methodology places the specification of both functional and quality requirements on an equal footing. It contributes to the understanding of Requirements Engineering, especially the requirements specification process, reducing the problems relating to the terminology used and the activities involved in each phase.

Our methodology addresses the necessary procedures to obtain a clear interpretation of the specified requirements. It also fills documentation gaps from the moment users first describe their needs, by proposing to reflect technical aspects and establishing a correspondence between User Stories, Functional Requirements and Quality Requirements.

The main advantage of the methodology is that through the stages and documentation (artefacts) to be completed, it forces the analyst to fully and systematically detail the QRs, whereas in other methodologies they are not worked out in detail. It is important to keep in mind that if the analyst does not have sufficient knowledge, details of both QR and FR will be missed. We hope that our methodology will facilitate better testing, or at least that sufficient information is now available to do so.

The medium-term objective of our work is to create a tool that automates the activities included in the methodology. Indeed, a manual execution may require substantial efforts in the case of systems with large quantities of requirements. We are currently working on identifying important elements to consider in the description of requirements. We are also designing a standard template to specify quality requirements associated with availability, a distinctive feature of Distributed Systems. This will simplify the requirements specification process when applying the proposed methodology and will allow obtaining relevant information. In the long term, we will focus on early test generation, which is the true goal of the whole methodology: i.e., being able to dispense with implementation when performing tests thanks to a complete system specification that embraces all viewpoints.

References

[1] W. Behutiye, P. Karhapää, L. López, X. Burgués, S. Martínez-Fernández, A. M. Vollmer and M. Oivo. (2020). Management of quality requirements in agile and rapid software development: A systematic mapping study. Information and Software Technology, 12. https://doi.org/10.1016/j.infsof.2019.106225

[2] L. Montgomery, D. Fucci, A. Bouraffa, L. Scholz and W. Maalej. (2022). Empirical research on requirements quality: A systematic mapping study. Requirements Engineering, 27(2), 183-209. https://doi.org/10.1007/s00766-021-00367-z

[3] C. Ebert. (2019). Systematisches Requirements Engineering: Anforderungen ermitteln, dokumentieren, analysieren und verwalten. dpunkt. verlag.

[4] A. Ahmad, J. L. B. Justo, C. Feng and A. Khan. (2020). The impact of controlled vocabularies on requirements engineering activities: A systematic mapping study. Applied Sciences (Switzerland), 10(21), 1-29. https://doi.org/10.3390/app10217749

[5] G. Coulouris and J. Dollimore. (2012). Distributed systems - concepts and design. (5th Ed.). Addison-Wesley, Pearson.

[6] I. Sommerville. (2016). Software Engineering. (10th Ed.). Harlow, England: Pearson Education.

[7] B. Lawrence, K. Wiegers and C. Ebert. (2001). The top risk of requirements engineering. IEEE Software, 18(6), 62-63. https://doi.org/10.1109/52.965804
[8] J. Nikolic, N. Jubatyrov and E. Pournaras. (2021). Self-healing dilemmas in distributed systems: Fault correction vs. fault tolerance. IEEE Transactions on Network and Service Management, 18(3), 2728-2741. https://doi.org/10.1109/TNSM.2021.3092939
[9] M. Dadkhah, S. Araban and S. Paydar. (2020). A systematic literature review on semantic web enabled software testing. Journal of Systems and Software, 162. https://doi.org/10.1016/j.jss.2019.110485
[10] W. Behutiye, P. Rodríguez, M. Oivo, S. Aaramaa, J. Partanen and A. Abhervé. (2022). Towards optimal quality requirement documentation in agile software development: A multiple case study. Journal of Systems and Software, 183. https://doi.org/10.1016/j.jss.2021.111112
[11] M. Lahami and M. Krichen. (2021). A survey on runtime testing of dynamically adaptable and distributed systems. Software Quality Journal, 29(2), 555-593. https://doi.org/10.1007/s11219-021-09558-x
[12] T. Olsson, S. Sentilles and E. Papatheocharous. (2022). A systematic literature review of empirical research on quality requirements. Requirements Eng 27, 249–271. https://doi.org/10.1007/s00766-022-00373-9
[13] K. Petersen, S. Vakkalanka and L. Kuzniarz. (2015). Guidelines for Conducting Systematic Mapping Studies in Software Engineering: An Update. Information and Software Technology, 64, 1–18. https://doi.org/10.1016/j.infsof.2015.03.007
[14] J. Medeiros, A. Vasconcelos, C. Silva and M. Goulão. (2020). Requirements specification for developers in agile projects: Evaluation by two industrial case studies. Information and Software Technology, 117, 106194. https://doi.org/10.1016/j.infsof.2019.106194
[15] G. Brataas, G. K. Hanssen, N. Herbst and A. Van-Hoorn. (2020). Agile scalability engineering: The ScrumScale method. IEEE Software, 37(5), 77-84. https://doi.org/10.1109/MS.2019.2923184
[16] V. Bellandi, P. Ceravolo, A. Cristiano, E. Damiani, A. Sanna and D. Trojaniello. (2021). A design methodology for matching smart health requirements. Concurrency and Computation: Practice and Experience, 33(22), e6062. https://doi.org/10.1002/cpe.6062
[17] M. Duggal, N. Saxena and M. Gurve. (2020). SRS Automator-An Attempt to Simplify Software Development Lifecycle. 6th International Conference on Signal Processing and Communication (ICSC), IEEE, 278-283. https://doi.org/10.1109/ICSC48311.2020.9182768
[18] M. D. Dapena, A. M. Rojas, D. L. Uribazo, S. V. Marcos and P. B. Oliva. (2017). Model for Automatic Generation of Search-Based Early Tests. Computación y Sistemas, 21(3), 503-513. https://doi.org/10.13053/CyS-21-3-2716
[19] ISO/IEC/IEEE. (2018). ISO/IEC/IEEE 29148:2018 Systems and Software Engineering — Life Cycle Processes — Requirements Engineering. In ISO (2nd ed.). https://www.iso.org/standard/72089.html
[20] H. Edison, X. Wang and K. Conboy. (2022). Comparing methods for large-scale agile software development: A systematic literature review. IEEE Transactions on Software Engineering, 48(8), 2709-2731. https://doi.org/10.1109/TSE.2021.3069039