Ingeniería
ISSN: 0121-750X
ISSN: 2344-8393
Universidad Distrital Francisco José de Caldas

Zapata-Jaramillo, Carlos Mario; Henao-Roqueme, Antony
A proposal for improving the Essence standard by using terminology unification
Ingeniería, vol. 26, no. 2, 2021, May-August, pp. 213-232
Universidad Distrital Francisco José de Caldas

DOI: https://doi.org/10.14483/23448393.16428

Research

REVISTA Ingeniería

A Proposal for Improving the Essence Standard by Using Terminology Unification

Una propuesta de mejoramiento del estándar Essence mediante el uso de unificación terminológica

Carlos Mario Zapata-Jaramillo*[1], ***Antony Henao-Roqueme***[1]

[1]Universidad Nacional de Colombia (Medellín-Colombia).
correspondence email: cmzapata@unal.edu.co

Recibido: 29/05/2020. Modificado: 27/08/2020. Aceptado: 27/05/2021.

Abstract

Context: SEMAT (Software Engineering Method and Theory) is promoting a software engineering theory with adequate terminology to improve the transference of methods and practices between teams. Terminologies should be uniform in order to eliminate ambiguity, improve communication among teams, and support new concepts.

Method: The process of reaching uniformity is called terminology unification. In this paper we propose a method for improving the Essence standard based on terminology unification. This method comprises four stages: selection of base models and definitions for structuring terms, identification of terminology problems by comparing the base models and definitions, unification of terms among the base models and definitions, and measurement of the gap between the current standard terms and the proposed changes.

Results: We propose a set of modifications to the Essence standard in constructs like: alpha state cards, relationships among alphas, and names of activity spaces.

Conclusions: By solving such conflicts, we can define a common, unambiguous terminology for software engineering teams.

Keywords: Essence standard, uniformity problems, terminology problems, terminology unification.

Language: English

Cite this paper as: Zapata-Jaramillo, C. M. and Henao-Roqueme, A.: A Proposal for Improving the Essence Standard by Using Terminology Unification. INGENIERÍA, Vol. 26, Num. 2, pp. 213-232 (2021).

https://doi.org/10.14483/23448393.16428

Resumen

Contexto: En *SEMAT (Software Engineering Method and Theory*) se promueve una nueva teoría de la ingeniería de software con terminología apropiada para mejorar la transferencia de métodos y prácticas entre equipos. La terminología debe ser uniforme para eliminar la ambigüedad, mejorar la comunicación entre equipos y apoyar el surgimiento de nuevos conceptos.
Método: Al proceso para alcanzar uniformidad se le denomina unificación terminológica. En este artículo se propone un mejoramiento del estándar *Essence* basado en la unificación terminológica. Este método comprende cuatro etapas: selección de modelos base y definiciones para estructurar términos, identificación de problemas terminológicos comparando las bases y definiciones, unificación de términos con base en los modelos y definiciones y medición de la brecha entre los términos actuales del estándar y los cambios propuestos.
Resultados: Se propone un conjunto de modificaciones al estándar *Essence* en constructos como cartas de es- tado de los alfas, relaciones entre los alfas y nombres de los espacios de actividad.
Conclusiones: Al corregir estos conflictos, es posible definir una terminología común y sin ambigüedades para todos los equipos de ingeniería de software.
Palabras clave: Estándar *Essence*, problemas de uniformidad, problemas terminológicos, unificación terminológica.
Idioma: Inglés

1. Introduction

Teams are creating methods and practices in order to address the growing demand of the software engineering industry, so they can produce high quality software on time and on budget [1]–[5]. However, some circumstances in running the software engineering endeavor lead teams to continuously tailoring their own methods and practices, so the previously gained knowledge is abandoned. Consequently, knowledge transference among teams is getting harder [1], [6]–[8]. Aiming to improve such transference, the SEMAT (Software Engineering Method and Theory) community is promoting a software engineering theory (the Essence standard) which is focused on identifying universal elements covering all software engineering endeavors. Such elements are expressed in terms of a simple and structured language, thus allowing for the definition of methods and practices, so they can be easily transferred, tailored, measured, and compared among teams [6], [7]. Universal elements forming the Essence Kernel are known as alphas, activity spaces, and competencies. Software engineering as a discipline includes a set of concepts for easing communication among teams. Such concepts are included into the specific terminology for software engineering [6].

Nedobity [9] warns about human-to-machine and machine-to-machine communication problems arising from deficient terminologies. Accordingly, a theory with uniformity problems is unable to provide guidance to the procedures of a discipline. Such a problem generates gaps between the real progress of the team and the progress assessed by the theory. According to Cabré [10], a theory should have three degrees of adequacy: observational, descriptive, and predictive. Since some of the elements of the theory lack uniformity, theories fail to reach such degrees, and they are unable to support new concepts generated within a discipline. Also, lack of uniformity is associated with the impossibility to compare information in the documents of a discipline [11]. The process aimed to eliminate ambiguity and improve the communication among teams is called terminology

unification [10]. Some terminology problems are reported in domains like nursing [11], research and teaching [12], archive [13], automotive industry [14], and natural language requirements [15]. Such problems can be so severe that they sometimes need the intervention of governmental offices in order to achieve terminological systems [16].

We can find terminology problems in both the Essence standard and the language in which it is described [17], [18]. Constructs and definitions are affected by such problems. Consequently, in this paper we apply terminology unification to the Essence standard by selecting some base models and definitions for structuring terms, identifying disunited terms by comparing the base models and definitions, unifying terms among the base models and definitions, and measuring the gap between the current standard terms and the proposed changes. We can turn the Essence standard into a uniform theory that allows for the completion of the Essence kernel by solving such problems. Furthermore, a uniform terminology should help avoid ambiguity and improve communication among people and teams practicing the software engineering discipline.

This paper is organized as follows: in Section 2, we describe the Essence Kernel and its full set of elements; in Section 3, we present a review of terminology problems in some domains; in Section 4, we improve the Essence standard by applying terminology unification; and finally, we discuss some conclusions and future work in Section 5.

2. Theoretical framework

Growth of the software engineering industry has launched the need for creating new development teams with skills enough to supply high-quality, on-time, and on-budget software systems for covering the industry demand [1]–[5]. Teams are creating their own elements—methods and practices—in order to fulfill this purpose. Such elements are intended to provide guidance to processes and objects to be used on the methods [1], [6], [7], so, in this way, teams can produce high quality software systems. However, some circumstances when running the software engineering endeavor—e.g., tight deadlines, poor cost estimation, quality demands, volatile requirements, etc.—lead teams to entirely misuse their original methods and practices. They are often forced to tailor their own methods and practices and learn new ways of working [8], so new knowledge and experience gained is abandoned. Consequently, knowledge transference among teams is getting harder.

SEMAT is an initiative aimed to meet the software engineering challenges we face nowadays. As a way to reach this goal, the SEMAT community is promoting a scalable, actionable standard—called the Essence—based on proven principles and best practices [1], [6], [7]. Such a standard provides support to make the software engineering method and practice transference easier, tailoring, measuring, and comparison.

The Essence standard [6], [19] includes a set of elements and a structured language—known as the Es- sence kernel and language for software engineering methods. Elements contained in the Essence kernel are intended to be constructs for covering all software engineering endeavors [6], [7]: alphas (attributes for assessing the health and progress of the software engineering endeavor, by using states and checklists); activity spaces (groups of activities always present in any software

engineering endeavor); and competencies (what is needed for performing the work, including abilities and knowledge) [6]. Such constructs are grouped into three areas of concern: customer (related to the opportunity and the stakeholder), solution (a technical area including requirements and the software system itself), and endeavor (related to the work, the team, and the way of working) [6].

The Essence standard has inspired work on some areas like teaching [20], software startups [21], the anatomy of software requirements [22], and adaptive software engineering [23].

3. Background

Cabré [10] establishes two degrees for the adequacy of a theory: observational, for describing the observed data, and descriptive, for describing the non-observed data. Theories with the two degrees are predictive. The lack of uniformity prevents a theory from achieving the degrees of adequacy. Goosen [11] argues that uniformity is yet to be reached in the nursing terminology, so comparisons are difficult to achieve about data over time and documents coming from different sources.

According to Nedobity [9], concepts and conceptual systems are representations of reality and elaborations of the world. Thus, teams have created specialized terminologies and conceptual systems allowing for communicating among themselves. Such terminologies are composed of concepts representing objects of the world—concepts represent physical objects, as well as properties and relations of those objects. Terminology differences associated to a concept are the result of the diversity of languages. Nedobity [9] believes that deficient terminologies endanger the information flow among people and machine-to-machine communication. Similarly, several designations for the same object are results of the alternative usages of an object [10]. However, Cabré, citing Wüster, states that scientists and technicians should have a characterized and unambiguous terminology [10]. She claims ambiguity from technical languages can be removed by unifying terminology, so in this way scientists and technicians can establish an effective communication.

Terminology problems are common to different domains. Goosen [11] reports difficulties for mapping concepts between nursing terminologies and classifications, even though some international standards are defined for such a discipline. Goosen [11] shows that cross-mapping is still possible, but lack of uniformity can be demonstrated in this domain. Slisko and Dikstra [12] reveal the lack of a well-defined scientific language waiting to be used in research and teaching; they exemplify the problems with some terms related to science, which arise from the misconception and usage of terms in science as a need for improving teaching with a uniform, defined terminology. Dryden [13] summarizes all the effort devoted to standardizing terminology related to the archive domain and the main difficulties linked to this task: different languages, technological change, and the recent emergence of this discipline as a professional field. Sauberer *et al.* [14] claim that terminology should be self-explanatory in engineering environments, since time for discussions about the meaning of the terms can delay the work to-be-done; they suggest the development and implementation of a corporate terminology policy and they exemplify them in the context of the automotive industry. However, such policies are difficult to spread among several companies, thus causing lack of uniformity in the terms used in the whole environment. Finally, Misra [15] shows that the problems related to the usage of terms in requirements specification lead to misunders-

tanding of such specification along the software development lifecycle; he advocates for a careful review of specifications in order to generate a term-alias glossary for document interpretation. Even though this is a kind of terminology unification, we need to select a unique term for representing the concepts instead of dealing with all the possible aliases of a term.

Sonneveld and Loening [24] assert that new terms are constantly being created to express new ways of working. Such assertion makes sense in the software engineering discipline too. New methods, practices, and thinking frameworks are constantly created, and they commonly result from transformations made to existing methods, practices, and thinking frameworks. Such ways of working bring up the creation of new terms in the software engineering discipline. However, the theory fails to provide an unambiguous standard where the minimal parts forming either a method or a practice are terminologically uniform. Consequently, the theory is unsupportive of new concepts, i.e., we can say—according to Cabré [10]— that the Essence standard is descriptively and predictively inadequate. Elkin [25] says that the relationship between concepts should be uniform across parallel domains within the terminology. We look for such uniformity for the Essence standard in the next Section by using terminology unification.

4. Solution

As we previously mentioned, problems related to the uniformity of terminology should be solved, so software engineering teams can use the same terminology for improving their technical communication. In this Section, we propose an improvement to the Essence standard by solving such problems. We apply a four-stage method described in the following sub-sections.

4.1. Selection of base models and definitions

Ward [26] develops a method for addressing terminology problems. The first three stages of the method are devoted to developing a taxonomy and a glossary to be used for detecting terminology problems in the fourth stage. Similarly, Goosen [11] employs the ISO reference terminology model for nursing diagnosis and some definitions coming from different standards. Consequently, we select some base models and definitions for applying the remainder of the method. Since the Essence standard [6] has structured models for alphas and activity spaces, we select such models (see Figs. 1 and 2) as the basis for our analysis. We also use the terms and definitions included in the fourth section of the Essence standard [6]. Some checklists of the alpha states are also reviewed [6].

A third model is selected in order to include expert judgement in the analysis. Morales-Trujillo *et al.* [27] reported a terminological analysis made to the Essence standard by using a pre-conceptual schema (see Fig. 3). In this figure, the terms of the Essence standard are colored in blue and yellow. Since the pre- conceptual schema was validated by some of the Essence standard authors, we can use it as a pivot for evaluating some of the terms used throughout the models, definitions, and checklists of the Essence standard.

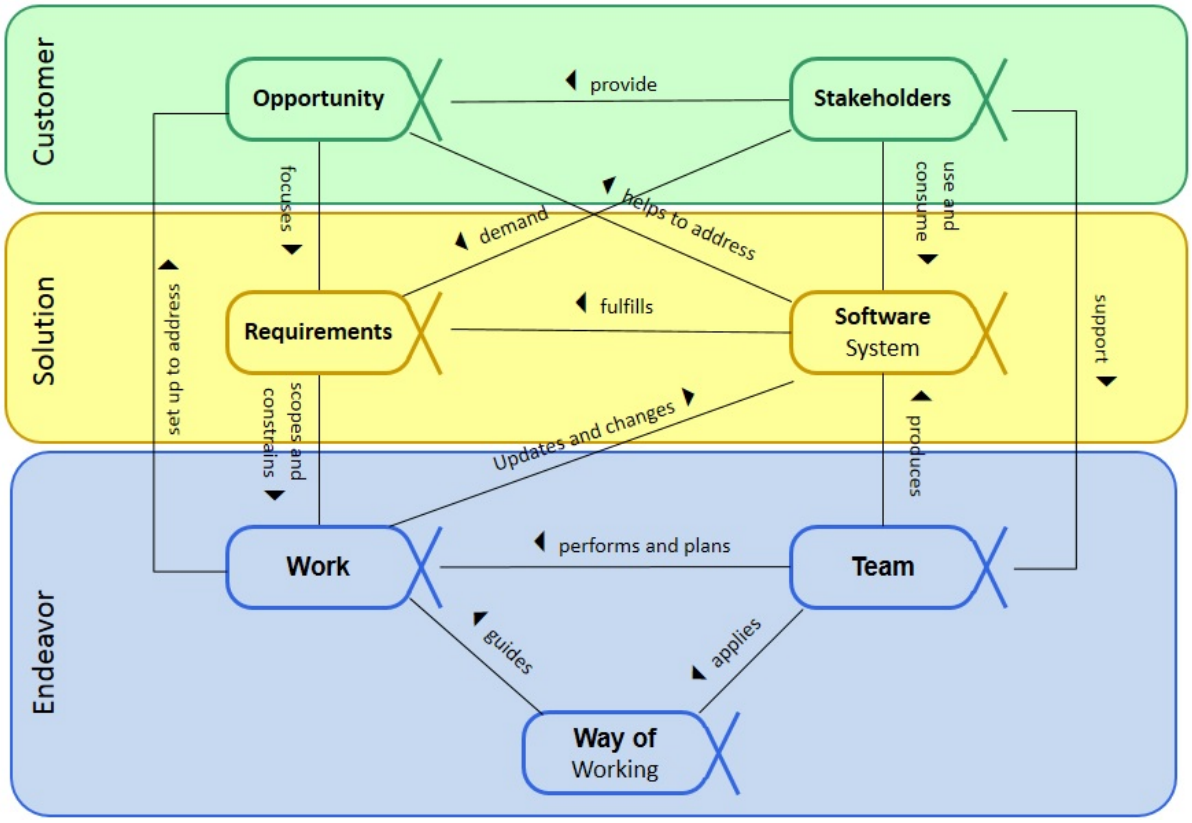

Figure 1. Alphas of the Essence standard and their relationships [6]

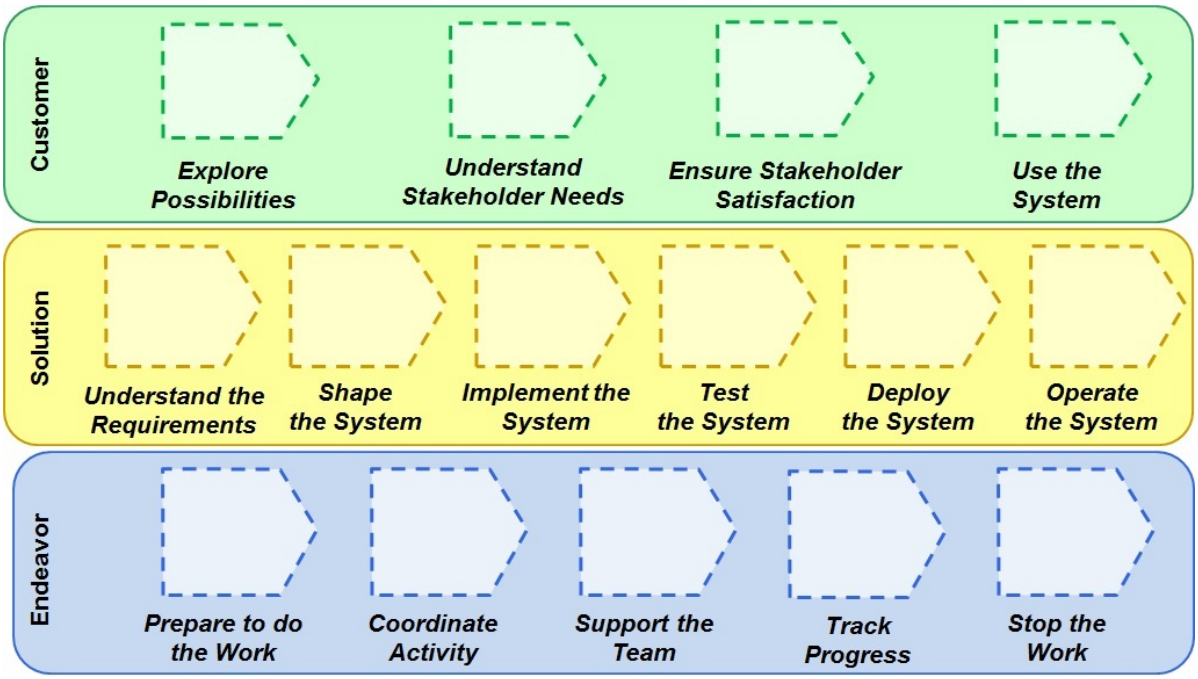

Figure 2. Activity spaces of the Essence standard [6]

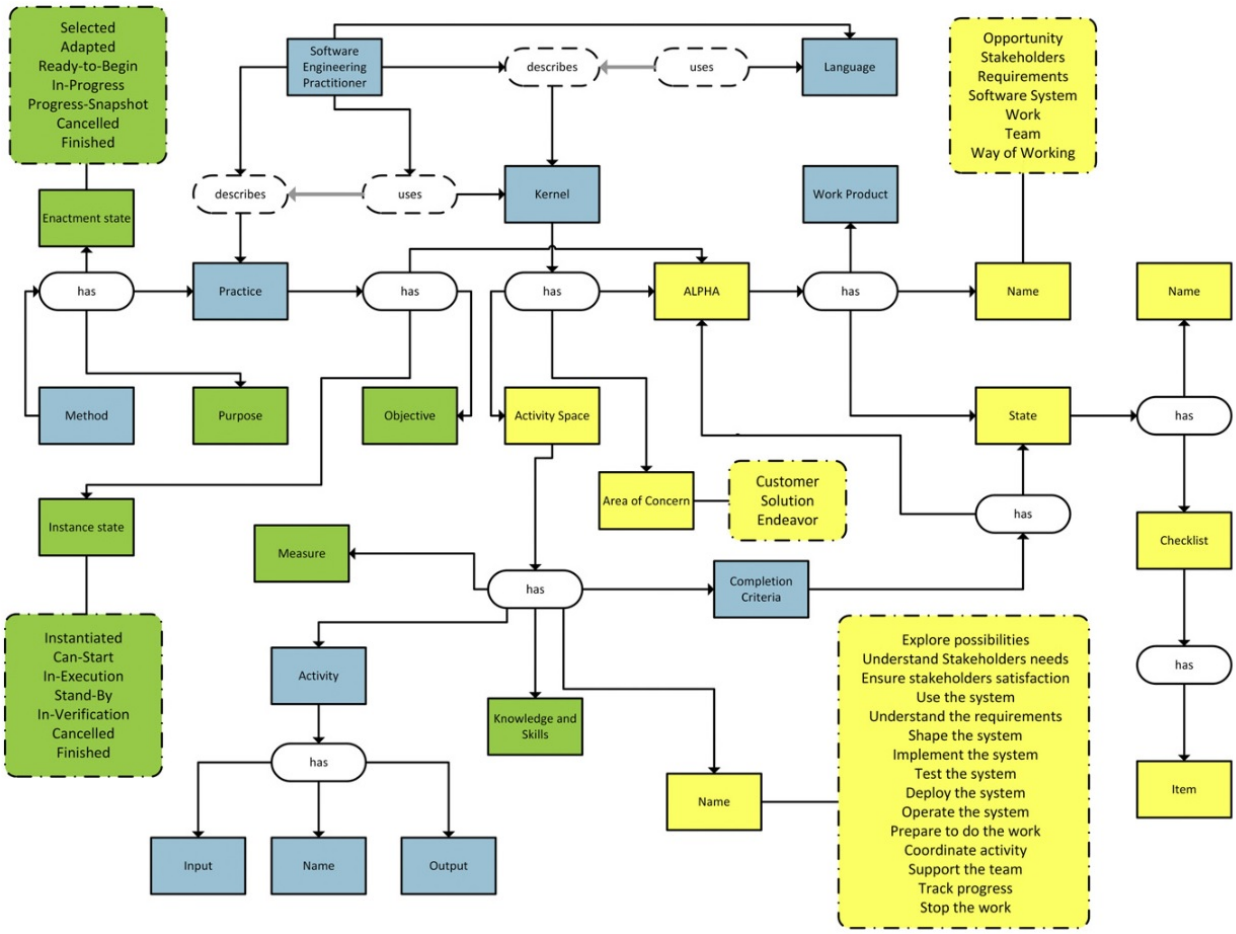

Figure 3. Pre-conceptual schema of the Essence standard [27]

4.2. Identification of terminology problems

We review the selected definitions, checklists, and models in order to establish the usage of terms in the Essence standard. Initially, a specific sample of terms and definitions from the Essence standard is presented in Table I.

Table I. Sample of identified terms and definitions from the Essence standard [6]

Term	Definition
Activity space	A placeholder for something to be done in the software engineering endeavor. A placeholder may consist of zero to many activities.
Opportunity	The set of circumstances that make it appropriate to develop or change a software system.
Work item	A piece of work that should be done to complete the work. It has a concrete result and it leads to either a state change or a confirmation of the current state. A work item may or may not have any related activity.

Along the Essence standard, *activity space* is defined as "descriptions of the challenges a team faces when developing, maintaining, and supporting software systems" [6, p. 15]. However, such a description is excluded from the *activity space* definition. This lack of uniformity can be found

in many of the Essence standard terms and definitions. Relationships of the *opportunity* alpha are excluded from the definition. The term *opportunity* should contain alpha relationships associated to the *opportunity* alpha for providing a better understanding of its definition. Finally, the Essence standard [6] exhibits a deeper definition of the *work item* by means of the alternative usages of this kind of element—*e.g.*, elements in which the work is broken down, elements with clear definitions of done, user stories from a sprint backlog. When the definition of *work item* is compared to the *work product* definition—"an artifact of value and relevance for a software engineering endeavor; a document or a piece of software" [6, p. 92]—we realize that work item and *work product* are the same term. *Work product* is defined as an element representing "concrete things to work with, providing evidence for the states an alpha is in" [6, p. 69]. Furthermore, a deeper definition of *work product* can be inferred from the alternative usages of this kind of element—*e.g.*, document where the user requirements are documented, use cases, product backlog or sprint backlog. Therefore, the term *work item* should be renamed as *work product*. Also, we need a definition of *completion criteria* (see Fig. 3), since the definition of *work item* includes the phrase "complete the work" [6, p. 7].

Another source of terminology problems can be related to constructs of the Essence standard like the *alpha state checklists*. Lack of uniformity can be found in several *alpha state checklists*; the *value established* state of the *opportunity* alpha, *seeded* state of the *team* alpha, and *bounded* state of the *requirements* alpha are shown in Table II. The *value established* state of the *opportunity* alpha is defined as "the value of a successful solution has been established" [6, p. 27], but *solution* is an area of concern and the closest construct for *solution* in this context is the *software system* alpha. In fact, *solution* and *software system* seem to be interchangeable in the context of the *alpha state checklist* in Table II when the *value established* state is detailed. The same problems can be detected in the other *alpha state checklists* included in Table II with terms like *mission, mechanisms, commitment, governance rules, leadership model, success, prioritization,* and *assumptions*. Some mentions of other constructs are unclear. For example, *leadership* is a competency of the *endeavor* area of concern with some levels, so probably the expression ***leadership model is selected*** is intended to be interpreted as ***leadership level is determined***. The *requirements alpha* is another example of the misuse of the terminology. The checklist item ***the way the requirements will be described is agreed upon*** is related to a requirements state, but ***described*** is outside the set of the requirement states—*i.e., conceived, bounded, coherent, acceptable, addressed*, and *fulfilled*.

The aforementioned terminology problems could generate mistakes in the way we assess the progress of the team by using the Essence standard. So, at some point, the team could think they are in an advanced state of a certain alpha when they should be in a previous one. Such mistakes can lead the team to a work bottleneck as the software engineering endeavor time goes on. Terminology problems generated in the *alpha checklists* can lead to completeness problems affecting the uniformity of the theory.

Regarding the models selected in the previous stage, terminology problems can also arise from the relationships among *alphas* shown in Fig. 1 and *alpha descriptions* provided by the Essence standard [6]. We identify *alpha relationships* with terminology problems in Table III. The relationship between the alphas *opportunity* and *requirements* exhibits terminology problems with the alpha description provided in the Essence standard [6]. *Opportunity* is "the set of circumstances that makes it appropriate to develop or change a software system" [6, p. 5] and also "the opportunity

Table II. Full Checklists for the *value established, seeded*, and *bounded* states of the Essence kernel [6]

State	Checklist
Value established	The value of addressing the opportunity has been quantified either in absolute terms or in returns or savings per time period (e.g., per annum). The impact of the solution on the stakeholders is understood. The value that the software system offers to the stakeholders that fund and use the software system is understood. The success criteria by which the deployment of the software system is to be judged are clear. The desired outcomes required of the solution are clear and quantified.
Seeded	The team mission has been defined in terms of the opportunities and outcomes. Constraints on the team's operation are known. Mechanisms to grow the team are in place. The composition of the team is defined. Any constraints on where and how the work is carried out are defined. The team's responsibilities are outlined. The level of team commitment is clear. Required competencies are identified. The team size is determined. Governance rules are defined. Leadership model is selected.
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.

articulates the reason for the creation of the new, or changed, software system (. . .) It represents the team's shared understanding of the stakeholders' needs, and helps shape the requirements for the new software system by providing justification for its development" [6, p. 17]. As a matter of fact, the description of the actual relationship between the alphas *opportunity* and *requirements*—*i.e.*, ***focuses***—is excluded from the description of the alpha, and other relationships are excluded from Table III—*i.e.*, ***shape***. Terminology problems arising from the relationship among *alphas* and *alpha descriptions* are represented by the existing and excluded relationships, thus leading to a completeness problem—*e.g.*, the description of opportunity makes it clear that the relationship ***opportunity makes appropriate creates, updates, or changes software system*** was omitted by the authors of the Essence standard [6].

The Essence standard should provide a detailed description of the challenges faced by a team when running *activity spaces* of a software engineering endeavor. However, if such elements exhibit problems, the team is unable to address those challenges in a proper way. In fact, the usage of several designations for the same object or action can be mistaken by teams and produce undesired results. Moreover, terminology problems lead to misunderstanding the completeness of the theory, but we can realize that the theory is incomplete—so the completeness problem should be solved—by addressing terminology problems. We can identify activity spaces with terminology

problems in Table IV.

Table III. Alpha relationships with identified terminology problems [6]

Alpha	Alpha	Relationship
Opportunity	Requirements	Focuses
Work	Software System	Updates and changes
Team	Way of Working	Applies
Way of Working	Work	Guides
Team	Software System	Produces

Table IV. Activity spaces with identified terminology problems [6]

Name	Area of concern
Explore possibilities	Customer
Understand Stakeholder Needs	Customer
Use the System	Customer
Shape the System	Solution
Implement the System	Solution
Test the System	Solution
Deploy the System	Solution
Operate the System	Solution
Coordinate Activity	Endeavor
Support the team	Endeavor
Track Progress	Endeavor

The phrase ***explore possibilities*** comprises a verb and a noun. In order to provide an accurate name for the *activity space*, the noun should be included in the specialized terminology defined by the Essence standard [6], as expressed in the pre-conceptual schema of Fig. 3, and ***possibility*** is outside such terminology. Something similar occurs to ***system***. Be advised that system was considered a name for the software system alpha, but it was rejected because it "was considered to be too general" and "the consensus was that all engineering disciplines produce some kind of system, and therefore software engineering needs to produce something more specialized than just a system" [19, p 11]. In this way, the activity spaces related to the software system should be named after the name of the alpha. The same applies for ***understand stakeholder needs. Need*** (absent from Fig. 3) was also considered a name for an alpha—in this case the requirements alpha included in the Essence standard—but it was rejected because it was "considered too confusing when compared and contrasted with requirements" [8, p 19].

Terminology problems associated with *coordinate activity, support the team*, and *track progress* are related to *activity space* descriptions. The description of *coordinate activity* is to "co-ordinate and direct the team's work," and "this includes all ongoing planning and re-planning of the work,

and adding any additional resources needed to complete the formation of the team" [6, p 19], the description of *support the team* is to "help the team members to help themselves, collaborate, and improve their way of working" [6, p 20]; and *track progress* is to "measure and assess the progress made by the team" [6, p 20]. Accordingly, the *activity space* names should be re-defined in a way to be consistent with the actual description of them.

4.3. Unification of terms

Terminology unification is first applied to the definitions in Table I. We need to add information and make uniform use of terms, as we propose in Table V. Regarding the definition of *activity space* and the *opportunity* definition, we need to add some information for making uniform usage of the terms in the standard. Also, as we stated before, *work product* and *work item* seem to be the same construct according to their definitions, so we propose to create just one single definition and use the term *work product* throughout the Essence standard. We also propose to add a definition to the completion criteria, since this is a term used several times in the standard.

Table V. Sample of Essence standard terms and definitions with terminology problem resolution

Term	Definition	Proposal
Activity space	A placeholder for **something to be done** in the software engineering endeavor. A placeholder may consist of zero to many activities.	A placeholder for **activities** to be done in the software engineering endeavor. A placeholder may consist of zero to many activities.
Opportunity	The set of circumstances that makes it appropriate to **develop** or change a software system.	The set of circumstances that makes it appropriate to **create, update**, or change a software system.
Work product	(Work item in the standard) A piece of work that should be done to complete the work. It has a concrete result and it leads to either a state change or a confirmation of the current state. Work item may or may not have any re- lated activity. (Work product implicit in the standard) Concrete things to work with, providing evidence for the states an alpha is in. An artifact of value and relevance for a software engineering endeavor; a document or a piece of software	Concrete **artifacts** that should be done to complete the work, providing evidence for the states an alpha is in. A work product is the result of a related activity.
Completion criteria	(Missing in the standard)	Definition of the conditions for an activity space to be considered complete. It is expressed in terms of alpha states.

Terminology problems in the *alpha checklists* are solved by changing non-standard terms, excluding redundant information, and including some missing information in the checklist items (see Table VI).

Table VI. Proposed checklists for the value established, seeded, and bounded states of the Essence kernel

State	Checklist
Value established	The value of addressing the opportunity has been quantified either in absolute terms or in returns or savings per time period ($e.g.$, per annum). ***The stakeholders are satisfied in use***. The value that the software system offers to the stakeholders that fund and use the software system is understood. ***The completion criteria to deploy the software system are clear***. ***The work products required of the operational software system are clear***.
Seeded	The team mission has been defined in terms of the opportunities and ***work products***. Constraints on the team's ***way of working*** are known. Mechanisms to ***form*** the team are defined. The ***form*** of the team is defined. Any constraints on where and how the work is under control are defined. ***Management and leadership of the team are clear***. ***Required competency levels are identified***.
Bounded	The stakeholders ***to be satisfied in use with the software system*** are ***recognized***. The stakeholders agree on the purpose of the software system. ***The completion criteria to develop the software system are clear***. The stakeholders ***are in agreement about the value that the software system offers***. The way the requirements ***will be bounded*** is agreed upon. ***Management of the requirements is clear***.

Some of the terms we are proposing to change (and by which ones) are: new system (software system), solution (software system), success criteria (completion criteria), operation (way of working), grow (form), composition (form), impact on the solution is understood (satisfied in use), identified (recognized), shared understanding (in agreement), extent (value), and described (bounded). Some of the changes obey to definitions of the Essence standard constructs; for example, we change the term *impact on the solution is understood* because no states are named in this way. However, we have a clear definition of the *satisfied in use* state of stakeholders. Something similar happens to the term *described*, which is related to requirements; the next state to such a description is the *bounded* state.

The redundant information we propose to exclude is the following:

- ***The impact of the solution on the stakeholders is understood*** is a description included in the definition of the ***satisfied in use*** state of stakeholders.
- ***Team responsibilities are outlined, the level of team commitment is clear, the team size is determined, governance rules are defined, and leadership model is selected*** are checklist items related to our proposal: ***management and leadership of the team are clear***. In this case, we are using two competencies of the endeavor area of concern for covering all redundant topics.
- ***The mechanisms for managing the requirements are in place, the prioritization scheme is clear, constraints are identified and considered, and assumptions are clearly stated*** are

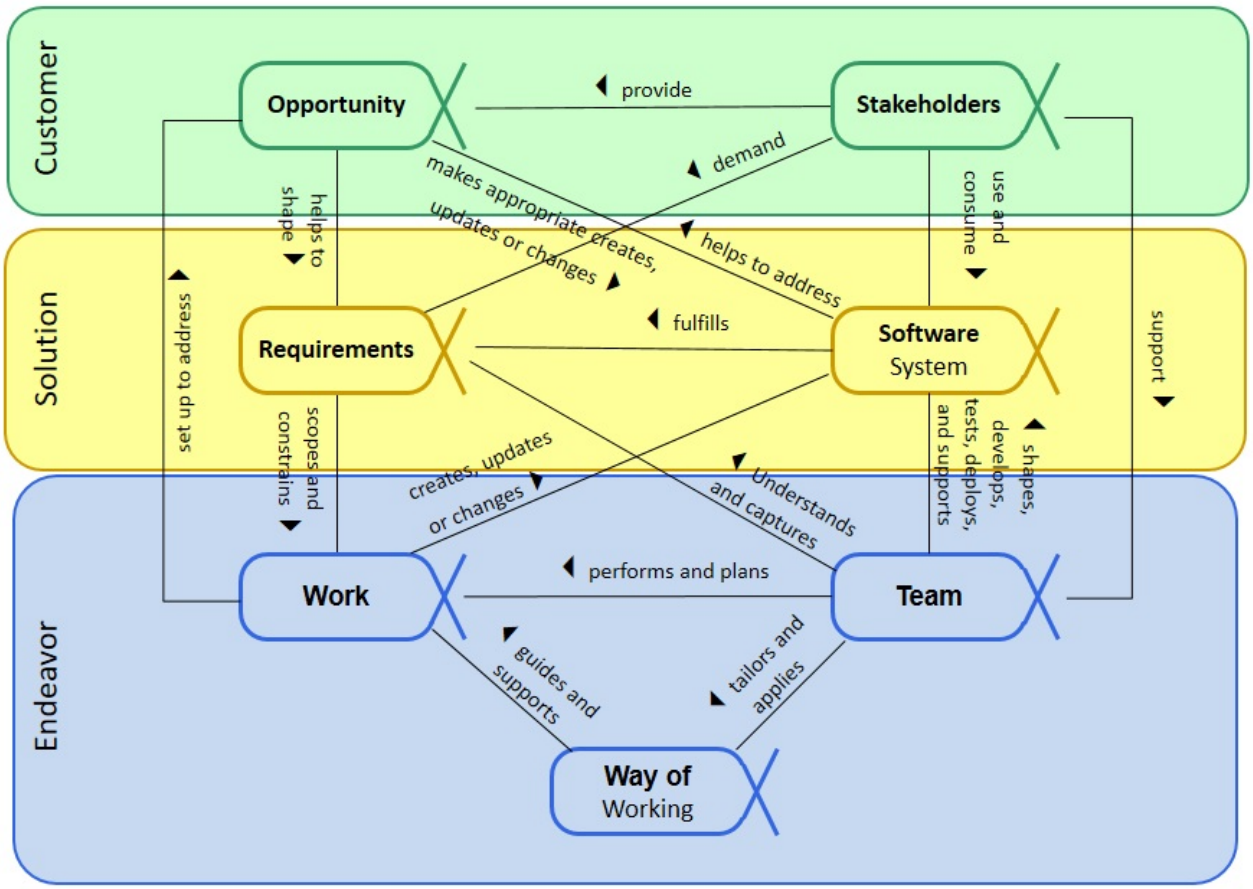

Figure 4. Proposal for solving terminology problems in the alpha relationships

checklist items related to our proposal: ***management of the requirements is clear.***

The missing information we are including is the following: the work products related to the solution are those related to the *software system* in the *operational* state; we always know the *required competencies*, since the Essence standard only recognizes six of them, but what we need to identify is the *competency level* required by the team members.

We propose a solution to terminology problems of the alpha relationships in Fig. 4. Also, the details of the alpha relationships are summarized in Table VII. Most of the changes are proposed after reviewing the activity spaces defined in the Essence standard. For example, *produces* is very short for describing how the team is related to the software system, since we have five activity spaces related to software system in the *solution* area of concern: ***shape, develop, test, deploy***, and ***support***. Some other changes are related to the definitions included in the Essence standard. For example, *way of working* is defined as "the tailored set of practices and tools used by a team to guide and support their work" [6, p. 57], so the team ***tailors*** and ***applies*** the way of working, and the way of working ***guides*** and ***supports*** the work. Some other relationships from Fig. 1 are omitted, but we can recover them by reviewing some other constructs of the Essence standard. For example, we find that the team ***captures*** and ***understands*** the requirements in the definition of the ***analysis*** competency.

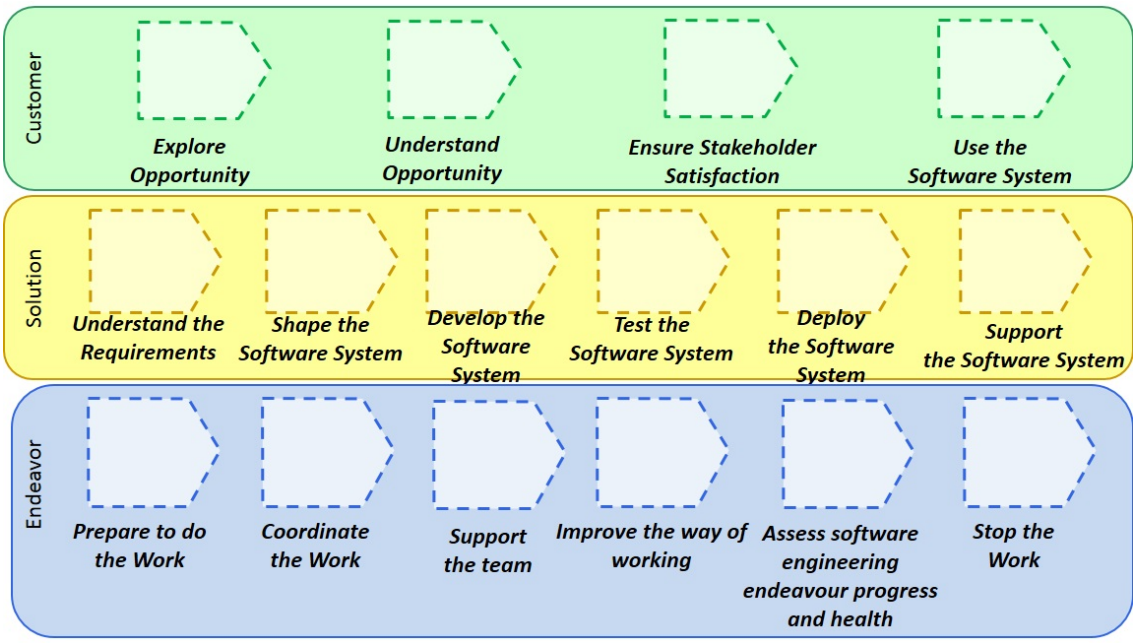

Figure 5. Proposal of names for activity spaces in order to solve terminology problems

We propose a solution for terminology problems of the activity spaces in Fig. 5 and Table VIII. As we said before, activity spaces represent the placeholders for activities to be done in a software engineering endeavor, *i.e.*, what the team should perform to produce a software system. Thus, changes made to activity spaces should be reflected in the elements related to the alphas. Some of the changes we propose for activity spaces are related to the names of the alphas involved. We discussed in Section 4.2 how the name ***system*** has been rejected and the name ***software system*** was adopted. However, the older name is still applied in five out of six activity spaces belonging to the *solution* area of concern. This assertion is ratified by the experts by including ***software system*** as an alpha in Fig. 3. For this reason, we propose changing ***shape the system for shape the software system***. In the same way, we propose changing ***possibilities*** and ***stakeholder needs*** for the adequate alpha name (see Figs. 1 and 3): ***opportunity***. Some other changes are related to the definitions included in the Essence standard. For example, the *team* alpha is defined as "the group of people actively engaged in the development, maintenance, delivery, or support of a specific software system" [6, p. 6]; in this way, we propose changing the verb ***implement*** for ***develop*** and the verb ***operate*** for ***support***. Finally, we propose changing ***coordinate activity*** for ***coordinate the work***, since ***activity*** is not considered an alpha and the next name related to it as alpha is ***work***.

Solving terminology problems within the Essence standard implies that it is incomplete. When we solve such problems, some terms are still outside of the terminology provided by the Essence standard. This leads to the definition of new elements and terms. Some evidence about such definition is the ***support the team*** activity space. When we solve the terminology problems we need a new activity space named ***improve the way of working*** (see Fig. 5), which is used to describe activities related to the way of *working* alpha for promoting the advance in the way of *working* states. Given the above, the activity space ***completion criteria*** (see the proposed definition in Table V) are compromised—the completion criteria include the ***collaborating*** state of the *team* alpha and ***in place*** state of the *way of working* alpha—since the description is outside the work to be done

Table VII. Proposed names for the relationship between alphas

Alpha	Alpha	Relationship	Proposal
Opportunity	Requirements	Focuses	Helps to shape
Work	Software System	Updates and changes	Creates, updates, or changes
Team	Way of working	Applies	Tailors and applies
Way of working	Work	Guides	Guides and supports
Team	Software System	Produces	Shapes, develops, tests, deploys, and supports
Opportunity	Software System		Makes appropriate creates, updates, or changes
Team	Requirements		Understands and captures

Table VIII. Sample of Essence standard terms and definitions with consistency and terminological problems resolution

Name	Area of concern	Proposal
Explore possibilities	Customer	Explore Opportunity
Understand stakeholder needs	Customer	Understand opportunity
Use the system	Customer	Use the software system
Shape the system	Solution	Shape the software system
Implement the system	Solution	Develop the software system
Test the system	Solution	Test the software system
Deploy the system	Solution	Deploy the software system
Operate the system	Solution	Support the software system
Coordinate activity	Endeavor	Coordinate the work
<None>	Endeavor	Improve the way of working
Track progress	Endeavor	Assess software engineering endeavor progress and health

for achieving the checklists associated to the collaborating state of the team alpha. Based on such facts, we propose the redefinition of the *support the team* and *improve the way of working* activity spaces as follows:

- *Support the team*
  - Description: Support the team to make it work as a cohesive unit, make the communication open and honest, inform each other and focus on achieving the team mission [6].
  - Input: Team.
  - Entry criteria: Team::Formed.
  - Completion criteria: Team::Collaborating.
- *Improve the way of working*

- Description: "Help the team members to help themselves, collaborate, and improve their way of working" [6, p. 20].
- Input: Way of Working.
- Entry criteria: Team::Formed, Way of Working::Foundation Established.
- Completion criteria: Way of Working::In place.

4.4. Measurement of the gap between the current standard terms and the proposed changes

Misra [15] proposes a combination of a latent semantic analysis and a dissimilarity degree between two-word chains as a final stage of the terminological inconsistency analysis of natural language requirements. Similarly, Dalpiaz *et al.* [28] propose semantic similarity as a measure of the relatedness of two terms when analyzing terminological problems in a specification. Consequently, we include this fourth stage in our method and select the lexical semantic relatedness [29] for evaluating changes in terms and the Levenshtein distance for evaluating changes in word chains as a way to measure the proposed changes and their impact in the Essence standard.

We calculate the distance between the original and the proposed terms by using lexical semantic relatedness [29], a measure of how two words are related in meaning. To this effect, we use the online calculator included in `www.olesk.com` and summarize the results in Table IX. Even though some of the meanings are close (more than 90 %), we can see from Table IX we are improving the accuracy in the terminology by using the right words.

As Misra [15] suggests, we use the Levenshtein distance for evaluating the smaller number of insertion, deletion, and substitution operations required to change one word chain to the other. We also compare the dissimilarity as a percentage of the longer word chain, as we summarized in Table X. We calculate the Levenshtein distance by using the calculator included in `https://es.planetcalc.com/1721/?language_select=es`. As you can see from Table X, lower numbers of dissimilarity are associated with shorter distance between word chains. The usage of the same word chains is zero, the lowest number of dissimilarity. Again, we are improving the accuracy of the Essence standard terminology by adding uniform information into some constructs of the standard.

Due to the space requirements of this paper, we exemplify the terminology problems the Essence standard exhibits in the current version. Be advised that we have selected some of the constructs included in the standard and just a small number of each construct. For example, the standard has 27 reported definitions, and we work with just three of them. We work with three out of 41 states reported in the standard. The coverage is bigger in the case of the alphas and the activity spaces, but we demonstrated that terminology problems are linked to many constructs of the Essence standard. Fortunately, as we propose in this paper, the solutions to such problems can be easily achieved and they can drive improvements in accuracy, as we show with the lexical semantic relatedness and dissimilarity we calculate. For this reason, we strongly believe the guidelines we propose in this paper could help to solve the problems in question.

Table IX. Summary of lexical semantic relatedness

Original term	Proposed term	Lexical semantic relatedness (%)
Item	Product	75
System	Software system	92
Solution	Software system	33
Success	Completion	50
Operation	Way of working	82
Grow	Form	91
Composition	Form	90
Understood	Satisfied in use	41
Identified	Recognized	87
Shared understanding	In agreement	41
Extent	Value	91
Described	Bounded	33
Focuses	Help to shape	56
Possibility	Opportunity	92
Stakeholder need	Opportunity	49
Implement	Develop	45
Operate	Support	56
Activity	Work	92

Table X. Summary of lexical semantic relatedness

Original word chain	Proposed word chain	Levenshtein distance	Dissimilarity (%)
Updates and changes	Creates, updates, or changes	10	36
A placeholder for something to be done in the software engineering endeavor	A placeholder of the essential things to do in the software engineering endeavor	20	25
The set of circumstances that makes it appropriate to develop or change a software system	The set of circumstances that makes appropriate to create, update, or change a software system	16	17
Concrete things to work with, providing evidence for the states an alpha is in	Concrete things that should be done to complete the work, providing evidence for the states an alpha is in	34	32
Applies	Tailors and applies	13	68
Guides	Guides and supports	13	68
Produces	Shapes, develops, tests, deploys, and supports	42	91

5. Conclusions and future work

In this paper we proposed the solution to some terminology problems in the Essence standard. We used a method based on terminology unification in order to intervene constructs like definitions, alpha state checklist items, relationships between alphas, and activity spaces. We identified main problems such as the use of non-standard terms for naming standard elements, the addition of redundant information, and the lack of pertinent information related to some constructs. After solving the aforementioned problems, we evaluated the accuracy of our solution with two metrics related to semantic and morphological distance. Even though we sampled the problems with some constructs, we believe the presented guidelines could help to solve other problems in the Essence standard. Also, as a result of such problem resolutions, we proposed two new alpha relationships excluded from the Essence standard, one term definition, and one new activity space. We redefined three terms, three alpha state checklists, five alpha relationships, and 11 activity spaces. Such problem resolution provides a better understanding of the Essence standard. Consequently, software engineering practitioners could have a specialized terminology that allows unambiguous communication with each other. Also, we contribute to reducing the gaps between the real progress of the team and the progress assessed by using the terminology defined in the Essence standard.

As future work, we can define the following lines of work:

- Developing a focus group in order to validate the proposed changes with practitioners and experts of the software engineering field of knowledge.
- Applying the solution to the rest of the constructs defined in the Essence standard.
- Completing the terms defined in the Essence standard by following the guidelines defined in this paper. We believe we should have more than the 27 current definitions of the Essence standard. Some constructs such as competency level, milestone, phase, etc. are still missing in the standard.
- Detecting other problems arising from terminology of the Essence standard. We can suppose we can discover more additions and deletions for the standard while reviewing the rest of the constructs.
- Applying the method followed in this paper to other bodies of knowledge and standards related to software engineering—e.g., the Software Engineering Body of Knowledge SWEBoK—and other disciplines like the Project Management Body of Knowledge PMBoK.

References

[1] I. Jacobson, "Discover the Essence of Software Engineering," *CSI Communications*, vol. 35, no. 4, pp. 9-11, 2011. ↑214, 215

[2] V. Gaorousi, A. Coskuncay, A. Betin-Can, and O. Demirörs, "A survey of software engineering practices in Turkey," *J. Sys. Soft.*, vol. 108, pp. 148-177, 2015. https://doi.org/10.1016/j.jss.2015.06.036 ↑214, 215

[3] C. Riemenschneider, B. Hardgrave, and F. Davis, "Explaining Software Developer Acceptance of Methodologies: A Comparison of Five Theoretical Models," *IEEE Trans. Soft. Eng.*, vol. 28, no. 12, pp. 1135-1145, 2002. https://doi.org/10.1109/TSE.2002.1158287 ↑214, 215

[4] A. Arora and S. Athreye, "The software industry and India's economic development," *Inf. Econ. Pol.*, vol. 14, no. 2, pp. 253-273, 2002. https://doi.org/10.1016/S0167-6245(01)00069-5 ↑214, 215
[5] J. Verner, M. Babar, N. Cerpa, T. Hall, and S. Beecham, "Factors that motivate software engineering teams: A four country empir- ical study," *J. Sys. Soft*, vol. 92, pp. 115-127, 2014. https://doi.org/10.1016/j.jss.2014.01.008 ↑214, 215
[6] Object Management Group "Kernel and Language for Software Engineering Methods (Essence) version 1.2", 2018. [Online] Available: http://www.omg.org/spec/Essence/ ↑ 214, 215, 216, 217, 218, 219, 220, 221, 222, 223, 225, 226, 227, 228
[7] I. Jacobson, P.-W. Ng, P. McMahon, I. Spence, and S. Lidman, *The Essence of Software Engineering. Applying the SEMAT Kernel*. Crawfordsville, IN, USA, Addison-Wesley Professional, 2013. ↑214, 215
[8] L. Singer and K. Schneider, "Influencing the Adoption of Software Engineering Methods Using Social Software," in *Int. Conf. Soft. Eng. (ICSE)*, 2012, pp. 1325-1328. https://doi.org/10.1109/ICSE.2012.6227088 ↑214, 215, 222
[9] W. Nedobity, "The General Theory of Terminology: A Basis for the Preparation of Classified Defining Dictionaries," *J. Dict. Soc. North America*, vol. 5, pp. 69-75, 1983. https://doi.org/10.1353/dic.1983.0000 ↑214, 216
[10] M. T. Cabré, "Theories of terminology, Terminology:" *Int. J. Theor. App. Iss. Spec. Comm.*, vol. 9, no. 2, pp. 163-199, 2003. https://doi.org/10.1075/term.9.2.03cab ↑214, 215, 216, 217
[11] W. Goosen, "Cross-mapping between three terminologies with the international standard nursing reference terminology model," *Int. J. Nurs. Term. Class.*, vol. 17, no. 4, pp. 153-164, 2006. https://doi.org/10.1111/j.1744-618X.2006.00034.x ↑214, 215, 216, 217
[12] J. Slisko and D. Dikstra, "The role of scientific terminology in research and teaching: is something important missing?," *J. Res. Sci. Teach.*, vol, 34, no. 6, pp. 655-660, 1997. https://doi.org/10.1002/(SICI)1098-2736(199708)34:6<655::AID-TEA7>3.0.CO;2-M ↑215, 216
[13] J. Dryden, "A tower of Babel: standardizing archival terminology," *Archival Sci.*, vol. 5, pp. 1-16, 2006. https://doi.org/10.1007/BF02510902 ↑215, 216
[14] G. Sauberer, B. Nájera, J. Dreßler, K. Schmitz, P. Clarke, and R. O'Connor, "Do we speak the same language? Terminology strategies for (software) engineering environments based on the Elcat model—innovative terminology e-learning for the automotive industry," *EuroSPI 2017, CCIS*, vol. 748, pp. 653-666, Ostrava, 2017. https://doi.org/10.1007/978-3-319-64218-5_54 ↑215, 216
[15] J. Misra, "Terminological inconsistency analysis of natural language requirements," *Inf. Soft. Tech.*, vol. 74, pp. 183-193, 2016. https://doi.org/10.1016/j.infsof.2015.11.006 ↑215, 216, 228
[16] M. Varulenko and K. Meijnyk, "Term properties and modern terminological systems development," *Int. Ins. Term. Res. IITF J.*, vol. 24, pp. 29-38, 2013-2014. ↑215
[17] C. Zapata, R. Arango, and L. Jiménez, "Mejoramiento de la consistencia entre la sintaxis textual y gráfica del lenguaje de SEMAT," *Polibits*, vol. 49, pp. 83-89, 2014. https://doi.org/10.17562/PB-49-10 ↑215
[18] R. Arango, "Especificación en OCL de los elementos del Núcleo de SEMAT", MSc. Thesis, Departamento de Ciencias de la computación y la decisión, Universidad Nacional de Colombia, Medellín, Colombia, 2016. ↑215
[19] Object Management Group, "Essence—Kernel and Language for Software Engineering Methods. Revised Submission", 2013. [Online] Available: http://www.omg.org/cgi-bin/doc?ad/2013-02-01 ↑215, 222
[20] I. Jacobson, H. Lawson, P. Ng, P. McMahon, and M. Goedicke. *The essentials of modern software engineering: free the practices form the method prisons!*. Milton Keynes, UK, ACM Books, Morgan & Claypool Publishers, 2019. https://doi.org/10.1145/3277669.3277673 ↑216
[21] K. Kemell, A. Nguyen-Duc, X. Wang, J. Risku, and P. Abrahamsson, "Software Startup ESSENCE: How Should Software Startups Work?," In *Fundamentals of Software Startups A*. Nguyen-Duc, J. Münch, R. Prikladnicki, X. Wang, and P. Abrahamsson, Cham, Ger., Springer, 2020, pp. 97-109. https://doi.org/10.1007/978-3-030-35983-6_6 ↑216
[22] B. Meyer, J. Bruel, S. Ebersold, F. Galinier, and A. Naumchev, "Towards an Anatomy of Software Requirements," *Lect. Notes Comp. Sci.*, vol. 11771, pp. 10-40, 2019. https://doi.org/10.1007/978-3-030-29852-4_2 ↑216
[23] J. Park, J. Jang, and E. Lee, "Theoretical and empirical studies on essence-based adaptive software engineering," *Inf. Tech. Man.*, vol. 19, pp. 37-49, 2018. https://doi.org/10.1007/s10799-016-0273-5 ↑216
[24] W. Sonneveld and K. Loening, "A terminologist's and a chemist's look at chemical neologisms," *Stand. Tech. Term.: Prin. Prac.*, vol. 2, pp. 23-28, 1988. https://doi.org/10.1520/STP29546S ↑217

[25] P. Elkin, "Theoretical foundations of terminology," In *Terminology and terminological systems, health informatics*, London, UK, Springer-Verlag, 2012, pp. 51-70. https://doi.org/10.1007/978-1-4471-2816-8_4 ↑217
[26] S. Ward, "The hierarchical terminology technique: a method to address terminology inconsistency," *Qual. Quant.*, vol. 46, pp. 71-87, 2012. https://doi.org/10.1007/s11135-010-9328-6 ↑217
[27] M. Morales-Trujillo, H. Oktaba, and M. Piattini, "The making of an OMG standard," *Comp. Stand. Interf.*, vol. 42, pp. 84-94, 2015. https://doi.org/10.1016/j.csi.2015.05.001 ↑217, 219
[28] F. Dalpiaz, I. van der Schalk, s. Brinkkemper, F. Aydemir, and G. Lucassen, "Detecting terminological ambiguity in user stories: tool and experimentation," *Inf. Soft. Tech.*, vol. 110, pp. 3-16, 2019. https://doi.org/10.1016/j.infsof.2018.12.007 ↑228
[29] R. Siblini and L. Kosseim, "Using a weighted semantic network for lexical semantic relatedness," *Proceedings of recent advances in natural language processing*, Hissar, 2013, pp. 7-13. ↑228

Carlos Mario Zapata-Jaramillo

Civil Engineer, Specialist in Information Systems Management, M.Sc. in Engineering, and PhD in Engineering focused on Systems, all degrees from the Universidad Nacional de Colombia at Medellín. Full Professor (Tenured) of the Computing and Decision Sciences Department from Universidad Nacional de Colombia at Medellín. Leader of the Computing Language Research Group. Areas of interest: software engineering, requirements engineering, computational linguistics, and didactic strategies for engineering teaching.
Email: cm zapata@unal.edu.co

Antony Henao-Roqueme

Systems and Informatics Engineer and MSc. in Systems Engineering, both degrees from Universidad Nacional de Colombia at Medellín; member of the Computing Language Research Group. Areas of interest: software engineering and massive data analysis.
Email: ajhenaor@unal.edu.co