Vol. 16, No. 3, Julio-Septiembre, 2022
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 137-156

Tipo de artículo: Artículo original
Temática: Ingeniería y gestión de software
Recibido: 21/06/2022 | Aceptado: 01/07/2022

Toward to a template for syntactic specification of functional requirements for embedded systems

Aproximación a una plantilla para la especificación sintáctica de requisitos funcionales de sistemas embebidos

Yosanne Garzón Mendoza [1] https://orcid.org/0000-0001-9395-4775
Dunia Ma. Colomé Cedeño [2*] https://orcid.org/0000-0002-3184-5984
Jessenia Guadalupe Chalen Ortega [3] https://orcid.org/0000-0002-9371-8614
Yorgelys González López [4] https://orcid.org/0000-0001-8447-0760

[1] Universidad de las Ciencias Informáticas. Carretera a San Antonio de los Baños, Km 2 ½, reparto Torrens, municipio Boyeros, La Habana, Cuba. CP: 19370.

[2] Universidad de las Ciencias Informáticas. Carretera a San Antonio de los Baños, Km 2 ½, reparto Torrens, municipio Boyeros, La Habana, Cuba. CP: 19370. dcolome@uci.cu

[3] Universidad Católica de Santiago de Guayaquil. Av. Carlos Julio Arosemena Km 1 ½. Guayaquil. Ecuador. jessenia.chalen@cu.ucsg.edu.ec

[4] Universidad de las Ciencias Informáticas. Carretera a San Antonio de los Baños, Km 2 ½, reparto Torrens, municipio Boyeros, La Habana, Cuba. CP: 19370. yorgelysg@uci.cu

*Autor para la correspondencia. (dcolome@uci.cu)

Vol. 16, No. 3, Julio-Septiembre, 2022
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 137-156

ABSTRACT

Natural language is an essential means of writing system requirements, among other reasons, because of its high expressiveness and flexibility. Despite its indisputable advantages, in requirements documentation in the natural language, it is common to find syntactic ambiguity, which occurs when more than one grammar rule represents a sentence. Templates are a proven means of writing well-structured functional requirements in natural language and mitigating some of the weaknesses of this language for technical writing. Different templates have been formulated to specification requirements for almost any type of system, thanks to a generic design. However, these templates have insufficiencies in their structure that make it difficult to formulate functional requirements for embedded systems. The objective of this paper is to propose a template for the syntactic specification of functional requirements of embedded systems using natural language. The proposed template took into account the study of four existing templates and the distinctive characteristics of embedded systems. The methods and techniques used are analytical-synthetic, documentary analysis, and modeling. The validation was carried out through the IADOV technique, resulting in a group satisfaction index of 0.81 on the proposed template.

Keywords: syntactic ambiguity; template; functional requirement; embedded system.

RESUMEN

El lenguaje natural es un medio esencial para escribir requisitos de los sistemas, entre otras razones por su alta expresividad y flexibilidad. A pesar de sus indiscutibles ventajas, en la documentación de requisitos en lenguaje natural es común encontrar la ambigüedad sintáctica, la que se presenta cuando más de una regla gramatical representa a una oración. El uso de plantillas es un medio probado para escribir requisitos funcionales bien estructurados en lenguaje natural y, por lo tanto, mitigar algunas de las debilidades de este lenguaje para la redacción técnica. Diferentes plantillas se han formulado, algunas de ellas con un diseño genérico que posibilitan la especificación de requisitos de casi cualquier tipo de sistema, sin embargo, se aprecian insuficiencias en su estructura que dificultan la formulación de requisitos funcionales de sistemas embebidos. El objetivo de este trabajo es proponer una plantilla para la especificación sintáctica de

Vol. 16, No. 3, Julio-Septiembre, 2022
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 137-156

requisitos funcionales de sistemas embebidos, empleando el lenguaje natural. La propuesta se basa en el estudio de cuatro plantillas existentes y en las características distintivas de los sistemas embebidos. Entre los métodos y técnicas empleadas en el desarrollo de este trabajo se encuentran el analítico-sintético, el análisis documental y la modelación. La validación de la plantilla propuesta se realizó a través de la aplicación de la técnica de IADOV, resultando un índice de satisfacción grupal de 0.81 sobre la plantilla propuesta.

Palabras clave: ambigüedad sintáctica; plantilla; requisito funcional; sistema embebido.

Introduction

Most electrical products include a computer and control software; moreover, it is an international practice that industrial manufacturing and distribution are fully computerized. Therefore, software engineering plays an important role in the economic and social development of countries since it is concerned with all aspects of systems production, from the early stages of system specification to its maintenance.

According to Bauer (1975), software engineering is "the establishment and use of fundamental engineering principles to economically develop software that is reliable and works efficiently on real machines." Pressman (2010), concludes that software engineering is a technology composed of several layers, which are: tools, methods, processes, and commitment to quality.

Requirements engineering is one of the activities of software engineering and constitutes a systematic and disciplined approach to requirements specification and management (Ambreenet al., 2018), (Glinz, 2017), (Raikar and Cholli, 2021). Among its activities is requirements documentation.

Requirements documentation can be done using Natural Language (NL) (Ferrari, 2018), (Glinzet al., 2020), (Zhaoet al., 2020), (Sonbol, 2020), (Liet al., 2022), using conceptual models, such as use case diagrams, class diagrams, etc., or combining both forms. Pohl (2010), Alhoshanet al., (2019) argue that LN is the most common way of communicating and documenting the requirements of a system and that it does not require any special training in the interpretation of notations or symbols as is the case when using an engineering

language such as the Unified Modeling Language (UML). However, this language has some disadvantages (Osama et al., 2020) (Ferrari et.al, 2021) (Mavin et al., 2009) among which are ambiguity, vagueness, complexity, omission, duplication, verbosity, implementation, and lack of testability.

To reduce the effects of the NL in the formulation of functional requirements, different requirements templates have been designed for the construction of requirements that are easy to learn and apply. Some of these templates have a generic design to specific requirements for any type of system. However, these templates have insufficiencies in their structure that make it difficult to formulate functional requirements for embedded systems.

An embedded system is an information processing system for specific use integrated into another larger system and made up of hardware and software components (Marwedel, 2021). Embedded software is important from an economic point of view because now, almost all electrical devices include software. Consequently, there are many more embedded software systems than any other type.

Regarding these systems, Sommerville (2011) argues that: "they must interface with a wide range of hardware devices that do not have separate device drivers." In addition, this author refers that the design process for embedded systems is a systems engineering process in which software designers must consider, in detail, the design and performance of the system's hardware. Part of the system design process may involve deciding which system capabilities will be implemented in software and which ones in hardware.

In these systems, the requirements engineering processes become complicated to the point that in typical application areas, more than 50% of the problems occur because the system does not meet the user's expectations due to the erroneous capture of information the requirements (Chenget al., 2006).

Existing solutions in the field of software engineering for the development of embedded systems are very design-oriented, almost completely ignoring the requirements capture and analysis phase. A requirements engineering methodology designed for this domain must allow the representation of requirements at the system level, that is, those that express functionalities; in addition, it must capture requirements at external levels, where are found the interfaces with the super-system and other embedded systems. The purpose of this paper is to propose a template for the syntactic specification of functional requirements of embedded systems using the NL.

Vol. 16, No. 3, Julio-Septiembre, 2022
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 137-156

Methodology

The development of this research followed these steps: 1) Characterization of embedded systems, highlighting the main elements of the design of this type of system, 2) Study of requirements engineering methodologies with application in different domains, 3) Analysis of the behavior of syntactic ambiguity in software requirements, 4) Analysis of templates for the selection of the structure of functional requirements of software, 5) Design of a template for the syntactic structure of functional requirements of embedded systems, 6) Validation of the template proposal. To guide the investigation, two research questions are presented: a) What are the main characteristics of embedded systems? b) What are the main solutions to minimize syntactic ambiguity in software requirements?

In the study selection process, the following digital libraries were identified as data sources: the ACM Digital Library (ACM) and the IEEE Xplore Digital Library (Xplore). The search strategy consisted of a direct search in the aforementioned databases based on the following search criteria "syntactic ambiguity", "functional requirements" and "embedded systems". The research methods used were Documentary Analysis, Analysis-Synthesis and Modeling.

Results and Discussion

In addition to the above, regarding the design of embedded systems, it is worth mentioning that, for many real-time systems embedded in consumer products, such as systems in cell phones, the costs and power consumption of the hardware are critics. (Sunget al., 2022)

The design of a product that incorporates embedded systems is oriented to minimize costs and maximize reliability, it’s also essential to incorporate safety considerations into the design, including cryptographic functions and protocols that protect information during all phases. Embedded systems often operate in a

dedicated environment with very specific operational conditions and scenarios. These conditions and threats must be taken into account when designing security features.

Low-level decisions on hardware, supporting software, and system timing should be considered early in the process. The before limits the flexibility of system designers and may mean that additional software functionality, such as battery and power management, must be included in the system (Sommerville, 2011).

Since embedded systems are reactive systems that react to events in their environment, the most general approach to real-time embedded software design is based on a stimulus-response model. A stimulus is an event that occurs in the software system environment that causes the system to react in some way; a response is a signal or message that the software sends to its environment (Sommerville, 2011).

Therefore, the system architecture must be organized in a way that as soon as a stimulus is received, control is transferred to the correct handler, which is impractical in sequential programs. (Sommerville, 2011) Therefore, real-time software systems are designed, generally, as a set of concurrent cooperative processes. To support the management of said processes, the execution platform in which the real-time system is carried out may include a real-time operating system. The functions provided by this operating system are accessed through the runtime support system for the real-time programming language used.

There is no standard embedded system design process. Instead, different processes are used, depending on the type of system, the hardware available, and the organization developing the system. The following activities can be included in a real-time software design process (Sommerville, 2011): Platform selection, Identification of stimuli/responses, Timing analysis, Process design, Algorithm design, Data design, Process planning.

The order of these activities in the real-time software design process depends on the type of system to be developed, its process, and platform requirements.

Vol. 16, No. 3, Julio-Septiembre, 2022
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 137-156

Requirements engineering methodologies with application in different domains

There are several methodologies for requirements engineering, including the following: ScenIC (Pottset al., 1999), SCRAM (Sutcliffe, 2003), KAOS (Dardenneet al., 1993) (Leiteret al., 2002) y ABC-Besoins (Urrego, 2005).

ABC-Besoins, with the concept of "Intervention of agents" manages to integrate and relate the requirements of different contexts. In addition, ABC-Besoins proposes guidelines for the analysis of requirements from the capture phase to the specification phase, achieving the operationalization of said requirements and the construction of a conceptual model. Gonz and Giraldo (2008) state that all these advantages are due to an expressive requirements model that allows relating requirements at different levels and capturing the needs of the agents.

Gonz and Giraldo (2008) intervened in the ABC-Besoins methodology, which was designed for the domain of web systems; they adapted and transformed it to take into account aspects of the embedded systems domain and build a conceptual model that facilitates entry into a specification language such as SystemC.

The adaptation of the ABC-Besoins methodology to embedded systems led to the creation of the requirements model for this type of system, named by its authors ABC-Besoins-SEM. This model presents 13 categories with additional subcategories for the domain of embedded systems and is taken as a reference for the elaboration of the template that is presented in this work.

Syntactic ambiguity

In the specification of requirements in NL, there is a tendency to use a special language, which has characteristics that endow it with great richness; however, this language has barriers to communication due to its ambiguity. (Riaz et.al, 2019) (Gariglianoet al., 2018) (Jadallah et.al, 2021) (Vierlboeck et.al, 2022)
According to Koscinski et al., (2021), Dalpiaz et al., 2018), Mavin (2009), some of the problems that can appear in the specification of requirements in LN are ambiguity, vagueness, complexity, omission,

duplication, verbosity, implementation, and lack of capacity to be tested. To minimize misinterpretation in the writing requirements in NL, Sommerville (2011) recommends following some guidelines. Additionally, Wiegers and Beatty (2013) presented a list of terms that can be ambiguous and should be avoided in requirements specifications.

Ambiguity, in the linguistic process, can occur when it is possible to admit different interpretations of the representation of a sentence. Also, it occurs when there is confusion by having several structures associated with the same sentence. Different types of ambiguities may be present in software requirements, including lexical, syntactic, and semantic. (Augustoet al., 2009) (Wiegers and Beatty, 2013) (Zhaoet al., 2020)

Lexical ambiguity refers to single expressions that can be reasonably interpreted in more than one way. (Dalpiazet al., 2019) Syntactic ambiguity occurs when a sentence has more than one syntactic representation associated with it, that is when more than one grammatical rule represents that sentence. Semantic ambiguity occurs when a sentence has more than one way to read it within its context, even though it contains no lexical or syntactic ambiguity. (Kiyavitskayaet al., 2007) (Zapataet al., 2008)

Analytic ambiguity occurs when the constituent parts within a phrase or sentence are ambiguous. Attachment ambiguity occurs when a particular constituent part of a sentence, such as a prepositional phrase or a relative clause, can legally be attached to two parts of a sentence. Coordinative ambiguity can occur when a sentence contains more than one conjunction-type word; it can be copulative, disjunctive, or mixed. Elliptical ambiguity occurs when it is not known with certainty whether a sentence contains ellipses or not. Prepositional ambiguity can occur when a sentence contains a preposition-type word. (Kiyavitskayaet al., 2007) (Zapataet al., 2008) (Schneider and Arndt, 2013) One of the solutions to avoid the presence of this type of ambiguity in the software requirements is the use of a template to take into account during its elaboration.

Main templates for developing requirements in natural language

Templates are a proven resource for writing well-structured requirements and mitigating some of the weaknesses of natural language. (Glinzet al., 2020) Several authors have proposed templates for writing requirements in natural language, such as those presented below.

The template proposed by Rupp (2014), also known as MASTeR (Mustergultige Anforderungen - die SOPHIST Templates fur Requirements) was recommended by the IREB (International Requirements Engineering Institute) and has been accepted as a standard for the syntactic specification of system requirements. (Mazoet al., 2019).

The EARS syntax is a mechanism for smoothly constraining textual requirements. EARS patterns provide structured guidance that enables authors to write high-quality textual requirements. It is a template that considers various conditional patterns from which various requirements are typified. (Vallejoet al., 2014)
Adv-EARS (Advanced EARS) (Majumdaret al., 2011) proposes a semi-structured language syntax to specify functional requirements, in such a way that automated support is given for the derivation of use cases and actors in use case models. In Adv-EARS, the requirements are represented in natural language following the syntax defined in EARS (Mavinet al., 2009).

The template created by Mazo and Jaramillo (2019) is based mainly on the analysis of the Rupp template, improving it, and drawing inspiration from concepts from other related works (EARS). This template, is composed of eight spaces and each one was structured using a simple and at the same time robust syntactic specification to cover the largest number of types of requirements in various types of systems. (Mazoet al., 2019)

The study of these templates allowed us to know that they can be used in different types of systems for writing functional or non-functional requirements, which shows their usefulness. However, these present difficulties when it comes to embedded systems. Rupp's template lacks specification as to the "why" or "what for" of the requirement, and also does not allow the possibility of detailing ranges or quantities of objects. Research at EARS focuses on the aviation industry; Unlike it, this research aims to focus on

Vol. 16, No. 3, Julio-Septiembre, 2022
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 137-156

embedded systems in general, regardless of the application domain. Adv-EARS and Mazo-Jaramillo come quite close to a formulation of requirements in various systems, but in terms of embedded systems, they do not have timing, which is a characteristic element of real-time systems.

Template for the syntactical specification of functional requirements of embedded systems

Considering the problems found in each of the templates presented, the new template was developed, see Fig 1. It consists of five sections, each one structured with the simplest possible syntactic form in mind. Section A represents the conditionals, section B refers to the embedded system or the family of embedded systems that correspond to a super-system, a part or functionality of it; section C represents the degree of obligation; D contains the elements referring to agent interaction, agent action, and failure prevention and detection; finally, section E represents the timing criterion. In the case of A and E, these are highlighted in another color to indicate that their use may be optional.

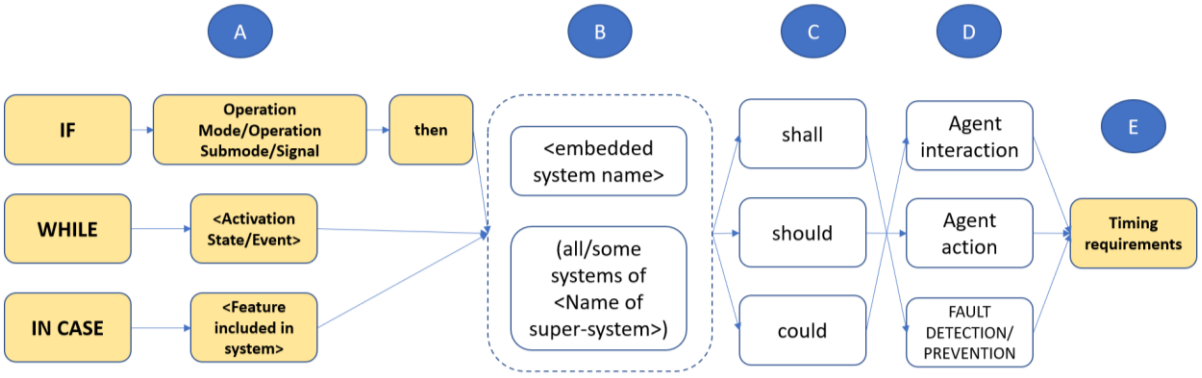

Fig. 1 – Template for functional requirements of embedded systems.

Here is an explanation of each of these sections:

A. Conditionals

1. Requirements with logical conditions.

Vol. 16, No. 3, Julio-Septiembre, 2022
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 137-156

IF *<Operation Mode/Operation Submode/Signal>* **THEN**, *[(all/some systems of <Supersystem Name>) | (the <embedded system name>)]* shall/should/could…

An operation mode is a set of system functionalities that are activated or deactivated with the occurrence of an event, sending a signal, and subsequent entry into an operation mode. Typical operating modes of an embedded system are: on, and off.

Through the hierarchy, each mode of operation can be refined into submodes. For example, if the embedded system is in power-on mode, it can in turn be in one of the following sub-modes: in configuration, in normal operation, in maintenance.

A signal communicates the occurrence of an event that generates consequences such as a change of state. Represents a necessary condition for a process to execute.

2. State Guided Requirements

WHILE (o**r DURING**) *<Activation State/Event> [(all/some systems of <Super-System Name>) | (the <embedded system name>)] shall/should/could…*

The state of these systems can correspond to modes and sub-modes of operation.

3. Requirements with an optional feature

IN CASE *<Feature included in system> [(all/some systems of <Name of super-system>) | (the <embedded system name>)] shall/should/could…*

These requirements are used when a particular characteristic is included when describing some behavior.

Here are some examples of requirements with conditionals, using the proposed template:

If it is in active mode, normal operation submode and the power cable is interrupted, then the system must take the impedance value at the Q1 output and enter the high impedance submode.

In case the temperature exceeds 30 degrees the system should automatically open the windows.

As long as two or more positive sensors are found, the system should start the alarm.

Vol. 16, No. 3, Julio-Septiembre, 2022
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 137-156

B. Family of systems, super-system or embedded system

In this case, it changed what was *[(all/some systems in the <Product Line Name>) | (the <Name of the system or part)]* of the Mazo and Jaramillo template by *[(all/some systems of <Name of the super-system>) | (the <name of the embedded system>)]* meaning that, in the case of embedded systems, there may be one or more of them within a containing system which is called a super-system.

C. Priority grade

For this section, the same proposal of [22], was left, they made use of the MoSCoW technique, in which three degrees of priority are established: essential, recommended and, desirable.

1. Essential requirements. They are requirements that must be implemented to achieve the success of the product or product line. The word "shall" is used.
2. Recommended requirements; that is, they are important, but not necessary for the success of the product or product line. The word "should" is used.
3. Desirable (but not necessary) requirements because they could improve user experience and customer satisfaction. The word "can(n)" or "could(n)" is used.

If a motion sensor is activated, the system **should** send an instant image to the homeowner's email.

D. Functional requirements

It was decided to name this section as functional requirements according to what was proposed by [14] which establishes that […] "a functional requirement indicates which are the functions that the system must provide to the agents that use it, including as agents the other systems with which it must interact". The use of these requirements with their respective subcategories is described below:

1. **Agent interaction**

Although embedded systems are more oriented to action than to interaction, some relationships are established with the interacting systems and that contribute to the fulfillment of the general function of the super-system.

Interactions with super-system: Here only the functions that allow the interaction of the embedded system and the super-system are detailed.

Interactions with other ES and the environment: The presence of hierarchy allows each embedded system included in a super-system to be assigned a specific function, which contributes to the achievement of a general function. In this subcategory, the interrelationships with the embedded systems included within the super-system and the communication with the environment will be taken into account.

An example of the use of agent interaction would be:
The motion-sensing system must communicate with its super-system through input I1 or "enable".
The movement sensing system must receive stimuli from the environment through physical signals.

2. **Agent action**

These requirements deal with the functionalities and services that the system must provide to fulfill its specific purpose within the super-system.

Input functionalities: To determine these functionalities, it is necessary to observe the embedded system as a black box, and analyze what inputs it must receive, both from the super-system and from the other embedded systems, to start its operation.

Process functionalities: At this point, it is necessary to observe the embedded system as a white box, to find the functions that it must provide and achieve the outputs that will be connected with other systems. Typical process functions in an embedded system include initiation, configuration, diagnostics, storage, and termination.

Output functionalities: Like the input functionalities, for this subcategory, the system is observed as a black box and it is determined that it must deliver to the other systems to fulfill its specific function.

Vol. 16, No. 3, Julio-Septiembre, 2022
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 137-156

3. Fault detection/prevention

They are requirements related to the system's reaction to errors and failures, therefore, it is the way to make a requirement such as a system reliability functional.

Failure prevention: These requirements express features that must be included to minimize the chances of system failure before it becomes operational. For example, it is important to incorporate reliable hardware components and to use rigorous specification and design methodologies.

Fault detection: Despite defining characteristics to prevent faults from occurring, they can occur, and it is necessary to have mechanisms that allow determining when the system entered an error state and what actions must be taken to recover. The incorporation of self-check facilities and the use of redundant system modules are examples. The inclusion of contingency plans also increases fault tolerance, this implies identifying the critical functions of the system, and for each function determining the possible faults that may occur, and, finally, indicating possible solutions to each fault. It is important to emphasize these requirements, bearing in mind that it will be the embedded system itself that is recovered, bearing in mind that it has little interaction with the human user.

E. Timing requirements

Timing is a key factor when working with real-time embedded systems, therefore three key factors must be taken into account:

Term of time: The times in which the stimuli must be processed and produce some response by the system.

Frequency: The number of times per second that a process must execute to rest assured that you can always meet the deadlines.

Execution time: It refers to the time required to process a stimulus and produce a response.

Validation of the proposed template

To assess satisfaction with the proposed template, the IADOV technique was used. A questionnaire was applied to 13 professors of Software Engineering at the University of Informatics Sciences, which consisted of five questions, three closed (2, 4, and 5) and two open (1 and 3).

The closed questions were: 2) Do you consider that the five sections of the proposed template are adequate to specify syntactically a functional requirement of an embedded system?, 4) If you had to choose between the proposed template and another? Would you choose this? and 5) Are you satisfied with the proposed template?

To obtain the Group Satisfaction Index (ISG), we worked with the different levels of satisfaction that are expressed on a numerical scale that ranges between +1 and - 1, as follows: 1. Maximum satisfaction, 2. More satisfied than dissatisfied, 3. Not defined or contradictory, 4. More dissatisfied than satisfied and 5. Maximum dissatisfaction.

The individual satisfaction according to the satisfaction scale used is Clear satisfaction (69.23 %), More satisfied than dissatisfied (23.08), Not defined or contradictory (0.08)

Group satisfaction was calculated using the following formula:

$$ISG = \frac{A(+1) + B(+0{,}5) + C(0) + D(-0{,}5) + E\,(-1)}{N}$$

In this formula, A, B, C, D, and E represent the number of subjects with individual index 1; two; 3 or 6; 4; 5, and N represents the total number of subjects in the group, which in this case was 13.
Taking into account the results of individual satisfaction and the formula presented, the ISG was: 0.81, indicating high satisfaction with the proposed template.

Conclusion

With the development of this research, the following conclusions were reached. Embedded systems have distinctive characteristics that lead to the description of their functional requirements being characterized as a stimulus-response model. The main existing templates to define the syntactic structure of requirements do

not cover the needs of embedded systems. The proposed template gathers the necessary characteristics to specify the syntactic structure of a functional requirement in an embedded system.

References

Dardenne, V. Lamsweerde, and S. Fickas, Goal Directed Requirements Acquisition. 1993.

Ferrari, "Natural Language Requirements Processing: From Research to Practice," 2018 IEEE/ACM 40th International Conference on Software Engineering: Companion (ICSE-Companion), 2018, pp. 536-537.

Ferrari, L. Zhao and W. Alhoshan, "NLP for Requirements Engineering: Tasks, Techniques, Tools, and Technologies," 2021 IEEE/ACM 43rd International Conference on Software Engineering: Companion Proceedings (ICSE-Companion), 2021, pp. 322-323, doi: 10.1109/ICSE-Companion52605.2021.00137.

Mavin, P. Wilkinson, A. Harwood, and M. Novak, "EARS (Easy Approach to Requirements Syntax)," Proc. IEEE Int. Conf. Requir. Eng., no. October, pp. 317–322, 2009, doi: 10.1109/RE.2009.9.

Sutcliffe, "Scenario-Based Requirements Engineering," in 11th IEEE International Requirements Engineering Conference (RE'03), 2003, p. 11: 320.

Potts, "ScenIC: A Strategy for Inquiry-Driven Requirements Determination," in 4th IEEE International Symposium on Requirements Engineering, 1999, pp. 58-65.

Rupp, Requirements Templates - The Blueprint of your Requirement. 2014.

Zapata, K. Palomino, and R. Rosero, "A method for coordinative and prepositional syntactic disambiguation," Dyna, vol. 75, no. 156, pp. 29–42, 2008.

Majumdar, S. Sengupta, A. Kanjilal, and S. Bhattacharya, "Adv-EARS: A formal requirements syntax for derivation of use case models," Commun. Comput. Inf. Sci., vol. 198 CCIS, pp. 40–48, 2011, doi: 10.1007/978-3-642-22555-0_5.

Leiter and A. Lamsweerde, "Deriving Operational Software Specifications from System Goals," in Proceedings 10th ACM Symposium on the Foundations of Software Engineering, Charleston., 2002.

Dalpiaz, A. Ferrari, X. Franch and C. Palomares, "Natural Language Processing for Requirements Engineering: The Best Is Yet to Come," in IEEE Software, vol. 35, no. 5, pp. 115-119, September/October 2018, doi: 10.1109/MS.2018.3571242.

Dalpiaz, I. Van Der Schalk, S. Brinkkemper, F. B. Aydemir, and G. Lucassen, “Detecting terminological ambiguity in user stories: tool and experimentation,” Information and Software Technology, vol. 110, pp. 3–16, 2019.

Bauer et al., “Software Engineering An Advanced Course,” in Software Engineering An Advanced Course, 1975.

Li, C. Zheng, M. Li and H. Wang, "Automatic Requirements Classification Based on Graph Attention Network," in IEEE Access, vol. 10, pp. 30080-30090, 2022, doi: 10.1109/ACCESS.2022.3159238.

Urrego, “ABC-Besoins: Une approche d’ingénierie de besoins fonctionnels et non-fonctionnels centrée sur les Agents, les Buts, et les Contextes,” Universidad Paris 1, Pantéon Sorbona, Francia, 2005.

Sung, H. Baek, J. Lee, "Necessary Feasibility Analysis for Mixed-Criticality Real-Time Embedded Systems", IEEE Transactions on Parallel and Distributed Systems, vol. 33, no. 7, pp. 1520-1537, 2022, doi: 10.1109/TPDS.2021.3118610.

Sommerville, Ingeniería de software. Novena edición. 2011.

Pohl, Requirements Engineering - Fundamentals, Principles, and Techniques. Springer, 2010.

Wiegers and J. Beatty, Software Requirements, Third Edition. 2013.

Gonz and U. Giraldo, “Modelo de requisitos para sistemas embebidos,” no. 13, pp. 111–127, 2008.

Schneider and S. Arndt, “Reducing Natural-Language Ambiguities in Requirements Engineering,” ASK Mag. Issue 49, Winter 2013, pp. 38–40, 2013.

Zhao, W. Alhoshan, A. Ferrariet al., , "Natural Language Processing (NLP) for requirements engineering: A systematic mapping study", 2020, doi: 10.1145/3444689.

Augusto, C. Vásquez, M. Hugo, V. Huerta, L. Jaime, and P. Quispe, “Procesamiento de lenguaje natural,” Rev. Ing. Sist. e informática, vol. 6, no. 2, pp. 45–54, 2009.

Glinz, “A glossary of requirements engineering terminology,” Stand. Gloss. Certif. Prof. Requir. Eng. Stud. Exam, Version, vol. 1, no. May, 2017.

Glinz, H. Loenhoud, S. Staal, S. Bühne, "Handbook for the CPRE Foundation Level according to the IREB Standard", 2020

Osama, A. Zaki-Ismail, M. Abdelrazek, J. Grundy and A. Ibrahim, "Score-Based Automatic Detection and Resolution of Syntactic Ambiguity in Natural Language Requirements," 2020 IEEE International

Conference on Software Maintenance and Evolution (ICSME), 2020, pp. 651-661, doi: 10.1109/ICSME46990.2020.00067.

Riaz, W. H. Butt and S. Rehman, "Automatic Detection of Ambiguous Software Requirements: An Insight," 2019 5th International Conference on Information Management (ICIM), 2019, pp. 1-6, doi: 10.1109/INFOMAN.2019.8714682.

Vierlboeck, D. Dunbar and R. Nilchiani, "Natural Language Processing to Extract Contextual Structure from Requirements," 2022 IEEE International Systems Conference (SysCon), 2022, pp. 1-8, doi: 10.1109/SysCon53536.2022.9773855.

Jadallah, J. Fischbach, J. Frattini and A. Vogelsang, "CATE: CAusality Tree Extractor from Natural Language Requirements," 2021 IEEE 29th International Requirements Engineering Conference Workshops (REW), 2021, pp. 77-79, doi: 10.1109/REW53955.2021.00018.

Kiyavitskaya, N. Zeni, L. Mich, and D. M. Berry, “Requirements for tools for ambiguity identification and measurement in natural language requirements specifications,” Proc. 10th Work. Requir. Eng. WER 2007, no. January 2007, pp. 197–206, 2007.

Marwedel, Embedded Design System: Embedded Systems Foundations of Cyber-Physical Systems, and the Internet of Things, 4ta ed. 2021.

Vallejo, R. Mazo, C. Jaramillo, and J. M. Medina, “Towards a new template for the specification of requirements in semi-structured natural language,” J. Softw. Eng. Res. Dev., vol. 8, no. Sophist 2014, p. 3, 2020, doi: 10.5753/jserd.2020.473.

Garigliano, D. Perini, and L. Mich, “Which Semantics for Requirements Engineering: From Shallow to Deep.,” In the 1st Workshop on Natural Language Processing for Requirements Engineering (NLP4RE) 2018.

Mazo, C. A. Jaramillo, and U. P. Sorbonne, “Hacia una nueva plantilla para la especificación de requisitos en lenguaje natural semi-estructurado,” 2019.

Pressman, Ingeniería del software. Un enfoque práctico. Séptima edición. México, 2010.

Sonbol, G. Rebdawi and N. Ghneim, "Towards a Semantic Representation for Functional Software Requirements," 2020 IEEE Seventh International Workshop on Artificial Intelligence for Requirements Engineering (AIRE), 2020, pp. 1-8, doi: 10.1109/AIRE51212.2020.00007.

Cheng, B, Konrad, S, Kamdoum, "Enabling a Roundtrip Engineering Process for the Modeling and Analysis of Embedded Systems," in Proceedings of the ACM/IEEE International Conference on Model Driven Engineering Languages and Systems. Italia, 2006.

Raikar and N. G. Cholli, "An Analysis of Ambiguity Detection Techniques for Software Requirement Specification," 2021 IEEE International Conference on Computation System and Information Technology for Sustainable Solutions (CSITSS), 2021, pp. 1-4, doi: 10.1109/CSITSS54238.2021.9683497.

Ambreen, N. Ikram, M. Usman, and M. Niazi, "Empirical research in requirements engineering: trends and opportunities," Requirements Engineering, vol. 23, no. 1, pp. 63–95, 2018.

Koscinski, C. Gambardella, E. Gerstner, M. Zappavigna, J. Cassetti and M. Mirakhorli, "A Natural Language Processing Technique for Formalization of Systems Requirement Specifications," 2021 IEEE 29th International Requirements Engineering Conference Workshops (REW), 2021, pp. 350-356, doi: 10.1109/REW53955.2021.00062.

Alhoshan, R. Batista-Navarro, and L. Zhao, "Using Frame Embeddings to Identify Semantically Related Software Requirements," In the 2nd Workshop on Natural Language Processing for Requirements Engineering (NLP4RE). 2019.

Conflicto de interés

El autor autoriza la distribución y uso de su artículo.

Contribuciones de los autores

1. Conceptualización: Yosanne Garzón Mendoza
2. Curación de datos: Dunia María Colomé Cedeño
3. Análisis formal: Yosanne Garzón Mendoza
4. Adquisición de fondos: Jessenia Chalen Ortega
5. Investigación: Yosanne Garzón Mendoza, Dunia María Colomé Cedeño, Yorgelys González López, Jessenia Chalen Ortega

6. Metodología: Dunia María Colomé Cedeño
7. Administración del proyecto: Yorgelys González López
8. Recursos: Jessenia Chalen Ortega
9. Software: Yosanne Garzón Mendoza
10. Supervisión: Yorgelys González López
11. Validación: Dunia María Colomé Cedeño
12. Visualización: Yosanne Garzón Mendoza
13. Redacción – borrador original: Yosanne Garzón Mendoza, Yorgelys González López
14. Redacción – revisión y edición: Dunia María Colomé Cedeño, Jessenia Chalen Ortega