Software quality measures and their relationship with the states of the software system alpha

Medidas de la calidad del producto de software y su relación con los estados del alfa sistema de software

Wilder Perdomo^{1*} Carlos M. Zapata¹

Received: June 20, 2019 Accepted: May 12, 2020

Recibido 20 de junio de 2019 aceptado 12 de mayo de 2020

ABSTRACT

Semat (Software Engineering Method and Theory) is an initiative developed for refounding software engineering by defining a theoretical basis, best practices, and a set of widely-agreed elements. ISO/IEC 25000 (System and Software Quality Requirements and Evaluation-SQuaRE) is an international standard, that evaluates software product quality. ISO 25000 replaces ISO/IEC 9126 and ISO/IEC 14598 standards. Some authors describe the relationship between quality measures of ISO/IEC 9126 and the requirements and software system states of the Semat Essence kernel. Also, they redefine the completion criteria of the activity spaces related to these alphas. Other proposals include a schematic relationship between ISO/IEC 9126 quality measures and the alpha states, but they avoid their correlation and some criteria for evaluating the product quality. Other proposals based on the ISO/IEC 25000 comprise other frameworks than Semat. Our proposal is focused on the selecting the appropriate measures of the software system states to structure the relationships between them, validate them by using a case study, and develop a model for product quality measurement. The results are promising since we can use them to establish a coherent and structural model for evaluating the software system alpha's health and progress.

Keywords: Semat, ISO/IEC25000, software quality, measure, software system.

RESUMEN

Semat (Método y teoría de ingeniería de software) es una iniciativa desarrollada para refundar la ingeniería de software mediante la definición de una base teórica, mejores prácticas y un conjunto de elementos ampliamente aceptados. ISO/IEC 25000 (Requisitos y evaluación de la calidad del sistema y el software-SQuaRE) es un estándar internacional, que permite evaluar la calidad del producto de software. ISO 25000 reemplaza las normas ISO/IEC 9126 e ISO/IEC 14598. Algunos autores describen la relación entre las medidas de calidad de la ISO/IEC 9126 con los estados de los alfas requisitos y sistema de software del núcleo de la Esencia Semat. Además, los autores redefinen los criterios de finalización de los espacios de actividad relacionados con los alfas. Otras propuestas incluyen una relación esquemática entre las medidas de calidad de la norma ISO/IEC 9126 y los estados del alfa, pero evitan su correlación y algunos criterios para evaluar la calidad del producto. Otras propuestas basadas en la ISO/IEC 25000

¹ Universidad Nacional de Colombia. Computer and Decision Sciences Department. Medellin, Colombia.
E-mail: cmzapata@unal.edu.co; wiperdomoch@unal.edu.co

* Corresponding author: wiperdomoch@unal.edu.co

comprenden otros marcos de trabajo diferentes a Semat. Nuestra propuesta se centra en la selección de las medidas apropiadas de los estados del alfa sistema de software para estructurar las relaciones entre ellos, validarlos mediante un estudio de caso y desarrollar un modelo para la medición de la calidad del producto del software. Los resultados son prometedores, ya que podemos usarlos para establecer un modelo coherente y estructural para evaluar la salud y el progreso del alfa sistema de software.

Palabras clave: Semat, ISO/IEC 25000, calidad de software, medida, sistema de software.

INTRODUCTION

The software engineering research community is paying attention to theories in software engineering [11]. This fact implies the need for establishing a common framework allowing software developers to exchange information in decision-making systems [16]. Semat (Software Engineering Method and Theory) is an initiative developed to refound software engineering as a rigorous discipline based on a solid theory, proven principles, and best practices [12, 28]. According to Jacobson *et al.* [11], a method is a composition of practices described in terms of the essential kernel elements. Such elements are called alphas-e.g., opportunity, stakeholders, requirements, software system, work, team, and working way. Each alpha is characterized by using a simple set of states representing its progress and health [13, 19].

ISO/IEC 25000 (SQuaRE-Software Product Quality Requirement and Evaluation) is an international standard for evaluating the software product quality, replacing ISO/IEC 9126 and ISO/IEC 14598 standards [4, 21]. Such a standard includes the following divisions: ISO/IEC 25010 quality model, ISO/IEC 25020 quality measurement, ISO/IEC 25030 quality requirements, and ISO/IEC 25040 quality evaluation [5, 6, 7, 23].

Measures defined in the ISO/IEC 25000 standard can relate to the alpha states of the Semat kernel for evaluating and validating the software product quality in each alpha state. Some studies [29] are intended to describe the relationship between ISO/IEC 9126 metrics and the alpha states of the Semat kernel, focusing on selecting the appropriate indicators for verifying and validating each one of the alpha states. However, such a proposal lacks specification of the relationship between the ISO/IEC 9126 metrics and a new proposal of metrics based on the standard. Furthermore, by identifying criteria to relate the usability to the software system alpha of

the Semat kernel, Perdomo and Zapata [19] show a state-of-the-art review about measurement models of software product quality for identifying some criteria related to the usability and the alpha states of the software system. Zapata *et al.* [27] develop the MetricC game to demonstrate the relationship of each metric with the completion criteria of the activity spaces of the software system alpha.

As mentioned earlier, the proposals schematically relate the ISO/IEC 9126 metrics to the alpha states, but they lack their correlation and criteria for evaluating the product quality. Moreover, they use a different standard than ISO/IEC 25000, which has new terminology, basic measures, and indicators. Nevertheless, companies who certify their products under ISO/IEC 25000 have evaluated such a standard with specific features, supported in different frameworks unrelated to the Semat theory. In this paper, we propose selecting appropriate measures for the software system alpha to assess the product quality and reflect the health and progress of the software engineering endeavor.

This paper is organized as follows. In Section 2, The theoretical framework is presented, including a description of the ISO/IEC 25000 standard for software product quality and a Semat kernel description. In Section 3, we present the state of the art on ISO/IEC 25000 software product quality measures related to the software system alpha. In Section 4, we propose the appropriate measures for the selected alpha of the Semat kernel. In Section 5, we discuss the results of a case study applied for validating the relations found. Finally, in Section 6, conclusions and future work are discussed.

THEORETICAL FRAMEWORK

ISO/IEC 25000

ISO/IEC 25000 (SQuaRE-Software Product Quality Requirement and Evaluation) is an international

standard for evaluating software product quality. This standard replaces the ISO/IEC 91216 and ISO/IEC 14598 standards [4, 21]. Such a standard includes the following divisions: ISO/IEC 25010 quality model, ISO/IEC 25020 quality measurement, ISO/IEC 25030 quality requirements, ISO/IEC 25040 quality evaluation, and ISO/IEC 25050-25099 extension division [5, 6, 7, 23]. According to Abran *et al.* [1], the ISO/IEC develops the series on Software product Quality Requirements and Evaluation (SQuaRE) to improve the interpretation and use of quality measures of software products. SQuaRE is structured as, shown in Figure 1.

In Figure 2, the logical sequence of ISO/IEC 25010 and 25020 divisions is shown. Quality characteristics and sub-characteristics can be quantified by applying measurement functions. A measurement function is an algorithm used to combine quality measurement elements. The results obtained when applying this function are called quality measures. A quality measure represents the quantification of characteristics and sub-characteristics of quality. More than one quality measure should be used for measuring a quality characteristic or sub-characteristic. In a similar way to ISO/IEC 9126, ISO/IEC 25000 has some characteristics to

Figure 1. Organization of the SQuaRE series of international standards [8].

Figure 2. Relationship among quality model, quality measure (QM), quality measure element (QME), property to quality, and target entity [9].

measure the software quality. In ISO/IEC 25000, the characteristics are reorganized for giving more importance to security and compatibility, which are previously considered sub-characteristics in the measurement processes (see Table 1).

Finally, ISO/IEC 25040 has a high level of importance for carrying out the quality assessment process as a way to replace the ISO/IEC 14598 series. It also includes the measurement, evaluation, and total evaluation of product quality [3]. ISO/IEC 25040 includes standards for providing requirements, recommendations, and guidelines for evaluating the software product process [10].

Software engineering method and theory (Semat)

The Semat initiative was launched with the aim of refounding software engineering as a rigorous discipline based on a solid theory, proven principles, and best practices. Also, the language ‘Essence’ and the kernel are defined [11], which constitutes a standard. Semat kernel “helps practitioners to compare software development methods and make better decisions about their practices;” also, “it can be used to analyze the strengths and weaknesses of a team’s way of working” [18]. The kernel has three areas of concern: customer, solution, and endeavor. In the customer area of concern, we understand the stakeholder needs and explore possibilities. In the solution area of concern, we understand the requirements and deploy and operate the system. In the endeavor area of concern, we focus on preparing to do the work, coordinating activities, supporting the team, tracking progress, and stopping work [11]. The kernel includes essential elements to work in every software development endeavor (the so-called alphas). In Table 2, we

show these elements with their names, symbols, and descriptions.

Furthermore, in Figure 3, we show the alphas (opportunity, stakeholder, requirements, software system, work, team, and a way to work; Jacobson *et al.* [12]) with the tasks to be achieved. Alphas help developers understand and evaluate software engineering endeavor’s health and progress [17, 28]. Alphas are different from work products since alphas represent critical indicators of the progress of a software product. Each alpha includes a set of states, which have 100 associated checklists [11]. Semat-in the form of the Essence specification-is being used in the development of the National Quality Plan for China (2012-2020) with the aim of training developers in software development, improving software development practices, and measuring the maturity and the quality of the Chinese software industry, on three levels: enterprise-level, product level, and engineer level [24].

STATE-OF-THE-ART REVIEW

Relationship of ISO/IEC 9126 metrics and the alpha states of the Semat kernel

Zapata and Montoya [29] present the relationship of the ISO/IEC 9126 metrics and the states of the Semat kernel alphas by focusing their proposal on the selection of the appropriate metrics of the alpha states to verify and validate them. Their proposal identifies a lack of specification of the relationship between the ISO/IEC 9126 standard and a new proposal of metrics based on the standard. Additionally, they recommend using the ISO/IEC 25000 standard to relate such metrics to the states of the other alphas of the Semat kernel. They select the ISO/IEC 9126

Table 1. Characteristics and sub-characteristics [8].

Characteristic	Sub-characteristic
Functional suitability	Functional completeness, functional correctness, functional appropriateness.
Performance efficiency	Time behavior, resource utilization, capacity.
Compatibility	Co-existence, interoperability.
Usability	Appropriateness recognizability, learnability, operability, user error protection, user interface aesthetics, accessibility.
Reliability	Maturity, availability, fault tolerance, recoverability.
Security	Confidentiality, integrity, non-repudiation, accountability, authenticity.
Maintainability	Modularity, reusability, analyzability, modifiability, testability.
Portability	Adaptability, installability, replaceability.

Table 2. Graphic elements of the Semat kernel [11].

Name	Symbol	Description
Alpha		An essential element for determining the progress and health of any software engineering endeavor.
Activity space		Activities always carried out in any software engineering endeavor: customer, solution, and endeavor.
Activity		Defines one or more types of work products and tasks and guidelines on how to use them in a practice context.
Practice		The description on how to handle a specific aspect of a development software endeavor.
Activity association		It is used to connect a work product with an alpha or a role. It is also used to connect an activity space with an activity or a phase.
Role/Phase		Either a set of responsibilities or a development process stage.
Work product		Device of value and relevance for a software engineering endeavor.

Figure 3. Alphas of the SEMAT kernel [11].

standard measures, which have changed in the last few years. Furthermore, they lack the criteria to relate the activity spaces, the activities, and the work products of each alpha with the respective measures.

Identification of criteria to relate usability with the software system alpha of the Semat kernel
Perdomo and Zapata [19] show a state-of-the-art review on quality measurement models in software products to identify some criteria to define the

relationship between the usability characteristic of the ISO/IEC 9126 standard with the software system alpha. The authors conclude the identified relationships evidence the need to integrate the concepts and functionality of alpha states with the metrics defined for evaluating software products. The representation of the relationships found is unclear because selection criteria are missing. Furthermore, the authors correlate some essential metrics with usability, and they exclude other characteristics.

Tutorial about Semat initiative and MetricC game
Zapata *et al.* [27] present a tutorial about a Semat-based game called MetricC. They develop the game for showing the relationship of each metric of the ISO/IEC 9126 standard with the completion criterion of the activity spaces for guaranteeing quality and consistency. Moreover, this game allows for interactively visualizing software engineering best practices' importance and operation by using Semat. The authors have a brief specification of which metrics of the standard they are using, which leads to a misunderstanding of the metrics they relate to the activity spaces.

Quality evaluation and security in software products

Mellado *et al.* [15] present the MEDUSAS framework, which independently allows companies to offer a set of software quality assessment and control services based on the ISO/IEC 25000 standard. The authors advocate the quality is treated with more breadth at the level of process quality than at the level of product quality, and the techniques necessary to effectively evaluate the quality and safety of a software product are still underdeveloped. In this framework, they allow for evaluating the quality and security of the source code (software) and the quality and safety of its design. Additionally, they allow evaluate product deliverable quality during some phases of the software development process: analysis, design, and development. However, this paper has some inconsistencies related to the terminology of the ISO/IEC 25000. The authors avoid any relationship between the model and alpha states. The implementation process is also insufficient for identifying the established relations and the work products to be generated.

ISO/IEC 25000 standard and the KEMIS project for its automation with open-source software

Marcos *et al.* [14] show the implementation of the ISO/IEC 25000 standard maintainability characteristics by using open-source software tools. The framework called KEMIS (Kybele Environment Tableurement Information System) provides an infrastructure for measuring and running in continuous integration environments, allowing for automatically and periodically obtaining a set of reports related to the product quality. They also obtain code metrics and micro-architecture based on the ISO/IEC 25000 standard. The restrictions of

KEMIS are based on the maintainability included in the 2502n division. These 165 papers include a combination of measures and attributes from the ISO/IEC 9126 and the ISO/IEC 25000 standard, which are unclear. Also, the authors avoid any relationship among the software system alpha states and maintainability measures.

PROPOSED MEASURES OF THE SOFTWARE SYSTEM ALPHA

This section explains the software system alpha, and then we propose ISO/IEC 25000 measures to be related to this alpha. The software system is one of the Semat kernel alphas. The software system includes software, hardware, and data to provide its main value to the software execution. A software system can be part of the software, hardware, business, and even a more comprehensive social solution. During its development, a software system progresses through several states: architecture selected, demonstrable, usable, ready, operational, and retired. In Figure 4, the states of the software system are shown. Such states provide software system's stability points from its conception to its eventual retirement [11]. According to the ISO/IEC 25000 standard, software product quality can be evaluated by measuring either internal properties (typically static measures of intermediate products) or external properties (typically by measuring the code's behavior when executed). Therefore, the internal measures can be applied to a non-executable software system during its development phases (such as requirements definition, design specification, and source coding), and the external measures can be used to measure the quality of the software product by measuring the system's behavior [7]. Software system alpha can be related to different characteristics and measures of the standard. Such characteristics are functional suitability, performance efficiency, compatibility, usability, reliability, security, maintainability, and portability.

According to the ISO/IEC 15939, a measure is "a variable to which a value is assigned as the result of a measurement." "However, the term 'measure' is used to refer collectively to base measures, derived measures, and indicators" [5]. The identified measures help achieve specific learning goals and make the operation and control of the software system alpha easy. The measurement function is a calculation

Figure 4. Software system states [11].

performed to combine two or more quality measure elements, shown in Table 4. According to ISO/IEC 25023, the measures would inevitably generate somewhat subjective results. In case of difficulties when measuring a ratio scale, a Likert scale can be used as an alternative depending on the situation, 200, e.g., 1.0 for excellent, 0.8 for good, 0.6 for average, 0.4 for poor, and 0.2 for bad [7].

Our proposal focuses on selecting the suitable measures of the ISO/IEC 25000 standard for evaluating the chosen states of the software system alpha to obtain a high-quality software product. We follow some steps to choose each measure and its relation to the software system states:

1. Documenting the software system alpha states and each measure of software product quality (ISO/IEC 25023).
2. Interpreting the measure meanings according to the ISO/IEC 25023 standard and the alpha states with the checklist to relate them.
3. Selecting appropriate measures for evaluating the software product quality in the software system states.
4. Representing the measures and the states of the software system alpha on the Semat kernel

by including the best practices to evaluate the software system's health and progress.

We select all the states to be measured by using ISO/IEC measures. The selection of such states is mainly based on their checklist, as shown in Table 3. The states remind us that "the software system alpha must be shown to be of sufficient quality and functionality to be useful to the users" [11].

We select the states' measures by interpreting the meaning of the measures and relating them to alpha states. Furthermore, we identify internal and external measures suitable to evaluate the alpha software system's health and progress. An internal measure is a "measure of the degree to which a set of static attributes of a software product satisfy stated and implied needs of the software product to be used under specific conditions." An external measure is a "measure of the degree to which a system or software product enables the behavior to satisfy stated and implied needs for the system including the software to be used under specific conditions" [7].

A close relationship with the functional suitability and performance efficiency characteristics and their

Table 3. State checklist of the software system alpha [11].

Alpha state	Items
Architecture selected	i) Architecture selected that addresses key technical risks. ii) Criteria for selecting architecture agreed on. iii) Platforms, technologies, and language selected. iv) Buy, build, and reuse decisions made.
Demonstrable	i) Key architecture characteristics demonstrated. ii) Relevant stakeholders agree that architecture is appropriate. iii) Critical interface and system configurations exercised.
Usable	i) System is usable and has desired characteristics. ii) System can be operated by users. iii) Functionality and performance have been tested and accepted. iv) Defect levels acceptable. v) Release known content.
Ready	i) User documentation available. ii) Stakeholder representatives accept the system. iii) Stakeholder representatives want to make the system operational.
Operational	i) System in use in an operational environment. ii) System available to intended users. iii) At least one example of the system is fully operational. iv) System supported to agreed-on service levels.
Retired	i) System no longer supported. ii) System Updates will no longer be produced. iii) the System has been replaced or discontinued.

Table 4. Measures to the *architecture selected* state of software system alpha.

Measure	Measurement function	Rationale
Functional suitability: Functional coverage	$X = 1 - A/B$ A = Number of functions missing B = Number of functions specified	The proportion of the specified functions and criteria to be used when selecting the architecture has been agreed.
Functional correctness	$X = 1 - A/B$ A = Number of functions that are incorrect B = Number of functions considered	The proportion of functions and criteria provide the correct results. An incorrect function does not provide a reasonable and acceptable outcome to achieve the specifically intended objective.
Functional appropriateness of usage objective	$X = 1 - A/B$ A = Number of functions missing or incorrect among those that are required for achieving specific usage objective B = Number of functions required for achieving specific usage objective	The proportion of the functions and criteria for selecting architecture agreed on by the user provides the appropriate outcome to achieve a specific usage objective.
Performance efficiency: Mean I/O devices utilization	$X = \sum (A_i/B_i)/n_i = 1_{ton}$ Ai = Duration of I/O device(s) busy time to perform a given set of tasks for i-th observation Bi= Duration of I/O operation to perform the tasks for i-th observation n = Number of observations	The degree of amounts and types of resources used by a product or system meets the requirements when performing its functions. Such requirements are platforms, technologies, and language to be used.

The Authors.

measures was found for the architecture selected state. Functional suitability measures “are used to assess the degree to which a product or system provided functions that meet stated and implied needs when used under specified conditions.” Performance efficiency measures “are used to assess the performance relative to the number of resources used under stated conditions.” “Resources can include other software products, the software, and hardware configuration of the system, and materials (e.g., print paper, storage media)” [7].

As a direct measure to evaluate the state, we identify the characteristics of functional suitability and performance efficiency, which belongs to the standard ISO/IEC 25023. Such characteristics have three sub-characteristics, respectively: *functional completeness, functional correctness, and functional appropriateness and time behavior, resource utilization, and capacity*. These sub-characteristics contain four and twelve measures, respectively. However, we select only four measures due to meanings and their direct relationship with the selected state architecture. The appropriate measures were identified for assessing the architecture selected state: *functional coverage, functional correctness, functional appropriateness of usage objective, and mean I/O devices utilization*. We show the complete description of the measures chosen in Table 4.

For the demonstrable state, we find a relationship with the functional suitability and performance efficiency characteristics and some measures as well. Also, this state is related to the compatibility

characteristic and its measures “are used to assess the degree to which a product, system or component can exchange information with other products, systems or components, and perform its required functions, while sharing the same hardware or software environment” [7]. We select four different measures for assessing the demonstrable state: *functional appropriateness of system, transaction processing capacity, data formats exchangeability, and external interface adequacy*. We show the complete description of two measures in Table 5.

On the other hand, we find the usable state has a close relationship with the usability characteristic and its measures. Usability measures “are used to assess the degree to which a product or system can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use” [7]. As a direct measure for evaluating the state, we identify the characteristic usability, which belongs to the standard ISO/IEC 25023. Usability has six sub-characteristics: appropriateness recognizability, learnability, operability, user error protection, user interface aesthetics, and accessibility [7]. Such sub-characteristics contain twenty-two measures. However, we only select six measures in total due to the meanings and their direct relationship with the usable state. We select the following measures: description completeness, self-explanatory user interface, appearance aesthetics of user interfaces, operational consistency, user entry error correction, and understandable categorization of information. We show the complete description of two measures in Table 6.

Table 5. Measures to the demonstrable state of software system alpha.

Measure	Measurement function	Rationale
Functional suitability: Functional appropriateness of system	$X = \sum A_i / n_i = 1 \text{ ton}$ A _i = Appropriateness score for usage objective i, that is, the measured value of functional appropriateness of usage objective for i-th specific usage objective n = Number of usage objectives.	The proportion of the functions and key architecture characteristics were demonstrated and required by the users to achieve their objectives and provide an appropriate outcome.
Performance efficiency: Transaction processing capacity	$X = A/B$ A = Number of transactions completed during observation time B = Duration of observation	The degree to which the maximum limits of a product or system parameter meet the requirements. Such parameters are associated with how the system can be exercised, and its performance can be measured.

The Authors.

Table 6. Measures to the usable state of software system alpha [The Authors]

Measure	Measurement function	Rationale
Learnability: Self-explanatory user interface	$X = A/B$ A = Number of information elements and steps that are presented in a way that the user could understand B = Number of information elements and steps to complete common tasks for a first time user	The proportion of functions is explained in sufficient detail in user documentation and help the facility to enable the user to apply the functions. Also, the release content is known.
User inter- face: Appearance aesthetics of user interfaces	$X = A/B$ A = Number of display interfaces aesthetically pleasing to the users in appearance; B = Number of display interfaces	The degree to which the user interface enables pleasing and satisfying interaction for the user. Also, the added value provided by the systems is clear.

A relationship was found between usability, reliability, and security characteristics and their measures For the ready state. Reliability measures “are used to assess the degree to which a system, product, or components performs specified functions under specified conditions for a specified period of time.” Reliability has four sub-characteristics: maturity, availability, fault tolerance, and recoverability. Security measures “are used to assess the degree to which a product or system protects information and data so that persons or other products or systems have the degree of data access appropriate to their types and levels of authorization.” Security has five sub-characteristics: *confidentiality, integrity, non-repudiation, accountability, and authenticity* [7]. Such characteristics have eleven measures each. We select four measures due to meanings and their direct relation with the ready state. We show two selected measures and their complete description in Table 7.

Furthermore, we find that the operational state has a relation with different characteristics such as usability, maintainability, portability, reliability, and compatibility. Maintainability measures “are used to assess the degree to which a system or computer program is composed of discrete components such that a change to one component has minimal impact on other components.”

Portability measures “are used to assess the degree of effectiveness and efficiency, with which a system, product or component can be transferred from one hardware, software or other operational or usage environment to another” [7].

Nine measures of all sub-characteristics were selected due to their direct impact on the operational state. Such measures are operational consistency, functional customizability, user interface customizability, appearance consistency, system log completeness,

Table 7. Measures to the *ready* state of software system alpha.

Measure	Measurement function	Rationale
Operability: Operational consistency	$X = 1 - A/B$ A = Number of specified interactive tasks that are performed inconsistently B = Number of specified interactive tasks that need to be consistent	The interactive tasks have a consistent behavior and appearance both within the task and across similar tasks. Also, the stakeholders want to make the system operational.
Integrity: Data integrity	$X = 1 - A/B$ A = Number of data items which are actually corrupted by unauthorized access B = Number of data items for which data corruption or modification have to be prevented	The degree to which a system or product prevents unauthorized access to or modifying computer programs or data.

The Authors.

system software environmental adaptability, operational environment adaptability, system availability, and co-existence with other products. We show the complete description of two measures in Table 8.

Finally, for the former state, we find a relationship with the maintainability and portability characteristics. These characteristics contain thirteen and nine measures, respectively. However, we select four measures, which are closer to the state items and their meanings. We identify the appropriate measures for assessing the former state: *modification efficiency, modification capability, functional inclusiveness, and data reusability/import capability*. We show the complete description of the two chosen measures in Table 9.

Figure 5 shows the complete relationship between usable state and measures in the Semat kernel

representation. We used the identified elements in Table 2 for drawing the representation. In part (a) the practice ‘Standard-based quality measurement’ is associated with an activity space called ‘Test the system,’ which is connected through an association with the alpha usable state checklist: (i) *identifying usable and desired characteristics of the software system*; (ii) *operating software system by users*; (iii) *evaluating functionality and performance*; (iv) *evaluating defect levels*; and (v) *releasing software system content* (see Table 3). All of these activities are executed in the development phase. Moreover, the practice associated with the software system alpha of Semat kernel and the work product defect logs, development checklist, and acceptance results are shown in part (b). Each checklist item described in part (a) is connected with the work product identified, linked to the developer by using

Table 8. Measures to the *operational* state of software system alpha.

Measure	Measurement function	Rationale
User interface Customizability	$X = A=B$ A = Number of user interface elements that can be customized; B = Number of user interface elements that could benefit from customization	Users can customize the user interface elements in appearance. At the same time, the system will be available to the stakeholders intended to use it.
Co-existence: Co-existence with other products	$X = A=B$ A = Number of other specified software products with which this product can co-exist; B = Number of other software products specified co-exist with this product in the operation environment	The product can share the environment with other software without adverse impact on their quality characteristics or functionalities. Also, the system is fully supported by the agreed service levels.

The Authors.

Table 9. Measures to the *retired* state of software system alpha.

Measure	Measurement function	Rationale
Modifiability: Modification efficiency	$X = \sum (A_i/B_i)/n$ $n_i = 1$ to n A _i = Total work time spent for making a specific type of modification i; B _i = Expected time for making the specific type of modification i; n = Number of modifications measured	The product can be effectively and efficiently modified without defects or degrading existing product quality. In addition, we can identify if the system has been replaced or discontinued.
Data reusability/import capability	$X = A = B$ A = Number of data which can be used continuously as before B = Number of data which are to be used continuously in the replaced software product	The same data of the product can be used after replacing the previous software product by the own. That means that the developers can reuse or reproduce parts of the coding in another project or the same in another implementation.

The Authors.

Figure 5. Semat representation of (a) standard-based quality measurement (practice) of the software system alpha usable state; (b) standard ISO/IEC 25000, defect logs, development checklist, and acceptance results (work products) to evaluate the alpha states.

the relationship works. Also, in the first item of this state, the developer can select from the Standard ISO/IEC 25000 whatever measures he/she needs to apply during the evaluation of the software product quality.

In Figure 6, we show the relationship between the operational state and measures in the Semat kernel representation. The software system alpha is associated with an extra work product: service levels report and the alpha operational checklist (i) *identifying the system in use in an operational environment*; (ii) *making the system available to intended users*; (iii) *evaluating at least one example of the system is fully operational*; and (iv) *identifying the system supported to agreed-on service levels*. The measures should be selected from the ISO/IEC 25000 standard during the first item.

In Figure 7, we show the relationship between ready state and measures in the Semat kernel representation. The software system alpha is associated with the three-work product: defect

logs, development checklist, and acceptance results and the alpha ready checklist (i) *making the user documentation available*; (ii) *accepting the system by the stakeholders' representatives*; and (iii) *Making system operational for the stakeholders representatives*. The measures should be selected from the ISO/IEC 25000 standard during the first item.

DISCUSSION

In this section, we present and summarize the results of our case study applied to a software development company in Australia for validating the relationships among selected measures and the software system alpha of the Semat kernel. The case study was developed by following the guidelines of Runeson and Host [22] for conducting and reporting case-study-based research in software engineering. This methodology helps evaluate the development methods and observe, explain and explore other phenomena in real-life settings [26]. Thus, we gain a greater understanding of why something happened as it did and what else

Figure 6. Semat representation of (a) standard-based quality measurement (practice) of the software system alpha operational state; (b) standard ISO/IEC 25000, defect logs, development checklist, and acceptance results (work products) to evaluate the alpha states.

Figure 7. Semat representation of (a) standard-based quality measurement (practice) of the software system alpha ready state; (b) standard ISO/IEC 25000, defect logs, development checklist, and acceptance results (work products) to evaluate the alpha states.

might be important for further investigation. Case study research involves an in-depth examination of a single case or a small number of cases. The methodology provides a systematic way of looking at events, collecting data, analyzing information, and reporting results. Commonly, one or more research questions are answered [25]. However, we defined for our validation a research question based on the Goal Question Metric (GQM) method described by Basili *et al.* [2].

The methodology includes six phases: pre-planning, planning, design, data collection, data analysis, and reporting. Using the GQM method, we defined the research goal, research question (RQ), and a metric for the case study:

1. Goal. Identifying the software developer viewpoints for validating a model suitable for evaluating the progress of software system alpha of the Semat kernel based on ISO standards.
2. RQ. What are the software developer viewpoints linked to the relationships among measures and software system alpha states of the Semat kernel?
3. Metric. Correlation of the model for evaluating health and progress of the software system alpha with the expert opinions.

Furthermore, during the case study planning, we defined a clear research proposition: “IF the software developer does not identify some relationships between measures and alpha states THEN the model for evaluating health and progress of the software system will fail”. We need to define a formal case study protocol for validating this proposition, which was designed by following the guidelines and recommendations provided by Pervan and

Maimbo [20] and cited by Runeson and Host [22]. We designed an interview and follow the protocol of the original study (see Table 10). Moreover, we added a consent form signed by both parties (researcher and participant), in which we are committed to keep confidential information provided by the company and the participants.

The main criteria for selecting the participants are:

- i) they should be working in a software development company in Australia.
 - ii) they should exhibit at least 2-year experience; and
 - iii) they should be junior/senior software developers.
- We applied five semi-structured interviews, which were directed towards employees of the software development company. Such interviews were analyzed and summarized as follows.

In Figure 8, we show twenty-eight measures of this case study, which were related to each alpha state: four to architecture selected; four to demonstrable; six to usable; five to ready; five to operational; and four to retired state. The number of measures accepted by the participants was twenty-five (95%), and three were rejected (5%). The measures rejected are general satisfaction, non-vulnerability, and modification efficiency according to the following reasons respectively:

- i) the measure is too general, and it is impossible to review its suitability;
- ii) the measure is not coherent with the usable state, and it could be related to some security attributes more than usability attributes; furthermore, this measure is not directly related to the user interface; and

Table 10. Outline of case study protocol, adapted from Pervan and Maimbo [20].

Section	Content
Preamble	Contains information about the purpose of the protocol, and guidelines for data and document storage and publication.
General	Includes a brief overview of the research project and the case research method.
Procedures	Detailed descriptions of the procedures for conducting each case, including practical details about contacts and timing.
Research instrument	Interview guides, questionnaires, etc. to be used to ensure consistent data collection.
Data analysis guidelines	Detailed description of data analysis procedures, including data schemas, priori codes, etc.
Appendix A	Participation letter.
Appendix B	Interview.
Appendix C	Consent form.

Figure 8. Number of measures accepted and rejected by participants.

iii) the measure is related to the maintainability characteristic. However, the modification efficiency is not directly related to the ready state.

In Table 11, we show the measures selected for software system alpha states and how these are related to each checklist item.

Finally, these results show a positive correlation between the selected measures of the ISO/IEC 25000 standard and the software system alpha states for evaluating the health and progress of the Semat kernel. Also, the proposition defined is accepted and validated by considering participant experience.

CONCLUSIONS

In this paper, we proposed an analysis of the ISO/IEC standard measures for selecting the appropriate measures of the software system alpha to structure the relationships between them. Also, we proposed three Semat kernel representations to show the identified relationships between usable, ready, and operational states with six, four, and nine measures, respectively, and different work products each, which should be implemented by the software developers.

We selected six alpha states-architecture selected, demonstrable, usable, ready, operational, and retired-thirty measures over eighty-six of the ISO/IEC 25000 standard during this research, for helping to evaluate health and progress of software system alpha. The usage of such measures supports the normalization for evaluating software product quality in the Semat kernel and make better decisions about their practices.

We validated the research question and the proposition using a case study, which shows a high correlation (95% accepted and 5% rejected) between the software system alpha states and measures selected during the study. Therefore, we can state some relationships accepted by experts for evaluating the health and progress of the software system alpha states of the Semat kernel with measures from the ISO/IEC 25000 standard.

When we measure the alpha states by using the ISO/IEC 25000 measures, we can guarantee a higher level of quality on requirements and on software systems since such measures are validated and widely accepted for verifying the quality of a software product.

We identified a lack of specification on the relationship between the ISO/IEC 9126 standard and the new

Table 11. Measures selected according to the case study applied.

State	Checklist item	Measure
Architecture selected	i) Architecture selected that addresses key technical risks, ii) Criteria for selecting architecture agreed on, iii) Platforms, technologies, and language selected, and iv) Buy, build, and reuse decisions made.	i) Functional coverage, ii) Functional correction, iii) Mean I/O devices utilization, and iv) Architecture consistency.
Demonstrable	i) Key architecture characteristics demonstrated, ii) Relevant stakeholders agree architecture is appropriate, and iii) Critical interface and system configurations exercised.	i) Functional appropriateness of system, ii) Data formats interchangeability, and iii) Data format consistency.
Usable	i) System is usable and has desired characteristics, ii) System can be operated by users, iii) Functionality and performance have been tested and accepted, iv) Defect levels acceptable, and v) Release content known.	i) Satisfaction with features, ii) Self-explanatory user interface, and iii) Operational consistency.
Ready	i) User documentation available, ii) Stakeholder representatives accept system, iii) Stakeholder representatives want to make system operational.	i) User trust, ii) Failure avoidance, iii) Data integrity, and iv) Record completeness.
Operational	i) System in use in operational environment, ii) System available to intended users, iii) At least one example of system is fully operational, and iv) System supported to agreed-on service levels.	i) Time efficiency, ii) System log completeness, iii) Co-existence with other products, and iv) User accessibility.
Retired	i) System no longer supported, ii) Updates to system will no longer be produced, and iii) System has been replaced or discontinued.	i) Modification capacity, ii) Data reuse/import capability, and iii) update frequently.

proposal of measures based on the standard ISO/IEC 25000. Software system alpha states can be assessed using the ISO/IEC 25000 measures to guarantee a higher-level quality of the software system.

In this paper, we only described the appropriate measures for assessing the software system quality. As future work, we propose to relate the alpha states to quality measures (ISO/IEC 25022) and data quality measures (ISO/IEC 25024). Additionally, we propose the evaluation process under ISO/IEC 25040 standard, related to constraints, inputs, roles, and resources. We also propose to relate measures under the same standard to the requirements alpha of the Semat kernel.

ACKNOWLEDGMENTS

Our project is funded by Universidad Nacional de Colombia. We thank all the reviewers for their valuable feedback.

REFERENCES

[1] A. Abran, R. A-Qutaish, J. Desharnais, and N. Habra. "ISO-Based models to

measure software product quality". The Icfai University Press, India. 2008.

[2] V. Basili, J. Heidrich, M. Lindvall, J. Munch, M. Regardie, and A. Trendowicz. "GQM+ strategies-aligning business strategies with software measurement". Proc. IEEE First International Symposium on Empirical Software Engineering and Measurement-ESEM. Madrid, Spain, pp. 488-490. 2007.

[3] K. Esaki. "System quality requirement and evaluation, importance of application of the ISO/IEC 25000 series". Global Perspective on Engineering Management. Vol. 2 N° 2, pp. 52-59. 2013.

[4] M. Hosni, and V. Kirinic. "Application of software product quality international standards through software development life cycle". Proc. Central European Conference on Information and Intelligent Systems, Varaždin, Croatia, pp. 284-296, 2013.

[5] ISO/IEC-15939. "System and software engineering-software measurement process". 2007.

- [6] ISO/IEC-25010. "Systems and software engineering-systems and software quality requirements and evaluation (SQuaRE) system and software quality model". 2011.
- [7] ISO/IEC-25023. "Systems and software engineering-systems and software quality requirements and evaluation (SQuaRE) measurement of systems and software product quality". 2016.
- [8] ISO/IEC-25000. "Systems and software engineering-systems and software quality requirements and evaluation (SQuaRE) guide to square". 2014.
- [9] ISO/IEC-25021. "Systems and software engineering-systems and software quality requirements and evaluation (SQuaRE) quality measure elements". 2012.
- [10] ISO/IEC-25040. "Systems and software engineering-systems and software quality requirements and evaluation (SQuaRE) evaluation process". 2011.
- [11] I. Jacobson, P.-W. Ng, P.E. McMahon, I. Spence, and S. Lidman. "The essence of software Engineering: applying the SEMAT kernel". Addison-Wesley, Boston, USA, 2013.
- [12] I. Jacobson, I. Spence, P. Johnson, and M. Kajko-Mattsson. "Re-founding software engineering - Semat at the age of three (keynote abstract)". Proc. 27th IEEE/ACM International Conference on Automated Software Engineering, Essen, Germany, pp. 15-19. 2012.
- [13] M. Kajko-Mattsson, M. Striewe, M. Goedicke, I. Jacobson, I. Spence, S. Huang, P. McMahon, B. MacIsaac, B. Elvester, A.J. Berre, "Refounding software engineering: The Semat initiative (invited presentation)", Proc. 34th International Conference on Software Engineering (ICSE), Zurich, Switzerland, pp. 1649-1650, 2012.
- [14] J. Marcos, A. Arroyo, J. Garzas, and M. Piattini. "La norma ISO/IEC 25000 y el proyecto KEMIS para su automatización con software libre". Revista Española de Innovación, Calidad e Ingeniería del Software (REICIS). Vol. 4 N° 2, pp. 133-144. 2008.
- [15] D. Mellado, E. Fernandez-Medina, and M. Piattini. "Security requirements engineering framework for software product lines". Information and Software Technology. Vol. 52 N° 10, pp. 1094-1117. 2010.
- [16] K. Muto, K. Kimita, and Y. Shimomura. "A guideline for product-service systems design process". Procedia CIRP. Vol. 30 N° 2, pp. 60-65. 2015.
- [17] E. Numata, S. Hosono, H. Sakaki, S. Izukura, K. Kimita, and Y. Shimomura. "Disciplines for designing pss actor network". Procedia CIRP. Vol. 30 N° 2, pp. 408-414. 2015.
- [18] Object Management Group. "Kernel and Language for Software Engineering Methods (Essence)". Needham, USA, pp. 1-280. 2014.
- [19] W. Perdomo, and C.M. Zapata. "Identificación de criterios para relacionar la usabilidad con el alfa sistema de software del núcleo de SEMAT". Proc. Latin American Software Engineering Symposium. Bogota, Colombia, pp. 17-22. 2015.
- [20] G. Pervan, and M. Maimbo. "Designing a case study protocol for application in is research". Proc. 9th Pacific Asia Conference on Information Systems-PACIS. Perth, Australia, pp. 1281-1292. 2005.
- [21] M. Rodriguez, P. Oscar, and F. Carlos. "Certificación de la mantenibilidad del producto software: un caso práctico". Revista Latinoamericana de Ingeniería de Software. Vol. 3 N° 3, pp. 127-134. 2015.
- [22] P. Runeson, and M. Host. "Guidelines for conducting and reporting case study research in software engineering". Empirical software engineering. Vol. 14 N° 131, pp. 131-164. 2008.
- [23] F. Schneider, and B. Brian. "A literature survey on international standards for system requirements engineering". INSIGHT. Vol. 17 N° 1, pp. 32-35. 2014.
- [24] SEMAT. "Semat adopted by the china software industry". National Quality Plan. 2014. URL: <https://bit.ly/2DChdMq>.
- [25] J.M. Verner, and L.M. Abdullah. "Exploratory case study research: Outsourced project failure". Information and Software Technology. Vol. 54 N° 8, pp. 866-886. 2012.
- [26] R.K. Yin. "Case study research: design and methods". Thousand Oaks, London, New Delhi. 1994.
- [27] C.M. Zapata, M. Grissa, and C. Luis. "Tutorial sobre la iniciativa Semat y el juego MetriCC". Proc. 8th Congreso Colombiano de Computación. Armenia, Colombia, pp. 1-3. 2013.

- [28] C.M. Zapata, and I. Jacobson. "A first course in software engineering methods and theory". Dyna. Vol. 81 N° 183, pp. 231-241. 2014.
- [29] C.M. Zapata, and Y. Montoya. "On the relationship of ISO/IEC 9126 metrics and the alpha states of the SEMAT kernel". Proc. 4th International Conference in Software Engineering Research and Innovation (CONISOFT). Puebla, Mexico, pp. 59-64. 2016.