International Journal of Information System Modeling and Design
Volume 13 • Issue 1

The Requirement Cube:
A Requirement Template for Business, User, and Functional Requirements With 5W1H Approach

Yasar Ugur Pabuccu, R&D Center, Kuveyt Turk Participation Bank, Konya, Turkey*
https://orcid.org/0000-0002-8263-5418

Ibrahim Yel, R&D Center, Kuveyt Turk Participation Bank, Konya, Turkey
https://orcid.org/0000-0002-0052-3296

Ayse Berrak Helvacioglu, R&D Center, Kuveyt Turk Participation Bank, Konya, Turkey
https://orcid.org/0000-0002-5533-4572

Büşra Nur Asa, R&D Center, Kuveyt Turk Participation Bank, Kocaeli, Turkey

ABSTRACT

Requirements engineering activities are carried out to come up with right and suitable quality requirements. However, problems in requirements engineering remain even though there is a vast amount of requirement elicitation and writing methods. Methods are either too specific for industry application or focus on a specific part of requirements engineering. This paper proposes a new requirement writing template, the 5W1H requirement cube. The template consists of the answers to six questions (why, who, when, where, what, and how) and links business, user, and functional requirements in a hierarchy within the enterprise business architecture. The template was developed in a prominent Islamic Bank in Turkey and tested on a case study. The authors have rewritten three software requirement specification (SRS) documents with the new approach and compared them, showing that new approach has more brief and well-organized documents. BOA Cube software has been developed to implement the 5W1H requirement cube approach within the organization.

KEYWORDS

5W1H, Requirement Patterns, Requirement Templates, Requirements Engineering, Software Requirements

INTRODUCTION

Requirement elicitation is an iterative process that obtains requirements from necessary stakeholders via communication, prioritization, negotiation, and collaboration, and it uses techniques from social sciences, knowledge engineering, and group dynamics rather than computer science (Zowghi & Coulin, 2005). The whole activity regarding the gathering, elicitation, and documenting of the software requirements is called requirements engineering (RE) and it is a fundamental part of the software development process (Garcia, Segura, & Aguirre, 2020). Insufficient requirement specifications are

DOI: 10.4018/IJISMD.297046 *Corresponding Author

still a major failover reason for software projects. Studies show that poor quality requirements cause 12%–71% failover in information systems projects (Davey & Parker, 2015).

Problems in RE can be listed as communication challenges, natural language burdens, requirement changes, unnecessary requirements, the uncertainty of business needs, a lack of client support, unelaborate requirements engineering, and the nondeterministic nature of RE (Davey & Parker, 2015). The mistakes during RE arise from human errors (communication, participation, domain knowledge, specific application knowledge, process execution, other cognition), process errors (inadequate methods, management, elicitation, analysis, traceability), and documentation errors (Walia & Carver, 2015).

Requirements also have a terminology problem. Shareholders may refer the same requirement statement as user, software, business, system, or functional requirement. Besides, each role (customer, developer, etc.) demands different levels of requirement details. Different classifications exist and the prominent three ones are as follows (Alrumaih, Mirza & Alsalamah, 2018). Lauesen's (2002) classification is based on data requirements, functional requirements, quality requirements, managerial requirements. Sommerville (2011) classifies requirements as user and system requirements (system requirements are classified as functional and non-functional). Wiegers & Beatty (2013) classify requirements as business, user, and functional requirements. Business requirements are related to the purpose of software development, or why the software is being implemented; user requirement is about what users will do with the software. Finally, functional requirements describe the software's behavior based on user requirements (Wiegers & Beatty, 2013). In addition, non-functional requirements represent the system constraints, such as compatibility, reliability, performance, portability, restrictions-legality, security, and usability (Silva, Pinheiro, Albuuerque, and Barroso, 2016; Silva, Pinheiro, Albuuerque, and Barroso, 2017).

All specified requirements have to be documented. They can be represented as informal representations, semi-formal languages like entity-relationship diagrams, formal languages, or knowledge representation languages, and each has an advantage. The documentation has to be done in parallel with the level of details expected by each stakeholder and must be consistent (Pohl, 1993). Non-textual notations and diagrams are supposed to describe all requirements universally but they require a complex translation process, which may cause problems (Mavin, Wilkinson, Harwood, & Novak, 2009). Even though formal languages sustain the highest precision, most stakeholders are not familiar with them. Structured natural language supported by visual models and diagrams is the most applicable way to document requirements (Wiegers & Beatty, 2013). Finally, the written requirements form a document named 'Software Requirement Specification' (SRS). Due to the aforementioned reasons, SRS has to be prepared for the audience it addresses. SRS is exposed to natural language-related problems, similar to requirements such as ambiguity, inconsistency, incompleteness, etc. Several templates are proposed in order to cope with these problems, such as Use Cases, activity diagrams, and requirement templates, etc. (Tiwari & Gupta, 2020).

In order to solve the 'requirement puzzle', requirement patterns are a promising option. Patterns are in great use in software engineering and can be applied in the RE process. A requirement pattern is "an approach to specifying a particular type of requirement". The major contribution of the requirement patterns is guidance. Moreover, they save time and bring consistency (Withall, 2007). Requirement patterns allow organizations to reuse previously written requirements which streamline the RE process, including elicitation, validation, and documentation activities (Franch, 2015). Requirement patterns provide a "systematic way to specify a particular type of requirement in the form of a template with categories of information (Kudo, Bulcão-Neto, & Vincenzi, 2020)." On the other hand, a requirement template is a layout that consists of natural language sentences and some structure. Requirement templates are a lower level abstraction compared to requirement patterns but templates are a RE method that can be encountered in the industry (Palomares, Quer & Franch 2017).

The 'requirement puzzle' is still unsolved even with tremendous efforts and methods proposed so far. It is hard for industry practitioners to find an applicable approach despite numerous studies about

RE (Mavin et al., 2009). As a company in the finance industry, our bank is no exception regarding the difficulties of requirement elicitation and engineering. Therefore, the Information Technology department decided to develop a tool along with the methodology to cope with the aforementioned problems in a holistic manner. We have come up with a solution to set up a requirement template based on 5W1H (Who, Where, When, What, Why, How) format, which can be used to write business, user, functional, and non-functional requirements. Requirements are kept hierarchical in order to track the relations between high-level business needs and functional details of the development. Requirements are linked to business domains in the enterprise business architecture in versions in order to provide a requirement repository. We have chosen 5W1H as the basis of the structure, because it is a widely used technique in requirement elicitation, and it can be understood by both business and IT actors of the organization. The new structure is named "Requirement Cube" because six questions of 5W1H resemble a cube that has six sides. The approach relies on structured but not restricted natural language. We would like to avoid complicated notation, which is why 5W1H acts as a guide with simple rules. Each dimension of 5W1H holds a certain portion of the requirement and its content slightly changes according to the requirement level. All the rules and the structure were examined by a group of business analysts (BAs) who are the co-authors of this paper. The requirement cube structure has been tested with real examples within the company as a case study. We have reformed three SRS documents and rewritten these three documents based on the 5W1H principles and have sent six documents to 17 evaluators and asked them to score them based on nine attributes. 5W1H requirement cube template achieved a 20% higher score on average compared to our current approach. BOA Cube software has recently been developed and our organization has started using 5W1H Requirement Cube in the actual software development process while this paper was being prepared. We are not aware of any other holistic approach in RE methods that can address all types of requirements.

This paper is organized as follows: Background summarizes the related work. 5W1H Requirement Template section explains the requirement cube model in detail. Case Study section explains the empirical results. After that, conclusion is given.

BACKGROUND

There are several definitions of what a good requirement is. However, identified characteristics are similar. A good requirement must be:

· unambiguous, concise, finite, measurable, feasible, testable, traceable (Westfall, 2005)
· complete, consistent, unambiguous (Pohl, 1993)
· complete, correct, feasible, necessary, prioritized, unambiguous, verifiable (Wiegers & Beatty, 2013)

Even though there is a common understanding of the good quality requirement, there is no clear idea of how to achieve it. RE has evolved to achieve this goal. It consists of elicitation, analysis, specification, validation, and management (Carrizo, Dieste, & Juristo, 2014). Finally, the SRS document is composed.

More specifically, requirement specifications can be written in natural language, structured natural language, semi-formal languages, or formal languages (Fabrini, Fusani, Gnesi, & Lami, 2000). While specifying requirements, natural languages usually remain "syntactically ambiguous and semantically inconsistent" which makes requirements erroneous and incoherent for software engineering (Umber & Bajwa, 2011). However, in practice, it is reported that 79% of the software requirement documents are formed with natural-language-based methods because of customer preference. Therefore, there is a need for introducing an accurate, consistent, and user-friendly semi-formal method for writing software requirements (Umber & Bajwa, 2011).

Semi-formal languages are templates or graphical notations like use case diagrams, user stories, activity diagrams, and Unified Modeling Language (UML). Use cases are among essential graphical

notations or semi-formal methods. "A use case is a description of the possible sequences of interactions between the system under discussion and its external actors, related to a particular goal (Cockburn, 1999)" and use cases are widely used by industry practitioners (Jacobson, 2004). However, they are not the ultimate solution for requirement specification. The effectiveness of the use case may vary based on the chosen use case template (Tiwari & Gupta, 2020). Use cases can only take place in describing functional requirements (Lethbridge & Laganiere, 2005). Other techniques are necessary for those missing elements, and use cases can specify 25% of a system (Cockburn, 1999). UML is another important graphical notation for requirements. UML can be considered as "de facto standard" or "the lingua franca" of software engineering. It can serve as a general-purpose language and can be customized for different domains, organizations, or projects (Ciccozzi, Malavolta, & Selic, 2019). UML is more suitable for detailed design and deployment of a software system, but it falls short of analysis, documentation, and requirement specifications (Siau, 2016). Formal notations are generated with syntactically and semantically well-formed expressions (Greenspan, Mylopoulos, & Borgida, 1994). Formal languages mathematically compose rules for grammar to achieve automatic requirement elicitation and analysis by clear semantics. However, they are rarely used in practice because they require specific software tools and users (both technical and business) with high knowledge of mathematics (Yue, Niu, Wang, & Yue'L, 2019).

Structured natural languages are another option for requirement specification. Requirements written in a constrained language behave as a subset of natural language and customers' requirements and can be developed with accuracy by software experts (Luisa, Mariangela, & Pierluigi, 2004). Structured natural languages or, simply, requirement patterns provide "a systematic way to specify a particular type of requirements in the form of a template with categories of information (Kudo et al., 2019) ." Toro, Jiménez, Cortes, & Bonilla (1999) proposed two types of patterns and a requirement template. The templates are similar to use case templates. However, steps and descriptions provide a pattern of writing statements such as *"{The {actor, system} action performed by actor/system, Steps described in use case (RF–x)are performed}*". EARS (Easy Approach to Requirements Syntax) is developed in Rolls-Royce Control Systems to define high-level stakeholder requirements built on Event-Condition-Action (ECA), which is a constrained natural language (Mavin et al., 2009). The generic syntax is "<optional preconditions> <optional trigger> the <system name> shall <system response>", and its five types are: Ubiquitous, Event-driven, Un-wanted behaviors, State-driven, and Optional features. The syntax is slightly different based on the type. EARS is tested as a case study. The pattern reduced ambiguity, vagueness, wordiness problems, eliminated complexity, omission, duplication, implementation, and non-testability. The AMAN-DA is a pattern for writing security requirements to guide requirement engineers. AMAN-DA utilizes a security ontology and domain ontology. Security goals and requirements can be distinguished and written in different patterns. A case study is conducted in the maritime industry to identify stakeholders' security goals, and participants have agreed on the model's success (Souag, Mazo, Salinesi, & Comyn-Wattiau, 2018).

Toval, Nicolás, Moros, & García (2002) propose a method called SIREN (Simple Reuse of Software Requirements), which focuses on the reuse of security requirements. SIREN consists of requirement templates that are similar to use case templates. Issa & Al-Ali (2010) use a use-case-based pattern. They propose a meta use case pattern at the top and discover use case patterns based on application domains, such as financial, sales, and marketing, and come up with 174 distinct patterns. Pacheco *et al* (2015) composed RRMSRC (The requirements reuse model for Software Requirement Catalog), which is based on a commercial software tool RequisitePro. They classify requirements in detail based on priority, criticality, viability, risk, source, and type. PABRE (Pattern-Based Requirements Elicitation) framework is proposed as a meta-model to serve as a reusable requirement pattern (Franch, Palomares, Quer, Renault, & De Lazzer, 2010). It consists of 29 patterns that all target non-functional requirements. As of 2014, PABRE has 111 patterns obtained (Palomares, Quer, & Franch, 2014).

In summary, requirement writing methods use either too specific and sophisticated approaches like formal notations and domain ontologies (Siau, 2016; Yue et al., 2019) or focus on a specific type of requirements, such as stakeholder and non-functional requirements (Souag et al., 2018; Toval et al., 2002; Franch et al., 2010). Sophisticated approaches are almost impossible to be implemented in industry practices (Siau, 2016). Use cases cannot handle non-functional requirements (Lethbridge & Laganiere, 2005), and their completeness is questionable (Cockburn, 1999; Lethbridge & Laganiere, 2005). Besides, some templates or pattern-based approaches include hundreds of different templates (Issa & Al-Ali, 2010; Palomares et al., 2014), which are hard to be implemented in practice. Therefore, we studied an alternative requirement writing method to maintain the RE process in our organization.

5W1H REQUIREMENT CUBE TEMPLATE

5W1H is a well-known method for comprehending a fact thru perceiving all its important aspects. It has emerged from journalism in order to report a complete news story (Carmagnola, Cena, & Gena, 2011). By asking six simple questions (Who, Where, When, What, Why, How), it is possible to find the actor, location, time, related action, process, and reason of the action (Jabar, et al., 2013). Completeness is a common goal for both 5W1H and RE. Therefore, 5W1H is an ideal method for requirement elicitation and engineering. 5W1H is widely used in different stages of the RE process (Tiwari & Gupta, 2020), to clarify the whole process of RE (Westfall, 2005), identify the scenarios from user stories to determine use cases (Ali & Lai, 2017), structure free text user requirements in an automated way (Jabar et al., 2013), build an ontology framework to classify the concepts in requirement elicitation (Awal, Gana, & Abdulrahman, 2018), or check the completeness of use case templates (Tiwari & Gupta, 2020). In summary, 5W1H has been used in three stages of the RE process: elicitation, analysis, and validation (Tiwari & Gupta, 2020). This paper proposes to utilize it in the specification stage as well, and we are not aware of other studies employing this methodology. We assert that all requirements (business, user, functional, and non-functional) can be written by answering six questions with the rules mentioned in Appendix A. All three levels–business, user, functional, and non-functional requirements–are written with parent-child relationships.

The requirement cube is easy to understand because it relies on a general clarification method, 5W1H. It keeps essential information for a "good, clear requirement". The requirements are kept hierarchically in relation to the business domain, which is why they are traceable. Requirement cube records the aim or purpose of the requirement in "why" dimension, so requirements are verifiable. Exact user roles, conditions, timing, and the place are written in "who", "when", and "where" dimensions. Hence, the requirement cube is testable. Sine qua non properties of a good requirement–completeness and unambiguity–are holistically sustained with the 5W1H method. Finally, "how" dimension includes how the software will be developed.

5W1H REQUIREMENT CUBE TEMPLATE EXPLANATION

The requirement cube consists of answers given to 5W1H questions. 5W1H questions guide BA on how to write the requirements and combine their necessary elements, such as goals, actors, and conditions, along with the systems' design.

Why: It is the goal of the requirement, such as cost reduction, customer satisfaction, compliance with regulations, performance, security.

Who: It is the person or the system object as the actor of the requirement.

Where: It shows where the requirement takes place, such as module/channel/user interface.

When-Trigger: It is the trigger that initiates the requirement, i.e., preconditions.

When-Conditions: It indicates the conditions that have to be met, i.e., post conditions.

What: It demonstrates the actions that will be carried out and depicts what the requirement will do.

How: It explains how the software will be developed and documents the system design details.

"Why" and "How" dimensions are not parts of the requirement statement. "Why" keeps the goal of the requirement, and "How" is used to document the system design. Instead of putting all design details in one section of the SRS, the requirement cube approach utilizes the "How" dimension of each requirement for technical issues. There is a separate technical design section in the proposed SRS form but BA's didn't need to use it in the case study. This section is added in the SRS especially for technical details of the large software developments.

Example:

When <Branch Staff> <clicks the Approve Button> <on Credit Application Screen> <credit application has to be sent to supervisor> *if* <credit amount is bigger than 10.000 USD.>

Why: Reduce default risk and fraud.

How: A new workflow state will be added to the current credit application flow. A new script has to be written. The script will find the supervisor of the branch member and assign the supervisor as the approver.

The italic words (when and if) are used as conjunctions to set up a proper sentence. 5W1H dimensions are given below.

Who: Branch Staff

When-trigger: clicks the Approve Button

Where: on Credit Application Screen

What: credit application has to be sent to the supervisor

When-condition: credit amount is bigger than 10.000 USD

"How" and "*Why"* are not included in the requirement text to keep the statement simple and make it easy to follow. The remaining dimensions are sequenced based on the natural language grammar structure. The sequence of the 4Ws and conjunctions can be changed according to different languages. We believe it is a unique strength of the requirement cube, whereas almost all prominent patterns are based on the English language. Besides, it is possible to compose additional patterns, such as restriction requirements or conditional requirements to make the requirements linguistically better. However, we did not create multiple patterns in order to achieve a simple solution. The formations were initially created for the Turkish language and then revised for English.

"Who" statements should include specific roles. In the case of a general role (such as customers), adverbs such as 'all', 'except', 'only' have to be used to make requirements clear. Writing "Individual Customers" or "All Customers" is recommended instead of writing just "Customers". Therefore, the template pushes BA to think about alternative scenarios. BA has to write "always" if there is no specific trigger or condition for the "*When"* section. Using an adverb like "always" or "never" also pushes the BA and business sides to think about alternative scenarios.

We propose a similar requirement hierarchy as in Wiegers and Beatty (2016), which consists of three levels: business requirements are at the top, user requirements in the middle, and functional at the bottom. There is a why-how relationship between each level. The business requirement is the purpose that user requirements have to achieve. Similarly, user requirements are the purpose of the functional requirements, which show how user requirements will be implemented. There is no "*How"* dimension in business requirements. Fig.1 illustrates the approach in three levels and shows how each level is written and related to each other. The "how-why" relation between each level is similar to Rasmussen's abstraction hierarchy (Rasmussen, Pejtersen & Goodstein, 1994). "*When"* and "*Where"* portions are optional in functional requirements. If "*When"* and "*Where"* dimensions are blank, they will be taken to be the same as "*When"* and "*Where"* portions of user requirements. There are some commonalities with Zachman's enterprise architecture (Zachman, 1987) which uses 5W1H as contextual dimensions with six different perspectives for stakeholders. However, the Requirement Cube has three perspectives which are business, user and functional.

Business requirements stand for high-level business objectives and they are written in business terms and definitions. System objects like screens' menus and buttons are not mentioned in the business requirement. Business requirements are grouped and linked to business modules and business functions

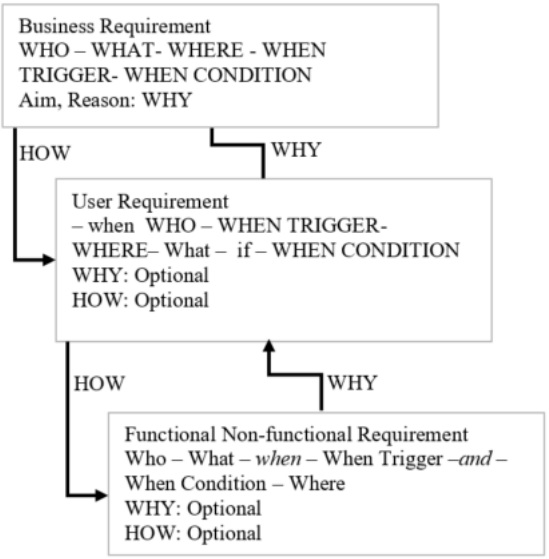
in the enterprise business architecture. "*Why", "Who", and "What"* dimensions are mandatory fields in the business requirement. Business requirements can be written in the following template.

WHO – WHAT- WHERE - WHEN TRIGGER- WHEN CONDITION

User requirements are the core of software requirement specification. They represent the end-user expectations from the software. Functional requirements lie at the bottom of user requirements and explain how the system works to sustain user requirements. "*How"* dimension comprises free text explanations, such as diagrams or mock-ups, showing the technical steps that need to be taken by the developer, such as creating a new table, or adding a new field in the screen.

The requirement cube approach manages non-functional requirements such as security and performance requirements as an attribute in the requirement. Each requirement (user or functional level) can be categorized as UI, Task, Business Process, Security, or a Performance requirement etc.

Figure 1. Requirement Cube Template Illustration



The user requirement structure for Turkish and English is given below. Words in italic are fixed conjunctions.

The structure of the user requirement for Turkish:

WHO – WHERE – WHEN TRIGGER – *if* – WHEN CONDITION – WHAT

The structure of the user requirement for English:

– *when* WHO – WHEN TRIGGER – WHERE– What – *if* – WHEN CONDITION

The detailed rule set is given in the Appendix A for English.

The structures of functional and user requirements are the same for Turkish. However, the structure in English is revised.

WHO – WHAT – *when* – WHEN TRIGGER –*and* – WHEN CONDITION – WHERE

Case Study

A case study was carried out in Kuveyt Turk Participation Bank, which is the 10th largest bank in Turkey and the number 1 in Islamic banking, with more than 400 branches. The Bank uses its own in-house developed banking system, namely BOA, and has two official R&D centers. The Information Technology department employs around 500 people. A case study was conducted to measure the effectiveness of the 5W1H requirement cube template.

Our research questions are:

Hypothesis One: Does the 5W1H requirement template guide BAs to write clear and distinctive requirements compared to the current approach?

Hypothesis Two: Does the 5W1H requirement template improve the quality of the requirements in the SRS document?

Three SRS documents were used in the case study. We have reformed and rewritten these three documents based on the 5W1H principles and have sent six documents to 17 evaluators and asked them to score them based on nine attributes. 5W1H documents achieved better results. The study was carried out in Turkish with the Turkish requirement cube template.

All three SRS documents belong to different business domains and they were written by different BAs. Personnel Relative Identification and Listing is human-resources-related software development. Stock Exchange Integration is related to a web service integration to buy and sell stocks online. Courier Management is about courier operations in branchless banking. All descriptive statistics are summarized in Table 1.

The current SRS format is quite permissive and BAs structure the requirements based on their individual experiences and choices. The document format consists of fixed titles, and BAs fill each title with free-text statements. One SRS document did not have distinctive requirements and all requirements were explained in mixed long texts. One of them had three simple requirements. One had 13 requirements, but they were defined with long and complicated wordings. After the 5W1H template was applied the number of requirements increased and requirements became distinctive. Besides, the amount of free-text statements was reduced drastically in two SRS documents. Therefore, it can be stated that the 5W1H requirement template guides BAs to write requirements distinctively.

Table 1. Descriptive Statistics of the Case Content

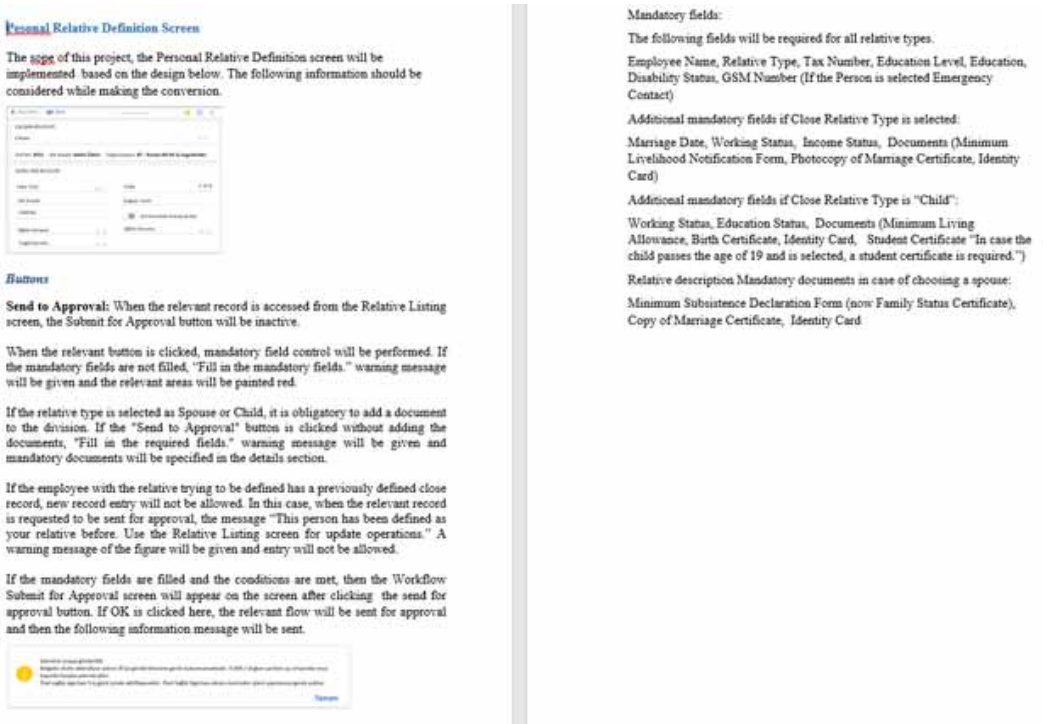
	Courier Management		**HR Personal Relative Definition**		**Stock Exchange Integration**	
	Current	5W1H	Current	5W1H	Current	5W1H
Number of requirements	3	28	0	47	13	52
Number of user requirements	0	9	0	34	13	15
Number of functional requirements	0	19	0	7	0	37
Number of non-functional requirements	0	0	0	6	0	0
Number of words	1385	1642	2545	1554	2796	2567
Average word per requirement	8	49.5	NA	20.11	70.92	13.09
Average word per user requirement	0	40.3	NA	19.59	70.92	13.13
Average word per functional requirement	0	53.8	NA	23.86	NA	9.3
Average word per non-functional requirement	0	0	NA	18.67	0	NA
Number of words as free text	1360	104	2406	373	1874	1886

This example from the case study is translated to English. Fig.2 presents the earlier condition of HR Personal Relative Definition for the "Send to Approval" button. Fig.3 shows the status after the 5W1H template was applied.

Our former approach is based on screen names, and buttons are listed under the screen title. Requirements are written in free text and it is hard to clarify how many requirements exist. Therefore, additional effort is necessary to write the test cases. This portion of the SRS has 389 words. Mock-ups, business needs, and functional requirements are written informally and business objective is missing.

However, the 5W1H requirement cube comprises a more sound SRS document. Business function is used as the title instead of the screen name. The business functions come from the enterprise architecture, and screen names are mentioned in the "where" portion in the requirement statement. The SRS documentation is more brief (only 260 words) and well-organized. Business, user, and functional requirements are clearly stated, and system design details are written under the requirement statements in the "How" dimension, along with the mock-ups.

Figure 2. Former Approach


Personal Relative Definition Screen

The scope of this project, the Personal Relative Definition screen will be implemented based on the design below. The following information should be considered while making the conversion.

Buttons

Send to Approval: When the relevant record is accessed from the Relative Listing screen, the Submit for Approval button will be inactive.

When the relevant button is clicked, mandatory field control will be performed. If the mandatory fields are not filled, "Fill in the mandatory fields." warning message will be given and the relevant areas will be painted red.

If the relative type is selected as Spouse or Child, it is obligatory to add a document to the division. If the "Send to Approval" button is clicked without adding the documents, "Fill in the required fields." warning message will be given and mandatory documents will be specified in the details section.

If the employee with the relative trying to be defined has a previously defined close record, new record entry will not be allowed. In this case, when the relevant record is requested to be sent for approval, the message "This person has been defined as your relative before. Use the Relative Listing screen for update operations." A warning message of the figure will be given and entry will not be allowed.

If the mandatory fields are filled and the conditions are met, then the Workflow Submit for Approval screen will appear on the screen after clicking the send for approval button. If OK is clicked here, the relevant flow will be sent for approval and then the following information message will be sent.

Mandatory fields:

The following fields will be required for all relative types.

Employee Name, Relative Type, Tax Number, Education Level, Education, Disability Status, GSM Number (If the Person is selected Emergency Contact)

Additional mandatory fields if Close Relative Type is selected:

Marriage Date, Working Status, Income Status, Documents (Minimum Livelihood Notification Form, Photocopy of Marriage Certificate, Identity Card)

Additional mandatory fields if Close Relative Type is "Child":

Working Status, Education Status, Documents (Minimum Living Allowance, Birth Certificate, Identity Card, Student Certificate "In case the child passes the age of 19 and is selected, a student certificate is required.")

Relative description Mandatory documents in case of choosing a spouse:

Minimum Subsistence Declaration Form (now Family Status Certificate), Copy of Marriage Certificate, Identity Card

We have sent the current and 5W1H versions of the SRS documents to 17 people within the organization. Participants consist of four analysts, five developers, two business members, two testers, two managers, and two auditors. We asked participants to fill out a questionnaire of nine questions regarding the overall quality of the requirements. The content of the questionnaire can be found below. Questions were determined based on the good quality attributes found in Mavin et al (2009) and Silva et al (2017).

· Requirements are clear and understandable. (Clarity)

· Requirements include relevant information regarding the development. There are no irrelevant statements. (Relevance)
· Requirements are consistent with each other. (Consistency)
· Requirements are implementable. (Implementability)
· Requirements are testable. (Testability)
· Requirements state the need completely. (Completeness)
· Requirements have no duplication. (No Duplications)
· Requirements do not include overstatement. (No Wordiness)
· How the system will operate and how it will be developed is stated separately. (Well Organized)

Participants picked an answer for each attribute from a nine-point scale, from extremely unsatisfactory as the minimum to extremely satisfactory as the maximum level. It was not possible to compare among requirements because they were not distinctive and were written in long free text in the former SRS format. Thus, participants made their evaluations document by document. Table 2 shows the descriptive results of the evaluations.

Results show that average scores have increased by 35%, 10%, and 18%, respectively, in all three cases of 5W1H. Another valuable outcome of the template is homogeneity. The scores of the former SRS documents vary between 5.31 and 6.45. However, the range of SRS documents with the 5W1H template is 7.04 to 7.2. Thus, it can be asserted that the 5W1H requirement cube template brings homogenous and good quality results independent from business domains and the BAs who prepared the document.

Each attribute's scores are presented in Fig. 4 for the 5W1H and the former approach, showing that the 5W1H template is more preferable in every aspect. Testability gets the highest score, while the clearance score is the lowest, which is also the lowest for the former approach.

Regarding professions, the 5W1H approach is mostly favored by managers and followed by the BAs. As potential writers of the new format, high scores from BAs are promising. Additionally, as potential implementers, the developers are also critical and their score has increased by 25%. Not only the IT side but also auditors and business units appreciate the new method. Test engineers believe that there is no such difference between the two approaches although the testability question gets the highest score. More senior employees tend to give higher scores for the 5W1H template. Females prefer the 5W1H approach more than males. Overall, the 5W1H template achieved a 20% better result on average, compared to the current approach. Table 3 presents more details.

The major weakness in this case study is the non-functional requirements. They are supposed to be the hardest requirements to find, as the business side rarely mentions them. The documents used in the case study have no non-functional requirements. During the case study, the team came up with some non-functional requirements and was able to write in 5W1H format. However, the 5W1H template does not guide the BAs for non-functional requirements unlike user and functional requirements. Hence, a template extension is useful in detecting non-functional requirements better.

There are validity risks associated with empirical studies that affect the trustworthiness of the results. The first risk in our case study was creating 5W1H documents with extra quality. We used past SRSs as they were and re-wrote them in 5W1H form. During this transformation, we might have given more and specific attention to the 5W1H SRSs. Usually, BAs may complain that they do not have enough time to prepare SRSs with extra care due to daily emergencies, meetings, and other daily work requirements. Therefore, we prepared 5W1H versions during their daily job routine as an extra task. Spelling mistakes and misconstrued sentences were left as they were. 14 of 17 reviewers had no prior information regarding the 5W1H approach. The logic of the template was not explained to them in detail. On the contrary, they just saw the fully constructed sentences. Before the case study, there was no face-to-face meeting due to COVID-19 precautions. Instead, short videos were prepared to explain what was expected from the study. We believe this distant approach had a positive effect on validity because we did not create a certain expectation that might affect results in favor of 5W1H.

Figure 3. After the 5W1H Template

Send to Approval

1. Any Bank staff should be able to send relative information and documents to HR.
   Why: Marital status and number of kids of the staff are required to calculate tax properly.

   1.1. When Any Bank Staff clicks the Send for Approval action on the Relative Identification screen, if the mandatory fields are filled, the HR approval flow shall be started.

   How:
   The Workflow Submit for Approval screen will opened. If OK is clicked here, the relevant flow will be sent for approval and then the following information message will be sent.

   1.1.1. If the mandatory fields are not filled, a warning message shall be given to the user, and the required fields shall be underlined with red.

   How:
   Employee Name, Relative Type, Tax Number, Education Level, Education, Disability Status, GSM Number (If the Person is selected Emergency Contact)

   1.1.2. If relative type is "Spouse" or "Child", additional fields and shall become mandatory related documents shall be added.

   if Type is "Spouse"
   Marriage Date, Working Status, Income Status fields shall be mandatory.
   Minimum Livelihood Notification Form, Photocopy of Marriage Certificate, Identity Card documents have to be added

   if Type is "Child"
   Working Status, Education Status fields shall be mandatory.
   Minimum Living Allowance, Birth Certificate, Identity Card, Student Certificate documents have to be added

   1.2. When Any Bank Staff clicks the Send for Approval action on the Relative Identification screen, if the relative is defined already, a message shall be given saying This person has been defined as your relative before. Use the Relative Listing screen for update operations."

Table 2. Descriptive Statistics of Evaluations

Project Name	Courier Management		Personnel Relative Identification & Listing		Stock Agency Integration		Overall	
	Current SRS	5W1H SRS	Current SRS	5W1H SRS	Current SRS	5W1H SRS	Current SRS	5W1H SRS
Mean	5.31	7.20	6.45	7.08	5.94	7.04	**5.90**	**7.10**
Median	5.00	7.00	7.00	7.00	6.00	7.00	**6.00**	**7.33**
Mode	5.00	7.00	7.00	7.00	6.00	7.00	**6.00**	**7.00**
Standard Deviation	1.51	1.06	1.22	1.62	1.31	1.16	**0.91**	**1.07**
Minimum	1.00	3.00	2.00	1.00	1.00	3.00	**2.67**	**3.67**
Maximum	8.00	9.00	8.00	9.00	8.00	9.00	**8.00**	**9.00**

Figure 4. Average Score by Evaluation Item

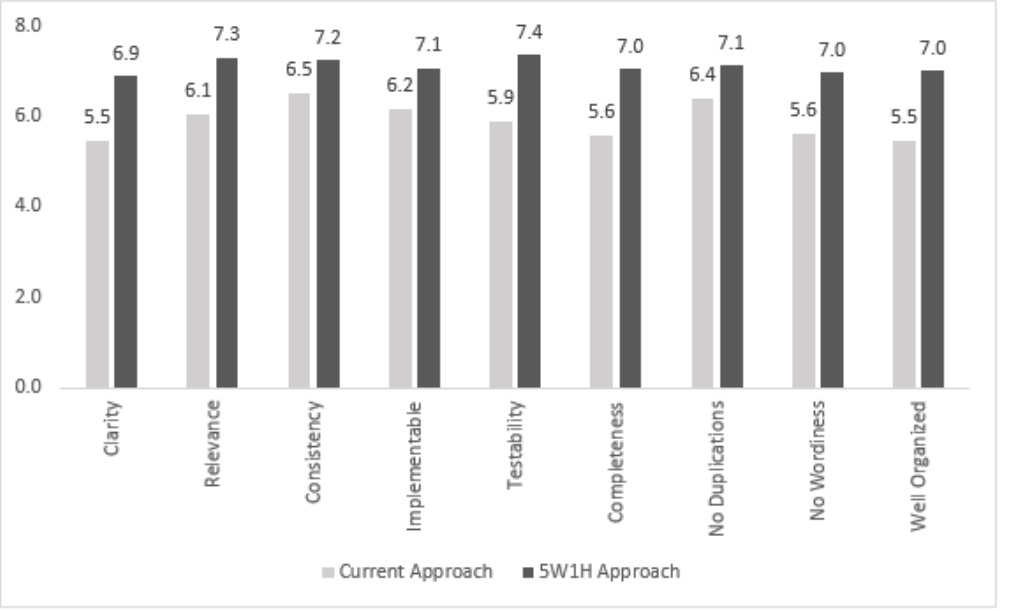


Table 3. Evaluations by Demography

Indicator	Groups	Frequency	Ratio	Current Approach (Average Score)	5W1H Approach (Average Score)
Job Title	Analyst	4	24%	6.16	7.36
	Auditor	2	12%	5.72	6.85
	Business Unit	2	12%	6.44	7.17
	Developer	5	29%	5.65	7.07
	Manager	2	12%	5.31	7.69
	Test Engineer	2	12%	6.22	6.28
Experience	< 5	4	24%	5.81	6.76
(years)	5 - 10	11	65%	6.02	7.21
	> 10	2	12%	5.43	7.2
Gender	Male	11	65%	5.87	6.97
	Female	6	35%	5.96	7.36
Number of Attendees		**17**	**100%**		

CONCLUSION

Requirements engineering is more a process rather than a set of activities for gathering and specifying different requirements' levels. Requirements engineering can be divided into three sub-processes: requirement definition, requirement specification, and software specification. There is a vast amount of methods and techniques in RE but they are mostly specific to a sub-process. Therefore, it is hard to carry on the "translation of business needs'' to software specifications holistically and smoothly. This study proposes a requirement template that can be used in all parts of the RE. We utilize the 5W1H method–a common technique in RE–to set up a hierarchical requirement template for business, user, and functional requirements. The 5W1H requirement cube stores necessary elements of a good requirement in "Why", "Who", "When", "Where", "What", and "How" dimensions. It gathers and links all types of requirements under the enterprise business architecture and composes a repository.

A case study was conducted to find if the 5W1H requirement template

- supports BAs to write clear and distinctive requirements compare to current approach
- improves the quality of the requirements in the SRS documents.

Three SRS documents were rewritten with the 5W1H template and sent to 17 people in the organization with a questionnaire consisting of nine questions. Results show that BAs wrote clearer and distinctive requirements and that of SRS documents' quality improved with the 5W1H template. However, the 5W1H template doesn't guide the BAs for the non-functional requirements and an extension is necessary to elicit the non-functional requirements.

Writing all three types of requirements in the 5W1H template solves the requirement transition problems moving from high level to detail level in the RE and gives a solid account of writing software requirement specifications. 5W1H requirement cube proposes the same template for business, user, and functional requirements. In other words, it helps shareholders speak a common language during the

RE process and software development lifecycle. Besides, everything is kept linked and it is possible to follow 'what' to do and 'why' to do.

All requirement writing methods have a trade-off between requirement's precision/clarity and the convenience of the method. The clearer the requirement, the harder it is to write. The 5W1H requirement cube offers a reasonable balance between the writing effort and the statement's quality. Consequently, we believe that the 5W1H requirement template is a promising candidate that can be applied in the industry.

REFERENCES

Ali, N., & Lai, R. (2017). A method of requirements elicitation and analysis for Global Software Development. *J Softw Evol Process.*, *29*(4), e1830. doi:10.1002/smr.1830

Alrumaih, H., Mirza, A., & Alsalamah, H. (2018, April). Toward automated software requirements classification. In *2018 21st Saudi Computer Society National Computer Conference (NCC)* (pp. 1-6). IEEE. doi:10.1109/NCG.2018.8593012

Awal, A., Gana, U. M., & Abdulrahman, A. (2018). *Ontology Development for the Domain of Software Requirement Elicitation Technique Development of Framework and an Application Model for Positive Contribution to Teaching View project Ontology Development for the Domain of Software Requirement Elicitation*. Artic Int J Eng Tech Res. doi:10.17577/IJERTV7IS040237

Carmagnola, F., Cena, F., & Gena, C. (2011). User model interoperability: A survey. *User Modeling and User-Adapted Interaction*, *21*(3), 285–331. doi:10.1007/s11257-011-9097-5

Carrizo, D., Dieste, O., & Juristo, N. (2014). Systematizing requirements elicitation technique selection. *Information and Software Technology*, *56*(6), 644–669. doi:10.1016/j.infsof.2014.01.009

Ciccozzi, F., Malavolta, I., & Selic, B. (2019). Execution of UML models: A systematic review of research and practice. *Software & Systems Modeling*, *18*(3), 2313–2360. doi:10.1007/s10270-018-0675-4

Cockburn, A. (1999). *Writing effective use cases*. Addison-Wesley Longman.

Davey, B., & Parker, K. R. (2015). *Requirements Elicitation Problems: A Literature Analysis* (Vol. 12). http://iisit.org/Vol12/IISITv12p071-

Fabrini, F., Fusani, M., Gnesi, G., & Lami, G. (2000). Quality evolution of software requirements specifications. Proceedings of Software and Internet Quality Week.

Franch, X. (2015). Software requirements patterns - A state of the art and the practice. In *Proceedings - International Conference on Software Engineering* (Vol. 2). IEEE Computer Society. doi:10.1109/ICSE.2015.298

Franch, X., Palomares, C., Quer, C., Renault, S., & De Lazzer, F. (2010). A metamodel for software requirement patterns. Lect Notes Comput Sci, 6182, 85-90. doi:10.1007/978-3-642-14192-8_10

García-López, D., Segura-Morales, M., & Loza-Aguirre, E. (2020). Improving the quality and quantity of functional and non-functional requirements obtained during requirements elicitation stage for the development of e-commerce mobile applications: An alternative reference process model. *IET Software*, *14*(2), 148–158. doi:10.1049/iet-sen.2018.5443

Glinz, M. (2011). A glossary of requirements engineering terminology. Standard Glossary of the Certified Professional for Requirements Engineering (CPRE) Studies and Exam. *Version*, *1*, 56.

Greenspan, S., Mylopoulos, J., & Borgida, A. (1994). *On formal requirements modeling languages: RML Revisited Invited Plenary Talk*. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.92.4047&rep=rep1&type=pdf

Issa, A. A., & Al-Ali, A. (2010). Use case patterns driven requirements engineering. *2nd Int Conf Comput Res Dev ICCRD 2010*, 307-313. doi:10.1109/ICCRD.2010.16

Jabar, M. A., Ahmadi, R., Shafazand, M. Y., Ghani, A. A. A., Sidi, F., & Hasan, S. (2013). An automated method for requirement determination and structuring based on 5W1H elements. *Proceedings - 2013 IEEE 4th Control and System Graduate Research Colloquium, ICSGRC 2013*, 32-37. doi:10.1109/ICSGRC.2013.6653271

Jacobson, I. (2004). Use cases – Yesterday, today, and tomorrow. *Software & Systems Modeling*, *3*(3), 210–220. doi:10.1007/s10270-004-0060-3

Kudo, T. N., Bulcão-Neto, R. F., & Vincenzi, A. M. R. (2020). Requirement patterns: A tertiary study and a research agenda. *IET Software*, *14*(1), 18–26. doi:10.1049/iet-sen.2019.0016

Lauesen, S. (2002). *Software requirements: styles and techniques*. Pearson Education.

Lethbridge, T. C., & Laganière, R. (2005). *Object-oriented software engineering: Practical software development using UML and Java*. McGraw-Hill Education.

Luisa, M., Mariangela, F., & Pierluigi, N. I. (2004). Market research for requirements analysis using linguistic tools. *Requir Eng.*, *9*(1), 40–56. doi:10.1007/s00766-003-0179-8

Mavin, A., Wilkinson, P., Harwood, A., & Novak, M. (2009) EARS (Easy Approach to Requirements Syntax). *Proceedings of the IEEE International Conference on Requirements Engineering*, 317-322. doi:10.1109/RE.2009.9

Pacheco, C. L., Garcia, I. A., Calvo-Manzano, J. A., & Arcilla, M. (2015). A proposed model for reuse of software requirements in requirements catalog. *J Softw Evol Process.*, *27*(1), 1–21. doi:10.1002/smr.1698

Palomares, C., Quer, C., & Franch, X. (2014). Requirements Reuse with the PABRE Framework. *Requir Eng Mag., 1*. https://www.researchgate.net/publication/260730928

Palomares, C., Quer, C., & Franch, X. (2017). Requirements reuse and requirement patterns: A state of the practice survey. *Empirical Software Engineering*, *22*(6), 2719–2762. doi:10.1007/s10664-016-9485-x

Pohl, K. (1993). *The three dimensions of requirements engineering*. Springer. doi:10.1007/3-540-56777-1_15

Rasmussen, J., Pejtersen, A. M., & Goodstein, L. P. (1994). *Cognitive systems engineering*. Academic Press.

Siau, K. (2016). *Identifying difficulties in learning UML supply chain management view project*. 10.1201/1078.10580530/46108.23.3.20060601/93706.5

Silva, A., Pinheiro, P. R., Albuquerque, A., & Barroso, J. (2017). Evaluation of an approach to define elicitation guides of non-functional requirements. *IET Software*, *11*(5), 221–228. doi:10.1049/iet-sen.2016.0302

Silva, A., Pinheiro, P. R., Albuquerque, A. B., & Barroso, J. (2016). A Process for Creating the Elicitation Guide of Non-functional Requirements In Computer Science. In *On-line Conference* (pp. 293-302). Springer. doi:10.1007/978-3-319-33622-0_27

Sommerville, I. (2011). Software engineering (9th ed.). Academic Press.

Souag, A., Mazo, R., Salinesi, C., & Comyn-Wattiau, I. (2018). Using the AMAN-DA method to generate security requirements: A case study in the maritime domain. *Requir Eng.*, *23*(4), 557–580. doi:10.1007/s00766-017-0279-5

Tiwari, S., & Gupta, A. (2020). Use case specifications: How complete are they? *J Softw Evol Process*, *32*(1), 9–11. doi:10.1002/smr.2218

Toroa, D., Jiménez, B.B., Cortés, R., & Bonilla, M.T. (1999). A Requirements Elicitation Approach Based in Templates and Patterns ? *Requir Eng.,* 17-29.

Toval, A., Nicolás, J., Moros, B., & García, F. (2002). Requirements reuse for improving information systems security: A practitioner's approach. *Requir Eng.*, *6*(4), 205–219. doi:10.1007/PL00010360

Umber, A., & Bajwa, I. S. (2011). Minimizing ambiguity in natural language software requirements specification. *2011 6th International Conference on Digital Information Management, ICDIM*, 102-107. doi:10.1109/ICDIM.2011.6093363

Walia, G. S., & Carver, J. C. (2009). A systematic literature review to identify and classify software requirement errors. *Information and Software Technology*, *51*(7), 1087–1109. doi:10.1016/j.infsof.2009.01.004

Westfall, L. (2005). *Software requirements engineering: What, why, who, when, and how*. http://www.westfallteam.com/Papers/The_Why_What_Who_When_and_How_Of_Software_Requirements.pdf

Wiegers, K., & Beatty, J. (2013). *Software Requirements*. Microsoft Press.

Withall, S. (2007). *Software requirement patterns*. Microsoft Press. www.microsoft.com/mspress

Yue, L., Niu, P., Wang, Y., & Yue, L. (2019). *Guidelines for defining user requirement specifications (URS) of manufacturing execution system (MES) based on ISA-95 standard*. 10.1088/1742-6596/1168/3/032065

Zachman, J. A. (1987). A framework for information systems architecture. *IBM Systems Journal*, *26*(3), 276–292. doi:10.1147/sj.263.0276

Zowghi, D., & Coulin, C. (2005). Requirements elicitation: A survey of techniques, approaches, and tools. In *Engineering and Managing Software Requirements* (pp. 19–46). Springer Berlin Heidelberg. doi:10.1007/3-540-28244-0_2

APPENDIX A - 5W1H RULES

Business Requirement Writing Rules:

Why: Mandatory. Free text explanation of the business goal.
Who: Mandatory. Free text or categorized selection of the actors of the requirement.
Where: Optional. Can be a channel name, system name, business process name
When Trigger: Mandatory. If there is no special case it has to be written "anytime" as *when trigger*. The actions in the *when trigger* has to be written in active tense.
When Condition: Optional. It starts with "if" before writing the related condition. Conditions that will lead to different results have to be written as a separate requirement.
What: Mandatory. The basic definition of the requirement.

User Requirement Writing Rules:

Why: Not mandatory, but recommended. It should include the goal of the requirement such as cost reduction or compliance with regulations, etc.
Who: Mandatory. It includes user profile roles, customer types, and automated tasks that carry on actions on behalf of actual users. It is recommended to include an adverb like "All", "Only", "Except"
Example: Individual Customers, All Customers, Supervisors and Above, Only Managers, All users except Customer Representatives.
When Trigger: Mandatory. If there is no special case, "always" should be written as the *when trigger*. The actions in the *when trigger* have to be written in the active tense.
Where: Mandatory. It can be a channel, a display screen, or a device. Mobile App, ATM, Expense Entry Display Screen, etc.
When Condition: Optional. It starts with "if", followed by the related condition. Conditions that lead to different results have to be written as a separate requirement.
What: Mandatory. It will be written with a "shall" statement. It can be in passive form if the *when trigger* is an active statement. It can include multiple actions as long as all actions are tied with each other with "and". "Or" statements are not allowed and have to be written as a separate requirement.

Functional Requirement Writing Rules:

Why: Not mandatory. The goal is assumed as the related user requirement if it is null. Specific reasons unique to a functional requirement can be written in the *Why* portion.
Who: Not Mandatory. It can be a system object, a task, workflow, process, etc.
What: Mandatory. It will be written with a "shall" statement. Active tense is recommended but can be passive. It can include multiple actions as long as all actions are tied with each other with "and". "Or" statements are not allowed and have to be written as a separate requirement.
When Trigger: Mandatory. If there is no special case, "always" has to be written. The actions can be written in an active or passive voice.
When Condition: Optional. It starts with "and", followed by a related condition. Conditions that lead to different results have to be written as a separate requirement.
Where: Optional. It can be a channel, a display screen, or a device. If it is empty, it will be perceived the same as the "*where"* portion of the related user requirement

Yasar Ugur Pabuccu got his bachelor's degree as an Industrial engineer in 2000. He received MS in Information Systems and completed his Ph.D. in Business Administration. He works as an Enterprise Architect in Kuveyt Turk Participation Bank Information Sytems and is responsible for internal systems regarding demand and portfolio management, business process management, software system analysis and testing.