Scientific Research Publishing

Journal of Computer and Communications, 2018, 6, 40-54
http://www.scirp.org/journal/jcc
ISSN Online: 2327-5227
ISSN Print: 2327-5219

Hermeneutical Theodolite of Requirements: Evaluating and Revealing the Quality Grades of Software Requirements and of Domain of Application

Wagner Varalda, Ítalo Santiago Vega

Program of Intelligence Technologies and Digital Design, Pontifical Catholic University of São Paulo, São Paulo, Brazil
Email: wagnervaralda@uol.com.br, ítalo@pucsp.br

How to cite this paper: Varalda, W. and Vega, Í.S. (2018) Hermeneutical Theodolite of Requirements: Evaluating and Revealing the Quality Grades of Software Requirements and of Domain of Application. *Journal of Computer and Communications*, **6**, 40-54.
https://doi.org/10.4236/jcc.2018.65004

Received: April 12, 2018
Accepted: May 21, 2018
Published: May 24, 2018

Abstract

Throughout the development of software, during Requirements Engineering activities, software requirements dynamically and constantly evolve and mature from an "identified" stage to an "approved" stage. This evolution takes place individually for each requirement, in a very particular way, because it depends on the level of understanding that the requirements engineer reaches in relation to it. How, then, to monitor the evolution of each software requirement? How to know the quality of each software requirement? How to measure the level of understanding and difficulty that the requirements engineer has in relation to each software requirement? This paper aims to present a proposal to answer these questions through the use of an instrument developed specifically to assess and reveal the quality grades of each software requirement and also to assess and reveal that the levels of understanding and of difficulty of the requirements engineer is in relation to each software requirement. This instrument was called the Hermeneutical Theodolite of Requirements, which also can be applied to evaluate that the levels of understanding and of difficulty of the requirements engineer is in relation to the domain of application, essential input artifact and primordial to the specification of the requirements of software.

Keywords

Hermeneutical Theodolite, SOLO Taxonomy, OMG Essence, Hermeneutical Engineering of Requirements

1. Introduction

During the development of a software, the requirements engineer must under-

stand the business needs to be met by this software, for your requirements to be specified in such a way that your behavior is defined correctly and in accordance with the fundamental needs to be met, with the problems to be solved, with the functionalities to be made available, with the information to be processed, with the performances to be achieved, with the restrictions to be considered and with the interfaces to be created [1]. The results of these activities culminate in the specifications of the requirements of this software, which will serve as a subsidy for your architectural project, its construction, its tests, its plans, its estimates and its deliveries.

Each software requirement has its own evolution dynamics. While some progress faster and more easily, others need more time and attention to reach the appropriate maturity level. All of these factors, however, also depend on the ability of the requirements engineer to understand and interpret the application domain for which the software will be developed. The better the requirements engineer understand this item, the better and more consistent the software requirements specifications will be. But how to know the level of understanding (and also the level of difficulty) in which the requirements engineer is in relation to the application domain? Likewise, how to know the degree of quality (and also the degree of difficulty) in relation to each of the software requirements?

The Hermeneutical Theodolite of Requirements is an instrument that aims, through the use of two mechanisms, to evaluate and reveal the levels of understanding (and also those of difficulty) in which the requirements engineer is in relation to the domain of the application and also evaluate and reveal the degrees of quality (and also those of difficulty) in relation to each software requirement. Thus, once we know these realities, it is possible to establish strategies to improve the performance of the Requirements Engineering in the project and, therefore, to improve the process of identification, analysis and specification of these software requirements.

Used as theoretical foundation, the SOLO Taxonomy [2], the OMG Essence [3] and the Hermeneutical Engineering of Requirements [4] were adapted and customized exclusively to be used in the Hermeneutical Theodolite of Requirements. The adequacy and customization of SOLO Taxonomy will meet the levels of understanding and difficulty of the requirements engineer in relation to the application domain. The suitability and customization of OMG Essence meet of the degrees of quality (and also of difficulty) grades of software requirements. In the case of Hermeneutical Engineering of Requirements, its hermeneutical conceptualization is in line with the formation of the elements that establish and determine the levels of understanding of the requirements engineer who used to assess their current state of knowledge and indicate their progression needed to gain a better understanding of the application domain and of software requirements.

In relation to the application of Requirements Engineering, the Hermeneutical Theodolite of Requirements has no dependence or relation with some specific

practices. This means that it can be applied regardless of the techniques, processes, and paradigms used for software development.

2. SOLO Taxonomy

The SOLO Taxonomy is a model that classifies students' learning outcomes in relation to any activity or task, describing their understanding results at one of five levels of complexity: no idea, one idea, loose ideas, connected ideas, extended ideas. With this, teachers are able to individually identify for each student their level of understanding of the subject they are studying and thereby create individual and personalized learning tasks to help them succeed in their studies and progress more easily in their apprenticeship, evaluating their current stage and planning the next steps to obtain their learning [2].

Educational institutions use the SOLO Taxonomy as a common language for the learning and are able to discover their students' prior knowledge and, thus, to develop better research plans for their students, describing the actual learning objectives to be achieved, clearly showing the cognitive complexity of the task and the evaluation criteria for each of the levels. As a result, it is also possible to better plan the necessary and appropriate resources to be used to support the learning process, avoiding unnecessary costs.

SOLO is the acronym for Structure of the Observed Learning Outcome. This taxonomy was elaborated by the authors John Biggs and Kevin Collis in 1982, of which they identified and organized cognitive characteristics in five structured stages [5], as highlighted in **Table 1**.

As mentioned by one of its founders and developers, the SOLO Taxonomy "SOLO is used in constructive alignment to design learning interactions and success criteria for results-based education" (Biggs, 2003).

3. OMG Essence

Created in 2014 through a partnership between SEMAT (Software Engineering Methods and Theory) [6] and OMG (Object Management Group) [7], Essence

Table 1. The five levels of SOLO Taxonomy.

SOLO Taxonomy The five levels with their respective capacities (cognitive characteristics)	
Levels	Capacities
Prestructural	Minimum ability to find suggestions, giving confusing answers.
Unistructural	Poor ability to find relevant suggestions or data.
Multistructural	Medium ability to discover suggestions and recognize relevant data.
Relational	High ability to find suggestions, expose relevant information and interrelationships.
Abstract (Extended)	Maximum capacity to find suggestions, to disclose relevant information, to establish interrelations and to elaborate hypotheses.

is a universal language for defining common methods and practices of Software Engineering and describes the essential elements to the software development, helping practitioners compare software engineering methods to, like this, make better decisions about their practices and apply them independently.

With OMG Essence [3], a software development team can create its own practices library and share it with other teams and thereby compare and discuss how can improve its Software Engineering practices.

With Essence it is also possible to evaluate the progress of a software development team during the development of a project so that it is clearly aware of its current state and what it must do to improve its performance.

In the case of Requirements Engineering, the evaluation and evolution of software requirements are made through the states conceived, bounded, coherent acceptable, addressed and fulfilled. The transition from one state to another occurs according to the evolution of the understanding that the software development team is acquiring in relation to the software requirements, as presented in **Table 2**.

4. Hermeneutical Engineering of Requirements

The Hermeneutical Engineering of Requirements [4] is a proposal that aims to enable the requirements engineer to better understand and interpret the application domain and, therefore, to specify more precisely the requirements of the software.

Its theoretical basis was constituted of some hermeneutic concepts created by the German philosopher Martin Heidegger (1889-1976), more specifically the Dasein, the being-in-the-world, the being-with-others and the being-for-death, which were adapted to form the Hermeneutical Engineering of Requirements. The results of these adaptations are presented in **Table 3**.

5. Hermeneutical Theodolite of Requirements

Throughout the lifecycle of software development, the requirements engineer's understanding of the requirements of this software evolves as one gains greater knowledge about them. Likewise, the levels of understanding that requirements engineer has about the application domain evolve according getting more knowledge about it.

To evaluate and reveal the different levels of understanding and difficulty that the requirements engineer has in relation to the application domain and also to evaluate and reveal the different degrees of quality in which each software requirement is found, it is proposed the utilization of the Hermeneutical Theodolite of Requirements, an instrument composed of two mechanisms: one that acts on the application domain and another that acts on the software requirements.

The instrument that acts on the domain of the application uses as theoretical foundations the SOLO Taxonomy (explained in Section 2) and the Hermeneutical Engineering of Requirements (explained in Section 4), which were adapted

Table 2. OMG Essence—states relating to Requirements Engineering.

OMG Essence States Relating to Requirements Engineering	
States	Progress to Successful Completion
Conceived	• The initial set of stakeholders agrees that a system is to be produced. • The stakeholders that will use the new system are identified. • The stakeholders that will fund the initial work on the new system are identified. • There is a clear opportunity for the new system to address.
Bounded	• The stakeholders involved in developing the new system are identified. • The stakeholders agree on the purpose of the new system. • It is clear what success is for the new system. • The stakeholders have a shared understanding of the extent of the proposed solution. • The way the requirements will be described is agreed upon. • The mechanisms for managing the requirements are in place. • The prioritization scheme is clear. • Constraints are identified and considered. • Assumptions are clearly stated.
Coherent	• The requirements are captured and shared with the team and the stakeholders. • The origin of the requirements is clear. • The rationale behind the requirements is clear. • Conflicting requirements are identified and attended to. • The requirements communicate the essential characteristics of the system to be delivered. • The most important usage scenarios for the system can be explained. • The priority of the requirements is clear. • The impact of implementing the requirements is understood. • The team understands what has to be delivered and agrees to deliver it.
Acceptable	• The stakeholders accept that the requirements describe an acceptable solution. • The rate of change to the agreed requirements is relatively low and under control. • The value provided by implementing the requirements is clear. • The parts of the opportunity satisfied by the requirements are clear. • The requirements are testable.
Addressed	• Enough of the requirements are addressed for the resulting system to be acceptable to the stakeholders. • The stakeholders accept the requirements as accurately reflecting what the system does and does not do • The set of requirement items implemented provide clear value to the stakeholders. • The system implementing the requirements is accepted by the stakeholders as worth making operational.
Fulfilled	• The stakeholders accept the requirements as accurately capturing what they require to fully satisfy the need for a new system. • There are no outstanding requirement items preventing the system from being accepted as fully satisfying the requirements. • The system is accepted by the stakeholders as fully satisfying the requirements.

Table 3. Hermeneutical engineering of requirements—results of the adaptations of the hermeneutical concepts.

Hermeneutical Engineering of Requirements The results of the adaptations of the hermeneutical concepts created by the philosopher Martin Heidegger	
Hermeneutical Concepts	Result of the Adaptation to the Hermeneutical Engineering of Requirements
Dasein's Triad	Composition of the Triad of Hermeneutical Engineering of Requirements, consisting of "Identification of Situational Difference", "Examination of Situational Difference" and "Specification of Requirement".
Being-in-the-world	Composition of the Situational Difference Identification, whose purpose is to identify and understand the "Context of Situational Difference" in relation to the "Business Community".
Being-with-others	Composition of the Examination of Situational Difference, which aims to understand the "Problems/Opportunities", their "Circumstances" and identify a set of "Possibilities" and "Benefits" to be offered by the software to be developed.
Being-for-death	Composition of the Requirement Specification, which aims to declare and approve the "Original Needs" (together with the "expectations"), produce and approve the "Acceptable Specification" and to specify and approve the "Software Requirements".

exclusively to the Hermeneutical Theodolite of Requirements to organize the five levels of understanding that the requirements engineer can be in relation to the application domain, as highlighted in **Table 4**.

When applying the Hermeneutical Theodolite of Requirements to evaluate and reveal the levels of understanding and difficulty in which the requirements engineer is in relation to the application domain, the following results will be displayed, as shown in **Figure 1**.

The instrument that acts on the software requirements uses OMG Essence (explained in Section 3) as a theoretical basis, which has been adapted exclusively to the Hermeneutical Theodolite of Requirements to organize the five states of evolution of the software requirement is, as highlighted in **Table 5**.

When applying the Hermeneutical Theodolite of Requirements to evaluate and reveal the quality grades of the software requirements, the following results will be displayed, as shown in **Figures 2-6**, organized by requirement state.

6. Application of the Hermeneutical Theodolite of Requirements

It is possible to apply the Hermeneutical Theodolite of Requirements in any software development project, regardless of its domain of application and its degree of complexity. By way of example, is presented in this article the application of the Hermeneutical Theodolite of Requirements in a software development project for the "Game of Memory".

The decision to choose the "Game of Memory" was taken by the fact that it is a game well known and easy to understand. Thus, the application of the Hermeneutical

Table 4. The five levels of understanding of the requirements engineer in relation to the application domain.

Hermeneutical Theodolite of Requirements The five levels of understanding of the requirements engineer in relation to the application domain	
Level	Knowledge Related to the Application Domain
1-Pre-conceptual	Identifies what is happening (the problem and/or the opportunity) and to whom it is happening (business community), but still does not know details about the events that occur between them.
2-Conceptual	Identifies what is happening (the problem and/or the opportunity) and to whom it is happening (business community) and also already knows the details about the events that occur between them.
3-Contextual	Knows and contextualizes why facts occur, along with their circumstances, situational differences, problems and/or opportunities.
4-Systemic	Knows how each involved of the business community perceives, is impacted and deals with each circumstance, situational difference and problem and/or opportunity.
5-Holistic	Identifies a set of possibilities (and their respective benefits) that can be adopted by stakeholders of the business community to address (or mitigate) problems and opportunities, as well as their business processes, scenario, environment and utensils (or inputs) that contextualize them.

Table 5. The evolution states of the software requirements and their respective sub-states.

Hermeneutical Theodolite of Requirements The evolution states of the software requirements and their respective sub-states	
States	Sub-States
Identified	• The requirement was identified individually for stakeholders. • The origin and classification of the requirement are clear. • The business rules and the restrictions imposed on the requirement are known. • The requirement has been described briefly and concisely.
Conceived	• The requirement communicates its essential characteristics. • The requirement has been prioritized. • The requirement has no conflict with another requirement. • It is possible to trace the requirement.
Described	• The requirement is clear in relation to its scope. • The requirement is consistent with the expectations of stakeholders. • Stakeholders accept that the requirement accurately captured it's what it does what does not. • The allocation of the requirement has been made.
Declared	• The requirement complies with the required standards. • The set of requirements items provides clear value to stakeholders. • The requirement has been specified consistently. • Is possible test and evaluate the requirement.
Approved	• The requirement is complete and consistent. • The requirement has no omissions and no ambiguities. • There are no items pending in the requirement, preventing its acceptance by stakeholders. • The requirement was accepted by stakeholders as being fully meeting their needs.

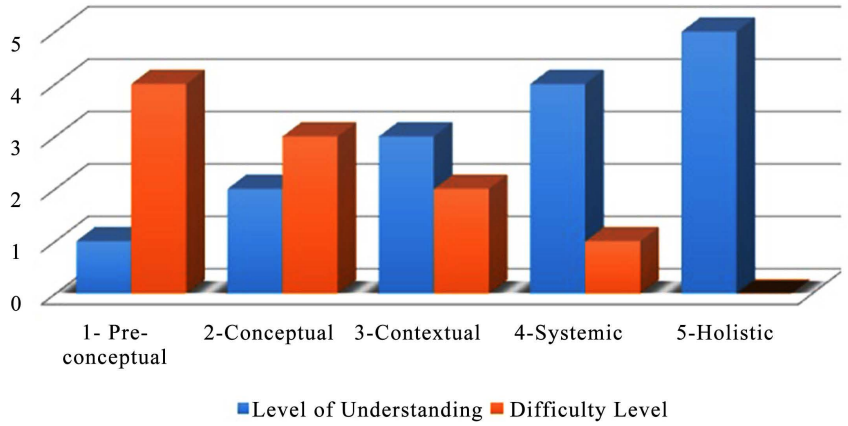

Figure 1. Revelation of the levels of understanding and of the difficulty in relation to the application domain.

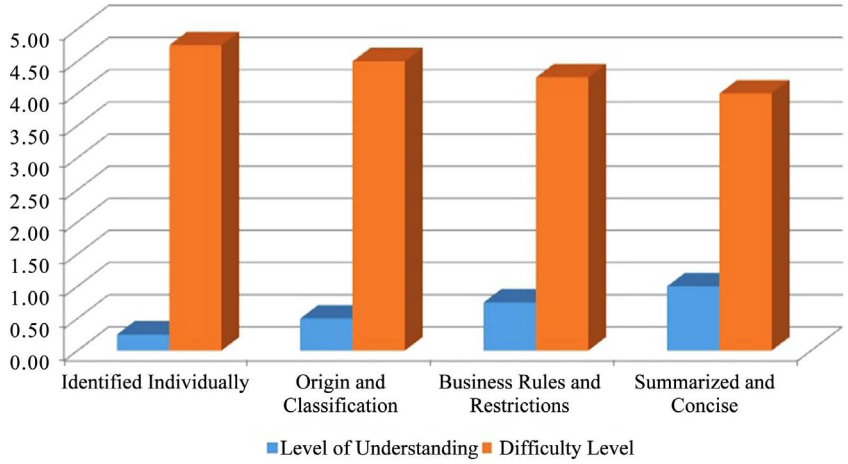

Figure 2. State Identfied—Quality grades of the software requirements.

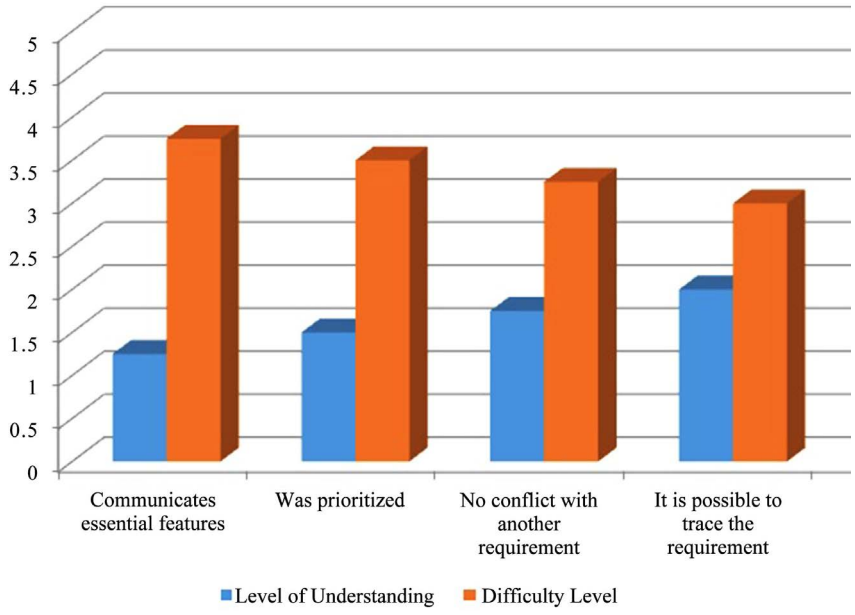

Figure 3. State Conceived—Quality grades of the software requirements.

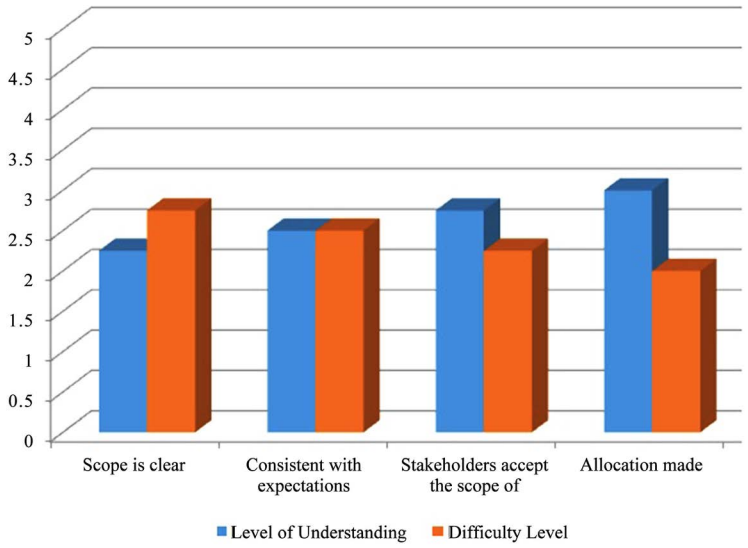

Figure 4. State Described—Quality grades of the software requirements.

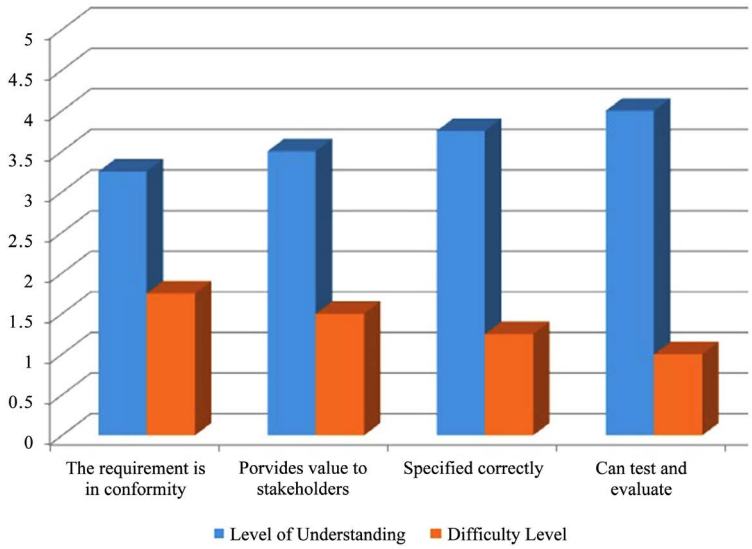

Figure 5. State Declared—Quality grades of the software requirements.

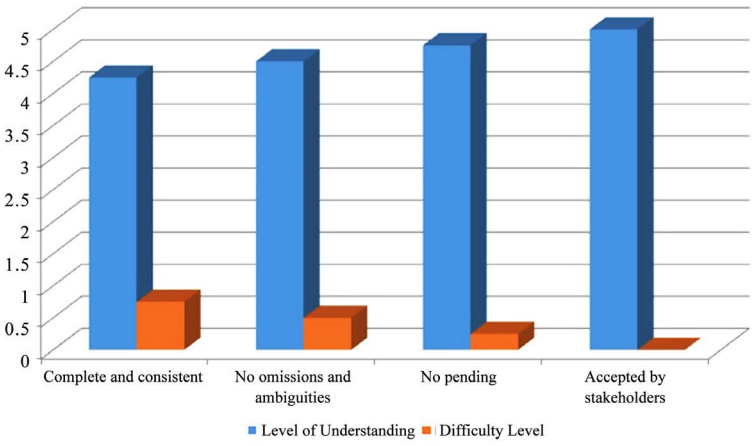

Figure 6. State Approved—Quality grades of the software requirements.

Theodolite of Requirements is presented more directly, without the need for large explanations about the application domain.

The "Game of Memory", in its essence, is a game of pieces (usually cards) that contains varied figures on one of its faces. Each figure is repeated in two pieces. To start the game, shuffle the pieces and distribute them with the faces of the figures facedown, so that the players do not see them. Each player, one at a time, to come two pieces, so that all other players also see them and know which figures have been revealed by these pieces. If the figures revealed by these pieces are the same, the player who took them off collects them for themselves and continues to play. If the figures of these pieces are different, the player who revealed them put gets them in their respective places and the time to play is by another player, who will repeat the process in order find pairs of identical pieces. This cycle repeats until there are no more parts to be untapped. The player who manages to collect the most equal pieces wins the game [8].

Given this brief description of the "Game of Memory", the software "Game Electronic of Memory" should automate this scenario, taking into account four particularities: 1) the game should be played by only one player; 2) the player can choose difficulty levels (determined by number of pieces, ranging from 10 to 40; 3) the player may stop a match to continue playing at another time; 4) the player may start a new match at any time.

In hypothetical way, we consider that to specify the requirements of this software it took four cycles of iterations. Thus, the applications of the Hermeneutical Theodolite of Requirement occurred in the following ways.

Cycle 1: During this cycle, the domain of the application was identified and well understood, but it was not possible to get a good view of the purpose of the software to be developed. Thus, when applying the Hermeneutical Theodolite of Requirements in this cycle, the following results were obtained, as shown in **Table 6** and **Figure 7**.

Cycle 2: During this cycle, it was fully understood; the application domain achieved a good level of understanding regarding the purpose of the software to be developed and it was identified the requirements "Choose difficulty level", "Start new game", "Save game" and "Recover the saved version", being that the first two software requirements were more understood than the last two. Thus, when applying the Hermeneutical Theodolite of Requirements in this cycle, the following results were obtained, as shown in **Table 7**, **Table 8**, **Figure 8** and **Figure 9**.

Cycle 3: During this cycle, the application domain and the purpose of the software to be developed were completely understood and the quality grades of software requirements evolved considerably, being that the requirements "Save game" and "Recover the saved version" still require more details. Thus, when applying the Hermeneutical Theodolite of Requirements in this cycle, the following results were obtained, as shown in **Table 9**, **Table 10**, **Figure 10** and **Figure11**.

Table 6. Game Electronic of Memory: Cycle 1—levels of the application domain under evaluation.

Example Application of Hermeneutical Theodolite of Requirements—Application Domain Game Electronic of Memory—Cycle 1	
Application Domain	Level of Understanding
Memory Game	3-Contextual
Game Electronic of Memory	2-Conceptual

Table 7. Game Electronic of Memory: Cycle 2—levels of the application domain under evaluation.

Example Application of Hermeneutical Theodolite of Requirements—Application Domain Game Electronic of Memory—Cycle 2	
Application Domain	Level of Understanding
Memory Game	5-Holistic
Game Electronic of Memory	4-Systemic

Table 8. Game Electronic of Memory: Cycle 2—evaluations of the quality grades of software requirements.

Example Application of Hermeneutical Theodolite of Requirements—Software Requirement Game Electronic of Memory—Cycle 2	
Software Requirement	Degree of Quality
Choose difficulty level	2.3—The requirement has no conflict with another requirement.
Start new game	3.3—Stakeholders accept that the requirement accurately captured it's what it does what does not.
Save game	1.4—The requirement has been described briefly and concisely.
Recover the saved version	1.3—The business rules and the restrictions imposed on the requirement are known.

Table 9. Game Electronic of Memory: Cycle 3—levels of the application domain under evaluation.

Example Application of Hermeneutical Theodolite of Requirements—Application Domain Game Electronic of Memory—Cycle 3	
Application Domain	Level of Understanding
Memory Game	5-Holistic
Game Electronic of Memory	5-Holistic

Table 10. Game Electronic of Memory: Cycle 3—evaluations of the quality grades of software requirements.

Example Application of Hermeneutical Theodolite of Requirements—Software Requirement Game Electronic of Memory—Cycle 3	
Software Requirement	Degree of Quality
Choose difficulty level	4.4—Is possible test and evaluate the requirement.
Start new game	4.3—The requirement has been specified consistently.
Save game	3.1—The requirement is clear in relation to its scope.
Recover the saved version	3.2—The requirement is consistent with the expectations of stakeholders.

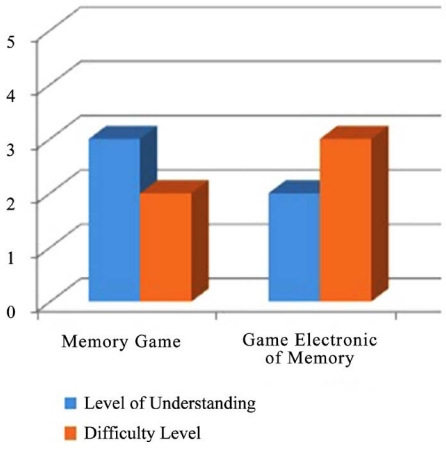

Figure 7. Game Electronic of Memory: Cycle 1—levels revealed about the application domain.

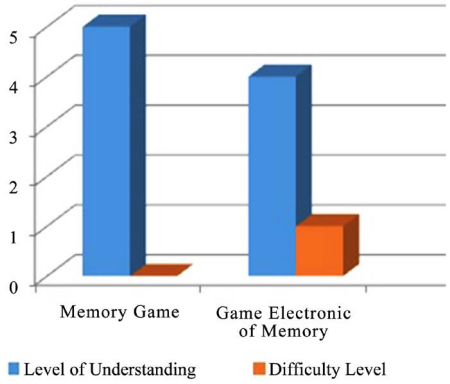

Figure 8. Game Electronic of Memory: Cycle 2—levels revealed about the application domain.

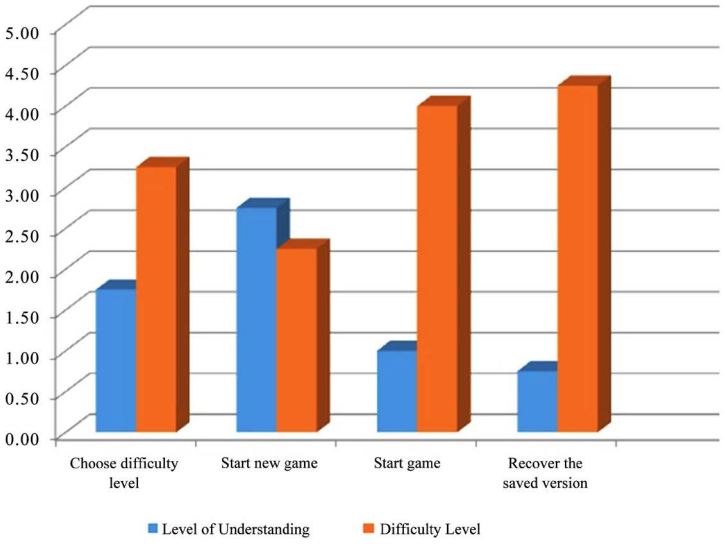

Figure 9. Game Electronic of Memory: Cycle 2—evaluations of the quality grades of software requirements.

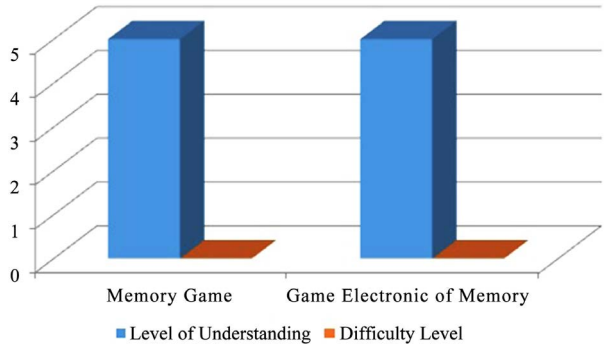

Figure 10. Game Electronic of Memory: Cycle 3—levels revealed about the application domain.

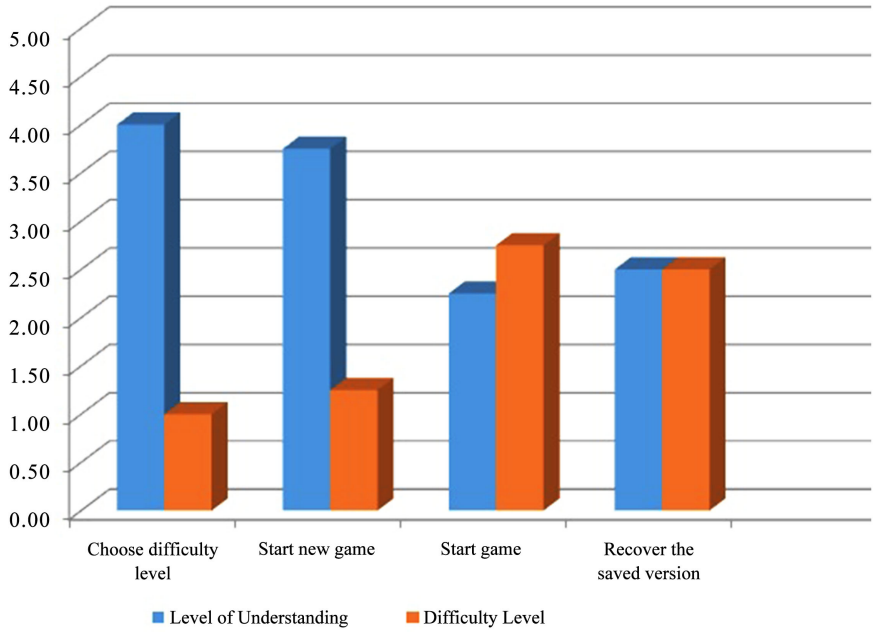

Figure 11. Game Electronic of Memory: Cycle 3—evaluations of the quality grades of software requirements.

Cycle 4: During this cycle, an excellent understanding of software requirements was achieved. Thus, when applying the Hermeneutical Theodolite of Requirements in this cycle, the following results were obtained, as shown in **Table 11** and **Figure 12**.

In this **Figure 12** a new possibility is presented: Turn on the Level Reference, which becomes a color scale to further aid the reading and interpretation of the results obtained, both for the software requirements and for the application domain. In this example, the red color represents high criticality, the yellow color means average criticality and the green color indicates low or no criticality.

7. Conclusions

Each software requirement, to reach the appropriate maturity level, progresses from an "identified" stage to an "approved" stage, in a very particular and

Table 11. Game Electronic of Memory: Cycle 4—evaluations of the quality grades of software requirements.

Example Application of Hermeneutical Theodolite of Requirements—Software Requirement Game Electronic of Memory—Cycle 4	
Software Requirement	Degree of Quality
Choose difficulty level	5.2—The requirement has no omissions and no ambiguities.
Start new game	5.3—There are no items pending in the requirement, preventing its acceptance by stakeholders.
Save game	4.3—The requirement has been specified consistently.
Recover the saved version	5.1—The requirement is complete and consistent.

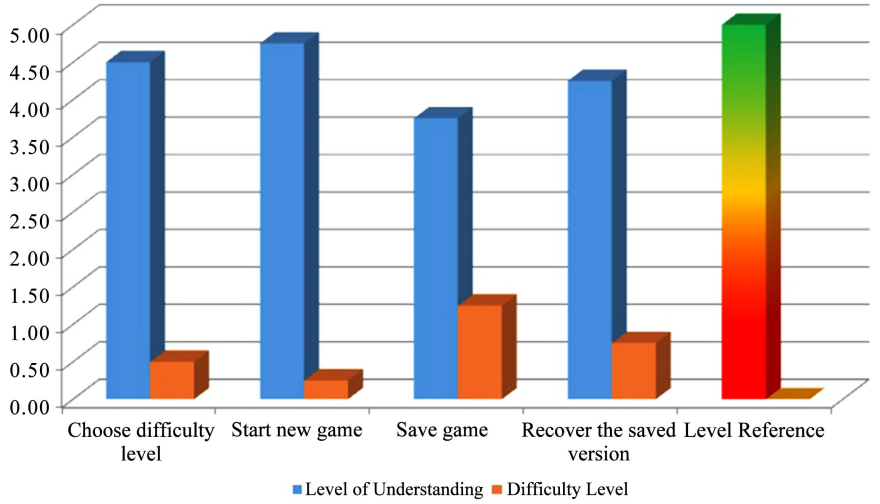

Figure 12. Game Electronic of Memory: Cycle 4—evaluations of the quality grades of software requirements.

individualized dynamics, as its evolution depends on the understanding and interpretation of the requirements engineer on this requirement, and also about the application domain for which the software will be developed. In this article, the Hermeneutical Theodolite of Requirements is presented, an instrument that aims to evaluate and reveal the levels of understanding (and difficulty) that the requirements engineer has in relation to the application domain, and also evaluate and reveal the quality grades of each software requirement. Thus, with these results at hand, one can determine a strategic plan to improve the application of Requirements Engineering.

In this article, the indicators of the levels of understanding of the requirements engineer in relation to the domain of the application, as well as the indicators of the quality grades of the software requirements, determined by their states and sub-states of evolution were presented. These indicators are the result of the adaptations made of the SOLO Taxonomy, of the OMG Essence and of the Hermeneutical Engineering of Requirements.

This article also presented how the Hermeneutical Theodolite of Requirements applies, using as an example a software development project for the

"Game of Memory". With this, it was possible to verify the practicality and simplicity of applying it, besides its independence with the methods, processes and tools adopted for the project.

In this example it was also presented the possibility of using the Level Reference to further facilitate the reading of the revealed results. This Level Reference indicates the degree of criticality of the application domain and software requirements, showing in a simple color scale how critical they are, with red being an indication for high criticality, the yellow is an indication for the average criticality and green an indication for low or no criticality. But this Level Reference can be configured in any way, according to the needs and characteristics of each project.

To evaluate the levels of understanding of the requirements engineer in relation to the application domain and the quality grades of software requirements is timely at a time when software complexity is increasing. With this, it is possible to improve the efficiency and effectiveness of Requirements Engineering and provide better planning and control over it, regardless of the application domain, the business area and its complexities.

References

[1] Roger, P. (2016) Software Engineering: A Professional Approach. McGraw-Hill Education, New York.

[2] Ceia, Mário José Miranda (2002) The SOLO Taxonomy and the Levels of Van Hiele. School Higher of Education of Portalegre, Portalegre.

[3] OMG (2015) Essence-Kernel and Language for Software Engineering Methods. SMSC.

[4] Varalda, W. and Vega, Í.S. (2017) Hermeneutical Engineering of Requirements. *Journal of Computer and Communications*, **5**, 7-16. https://doi.org/10.4236/jcc.2017.52002

[5] Official Page of John Biggs, Which Presents, among Other Things, the SOLO Taxonom. http://www.johnbiggs.com.au/academic/solo-taxonomy/

[6] Official Page of SEMAT. http://www.semat.org/

[7] Official Page of OMG. https://www.omg.org/

[8] Web College (2017-2018) How to Learn More with the Memory Game? https://www.colegioweb.com.br/educador/como-aprender-mais-com-o-jogo-da-memoria.html